COSC 4377 – Networking - Kevin B Long

# interlocking-uh-m-186.eps

Homework #6 v2

Due 11:59pm, Sun 28 abril 2019

Some problems due later, see below

Multiple submissions accepted.

|  |  |  |  |
| --- | --- | --- | --- |
| **Problem** | **Pts Allowed** | **Notes** | **Grade** |
| **1** | 10 |  |  |
| **2** | 10 |  |  |
| **3** | 1 |  |  |
| **4** | 1 |  |  |
| **5** | 1 |  |  |
| **6** | 1 |  |  |
| **7** | 1 |  |  |
| **8** | 1 |  |  |
| **9** | 3.5 | 7 parts |  |
| **10** | 6 | 10 blanks + b + c |  |
| **11** | 2 | 2 parts |  |
| **12** | 2.5 | 5 parts |  |
| **13a** | 4.5 | 9 parts |  |
| **13b** | 4.5 | 9 parts |  |
| **13c** | 2 | 4 parts |  |
| **13d** | 3.5 | 7 parts |  |
| **14** | 1 |  |  |
| **15** | 1 |  |  |
| **16** | 1 |  |  |
| **17** | 1 |  |  |
| **18a** | 1 |  |  |
| **18b** | 1 |  |  |
| **19** | 1 |  |  |
| **20** | 1 |  |  |
| **21a** | 1 |  |  |
| **21b** | 1 |  |  |
| **22** | 1 |  |  |
| **23a** | 3.5 | 7 parts |  |
| **23b** | 12 | 24 parts |  |
| **23c** | 2 |  |  |
| **24a-g** | 7 | 7 parts |  |
| **24h** | 10 | 20 parts |  |
| **Total:** | 100 |  |  |

Name: James Richardson

Peoplesoft ID: 1555520

1. Complete the 7th Socket lab on Traceroute
2. Complete the 10th Wireshark lab on Network Address Translation
3. What are the names of the units of information we learned for each of the 5 layers?

Application: message

Transport: Segment

Network: datagram

Data Link: frames

Physical: bits

1. Which layers have each of the following characteristics? Mark the answer true if there is at least one popular protocol within that layer with that feature that we discussed in class and in the book:

Flow Congestion Data Error

Control Management Retransmission Detection

Application ☐ ☐ ☐ ☐

Transport x x x x

Network ☐ ☐ ☐ x

Data Link ☐ ☐ ☐ x

Physical ☐ ☐ ☐ ☐

Network kinda has flow control, but not really: It will break up packets that are too large to go through particular layer 2 devices into multiple fragments.6

The following questions come from Chapter 3

1. Why do some streaming services and popular apps send voice and video traffic over TCP even though we learned UDP was often better? (Hint : the services are not looking to take advantage of TCP’s features)

Because system administrators often block UDP in the firewall because people try to use it to DOS hosts.

1. Is it possible for an application to enjoy reliable data transfer even when the application runs over UDP? If so, how?

**xYes ☐No How?** You can implement the ‘reliable data transfer’ in the application layer using UDP, just re request packets that are dropped rather than ignoring them.

1. In TCP, which of the following do sequence numbers provide? Mark all that apply.

x Tells us how may bytes have been sent since connecting

x Distinguishes between original and duplicate transmissions

☐ Re-orders bytes into the same sequence that the application sent

☐ Assigns packet numbers

☐ Lets you know you’re missing fragments

☐ Tells us how many hops are left

x Lets you know you’re missing bytes

☐ To reassemble fragmented packets

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. In TCP, why did we need to introduce timers, and precisely do we time?

We introduce timers to know if we need to re transmit a packet. If we go long enough without receiving an ACK back, we assume that the packet was lost and need to re transmit.

1. True or false?
   1. ☐T **xF** Host A is sending Host B many packets over TCP. Host B doesn’t have data to send to A; only acknowledgments. Host B won’t send acknowledgments to Host A because it has no place to put them.
   2. ☐T **xF** Once established at the beginning of a connection, the size of the TCP variable rwnd is fixed (doesn’t change).
   3. **xT** ☐F Host A is sending a large file to B. A’s window size is the smaller of what TCP calculates, and the value of the receiver’s rwnd variable.
   4. ☐rT **xF** Suppose Host A is sending a large file to Host B over a TCP connection. If one segment has sequence number m, then the next segment will be m+1.
   5. **x T** ☐F The TCP segment has a field in its header for rwnd .
   6. **xT** ☐F Suppose that the last SampleRTT in a TCP connection is equal to 1 sec. The current value of TimeoutInterval for the connection will necessarily be ≥ 1 sec.
   7. **xT** ☐F In the midst of a stream of TCP segments travelling between Hosts A and B, A sends one with sequence number 50 and 16 bytes of data. The number in the Acknowledgment field in the next packet from B back to A will always be 66.
2. The following series of packets is sent sequentially as part of the same pair of TCP connections from A to B.
   1. Fill in the missing blanks. Each row represents a packet sent from A to B and one sent from B to A in response. Assume all packets are sent in order and are part of a single, continuous bytestream (no gaps).

*If you have no way to determine the size of a packet otherwise, default to “300” bytes sent.*

Starting seq # is 0 for A

Starting seq # is 10,000 for B

Both use 14-bit sequence numbers

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A sends (in bytes)** | **A’s seq #** | **B’s ack # to A** | **B sends (in bytes)** | **B’s seq #** | **A’s Ack # to B** |
| 180 | 0 | 180 | 400 | 10000 | 10400 |
| 140 | 320 | 320 | 230 | 10400 | 10630 |
| 90 | 320 | 410 | 300 | 10630 | 10930 |

* 1. After how many bytes of transmission will the sequence numbers “roll over” back to their starting point?

Max Sequence Number: 2^14 == 16384 == 2048 bytes

* 1. Given that, what’s the maximum number of bytes that can safely be sent in a single window regardless if the network and recipient had infinite capacity? Review the text around figure 3.27 for an example of what happens when your window size is too close to the size of sequence numbers.

2048 / (num outgoing packets +1)

We need a number such that we won’t roll over and loose track of if we are talking about an older or more recent packet. The equation above takes care of that.

1. Suppose Host A sends two TCP segments back to back to Host B over a TCP connection. The first segment has sequence number 2620; the second has sequence number 3100.
   1. How much data is in the first segment?

3100 – 2620 = 480 bytes

* 1. Suppose that the first segment is lost but the second segment arrives at B. In the ACK that Host B sends to Host A, what will be the acknowledgment number?

2620, the next byte that it expects to receive

The following questions come from chapter 4

Consider the switch shown below. Input cards are on the left, output cards on the right. There are buffers on both sides. In each time slot, each output card can accept one packet. And in each time slot, an input card can send one packet. The task, of course, is to move everything out of the input side to the output side in as few time slots as possible.

1. Suppose you have the five packets as shown below, with X, Y and Y at the front of their cards’ queues. Without modification, how many time slots are required to drain the input card queues?

3 time slots [(x, y), (x, y), (z)]

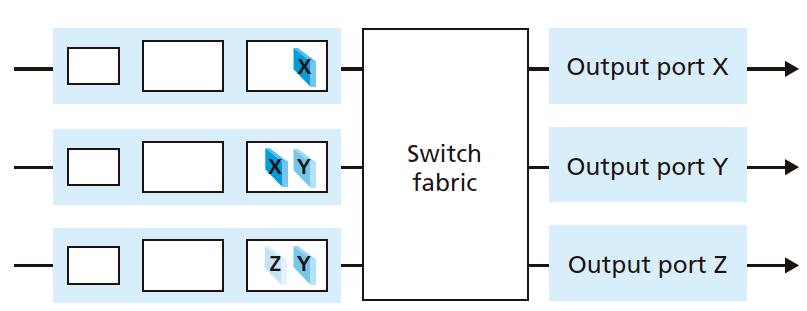
1. If you can reorder the packets in the input cards, what’s the fewest number of slots you find to transmit everything?

2 time slots [(x, y, z), (x, y)]

1. If you change the packet “Y” in the last queue to an “X”, what’s the number of slots required to empty all three input buffers?

4 time slots

[(x, y), (x), (x), (z)]



1. What is the formula for the minimum number of time slots required to transfer *n* packets in any combination and order you choose, in equal numbers in each queue, being sent from *p* input to *p* output ports?

Minimum number = p, if they can all be sent in parallel

1. What’s the formula for the maximum number of slots you could devise that would be required to transferred *n* packets in any combination and order you choose, sent from *p* input to *p* output ports?

Minimum number = n, if they all have to be sent sequentially.

1. Consider a datagram network using 32-bit host addresses. Suppose a router has four links, numbered 0 through 3, and packets are to be forwarded to the link interfaces as follows:

|  |  |
| --- | --- |
| **Destination Address Range** | Link Interface |
| 01110010 01011111 00000000 00000000 |  |
| through | 0 |
| 01110010 01011111 01111111 11111111 |  |
|  |  |
| 01110010 01011111 10000000 00000000 |  |
| through | 1 |
| 01110010 01111111 11111111 11111111 |  |
|  |  |
| 01110010 10000000 00000000 00000000 |  |
| through | 2 |
| 01110010 11111111 11111111 11111111 |  |
|  |  |
| 01110010 10000001 01010100 00000000 |  |
| through | 3 |
| 01110010 10000001 01010111 11111111 |  |
|  |  |
| otherwise | 4 |
|  |  |

1. How many bits in the each range are identical? This determines your subnet mask. The bits that match are part of the network number, and the rest are to be ignored when looking for matches with incoming packets. For example, the first range matches all possible binary numbers that begin with its first 17 bits, so its mask in what we call “slash notation” is /17. Any packet that arrives at the router with those same first 17 bits is a match to that entry, and should be forwarded to port 0 (according to the table above).

The other way to write that mask is write down your 32-bit IP address, setting the first 17 digits to 1’s, and the remainder to 0’s. That’s 11111111 11111111 10000000 00000000. Converting that to “dotted decimal” notation gets you 255.255.128.0. All three forms are equivalent.

Complete the table of subnet masks below:

|  |  |  |  |
| --- | --- | --- | --- |
| **Range** | **Slash** | **Binary** | **Dotted Decimal** |
| **0th** | /17 | 11111111.11111111.10000000.00000000 | 255.255.128.0 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

1. To build a forwarding table, we apply this mask to any address in the range we were given by AND’ing it (keep the bits where the mask is a 1, clear the rest).

For example, the first range includes a lot of IP addresses, but one of them is:

01110010 01011111 01011000 00110011.

Applying the mask keeps the first 17 bits but clears the rest:

01110010 01011111 0~~0000000 00000000~~.

This is the **network number** for this group of IP addresses. Perhaps a more convenient representation would be in dotted decimal: 114.95.128.0/17 or just 114.95.128/17, since the network bits are entirely within the first three octets.

Using your masks from above, extract the network numbers from the remaining ranges:

|  |  |  |  |
| --- | --- | --- | --- |
| **Range** | **Network # in Binary** | **In Dotted Decimal** | **In Short Decimal** |
| **0th** | 01110010 01011111 00000000 00000000 | 114.95.128.0/17 | 114.95.128/17 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

1. Now build your final forwarding table. Let’s use the short decimal format with the slash notation following it in one column, and the link interface in the other. You must always have a default route that matches anything without a more specific entry. I’ve added that for you at the end.

|  |  |
| --- | --- |
| **Destination Address Range** | Link Interface |
|  | 0 |
|  | 1 |
|  | 2 |
|  | 3 |
| \* (default) | 4 |

1. When a packet with the following destination IP addresses appears at the router, to which link interface will the router decide to send it? Always choose the entry from your routing table that matches more of the bits than any other entry, the so-called **longest-matching prefix**. Hint: it is usually easier to convert decimal addresses to binary and compare them with the binary versions of your network numbers from above.

\_\_\_\_ 114.242.117.0

\_\_\_\_ 01110010 01011111 01101011 11001101

\_\_\_\_ 114.92.214.33

\_\_\_\_ 01110011 10000001 01010101 00111100

\_\_\_\_ 114.129.85.114

\_\_\_\_ 01110010 01011111 10010001 01110111

\_\_\_\_ 01110010 10000000 11111111 11111111

1. What is stored in each of a high-speed router’s input cards to enable fast forwarding decisions?

Each of the input cards will have its own copy of the routing table to enable high speed routing. Additionally each input card will have its own queue.

1. In Section 4.2, we studied FIFO, Priority, Round Robin (RR), and Weighted Fair Queueing (WFQ) packet scheduling disciplines? Which of these queueing disciplines ensure that all packets depart in the order in which they arrived?

Only FIFO will ensure that packets leave in the same order that they arrive. All of the other queuening techniques will require some sort of ordering (based on weight or giving everyone a time slot),

1. When a host received an IP datagram, what field in the IP header tells IP to which layer-four protocol the payload should be delivered (e.g. to UDP vs TCP)?

Byte offset 10 which contains the protocol used as an 8 bit value



1. What field keeps packets from being forwarded around in endless loops or paths?

Time to live, byte offset 9. 8 bit value that determines how many more hops this particular packet can make before it should be dropped and no longer forwarded to the next location.

1. IP datagrams can be fragmented into multiple new datagrams at any point along the way from sending host to receiving host.
   1. In what device are fragments reassembled?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* 1. Why is it impossible to depend on them being able to be reassembled at a different point along the transmission path?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. A router has eight interfaces. How many IP addresses will it have?

Each interface has its own ip address, each ip address is connected to its own lan. Since there are 8 interfaces, the router will have 8 ip addresses.

1. What is the 32-bit binary equivalent of the IP address 202.3.14.25? Write in groups of 4 binary digits (e.g. 0101 1010 1010 0101, etc.):

202.3.14.25

0xCA 0x03 0x0E 0x19

1100 1010 0000 0011 0000 1110 0001 1001

1. Suppose there are four routers between a source host and a destination host.
   1. Ignoring fragmentation, an IP datagram sent from the source host to the destination host will travel over how many interfaces?

Input and output interface for each of the routers. So 2 \* number or routers == 8

* 1. How many forwarding tables will be indexed to move the datagram from the source to the destination?

Only one for each input interface, the routing table tells the router which interface to send the packet out of: So 4

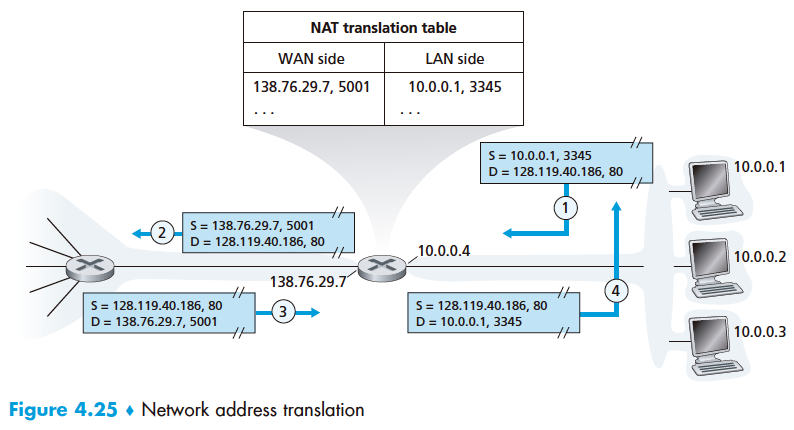
1. Suppose datagrams are limited to 1,500 bytes (including header) between source Host A and destination Host B. Assuming a 20-byte IP header, how many datagrams would be required to send an MP3 consisting of 5 million bytes? Explain how you computed your answer.

We could also have to consider the header / footer of tcp or udp but this is not given so we will ignore it.

Since 20 bytes is for the ip header, we have 1480 bytes to work with.

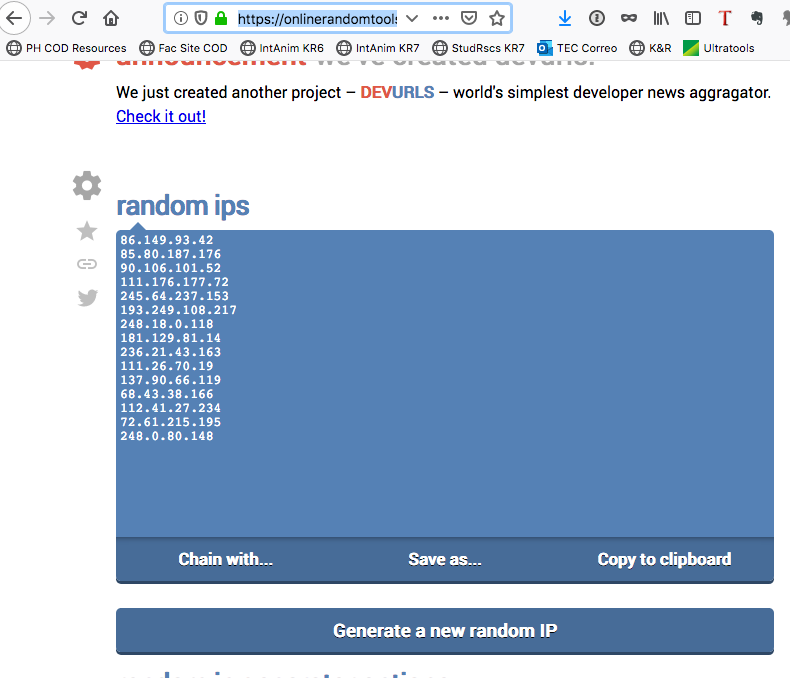
Because of this, the number of datagrams needed is equal to ceil(5000000 / 1480) = ceil(3378.3783783783783) = 3379

!6



1. Consider the network setup in Figure 4.25. Suppose that the ISP instead assigns the router a different public IP address (the one on the left side of the router) and that the network address of the home network is also different.

Go to <https://onlinerandomtools.com/generate-random-ip> and generate a list of random IPs. Copy and paste the block as a photo showing the URL here. Here’s mine you can delete:



You may delete my image above if you like, but then

<paste your image here>

* 1. Pick the first addresses from the list and assign it to the interface on the border router facing the Internet. Take the second address in the list, convert it to a /24 network address, and then assign addresses from that subnet to each interface on the LAN.



You may either label the diagram above, or fill in the table below:

Chosen LAN Network number: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /\_\_\_\_\_

Router LAN-side Interface IP Address: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Client 1 IP Address: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Client 2 IP address: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Client 3 IP address: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Router WAN-side Interface IP Address: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* 1. Suppose each host has two ongoing TCP connections, all to port 80 at the host with the third IP address from your list. NAT is configured to begin assignments at port 4000 and continue sequentially. The clients are running an operating system that assigns ports beginning with 2300 for its connections.

Armed with this information, fill in the six entries in the NAT translation table for the six TCP connections.

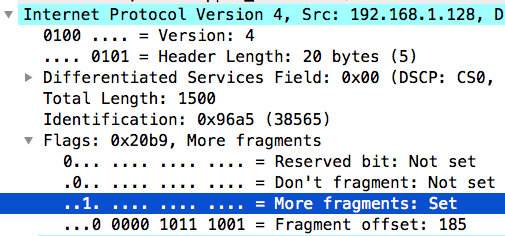
|  |  |  |  |
| --- | --- | --- | --- |
| **NAT translation table** | | | |
| WAN Side | | LAN side | |
| IP | Port | IP | Port |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

* 1. If the three PCs were running the same pair of client programs and the clients used well-known port numbers (this happens with some protocols), how would it change the entries in your translation table?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Look at the pcap file “**large ping trace.pcap**” that’s in the homework folder using Wireshark or a similar program. Find any series of frames that contain IP fragments – there are many. You can find them by expanding the IP window to show the flags and then scrolling through the captured frames until you find some with a “1” in the “More Fragments” bit like this one:



After an original IP datagram is created by a host, at any point along the way it may have to be fragmented if the Maximum Transmission Unit (MTU) for the next layer 2 link indicates it cannot handle a payload as large as this datagram.

* 1. What do you observe to be the MTU for this network? \_\_\_\_\_\_\_\_\_\_\_
  2. When a packet is reassembled from its fragments (which you will see in Wireshark), what is the relationship between the fragments’ IP identification numbers?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Examine the IP header for packet 49.

* 1. What is the Total Length of the IP header + payload? \_\_\_\_\_\_\_\_
  2. What is the Header Length? \_\_\_\_\_\_\_
  3. How much payload space is left to send data from the transport layer? \_\_\_\_\_\_\_\_\_
  4. What was the size of the payload in the prior packet, #49? \_\_\_\_\_\_\_
  5. What field in packet 50 tells us where packet 50’s payload should be placed in respect to the other fragments so as to put the pieces back together in the correct order, and what’s its value?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* 1. Complete the following table to show how you would use this network each with 20 bytes of header, to transmit am original 5000-byte IP datagram. The value of the identification field in the original datagram is 0x49d3. You may not need all rows.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Frag #** | **Data Length** | **ID** | **MF Flag** | **Offset** | **Data Still to Send** |
| **1** |  |  |  |  |  |
| **2** |  |  |  |  |  |
| **3** |  |  |  |  |  |
| **4** |  |  |  |  |  |
| **5** |  |  |  |  |  |
| **6** |  |  |  |  |  |