Problem Set 5

Multicast, IP Fragmentation, CIDR, Networking Utilities

Due: start of class, Wednesday, April 3rd

All problems carry equal weight. Follow the instructions on Piazza to add and commit new files.

1. IP Fragmentation

Consider two hosts, A and B, each on a separate shared Ethernet with MTU=1200 bytes. In addition to these LAN's, the route connecting host A to host B through the Internet contains an additional hop over a point-to-point link between a router on A's Ethernet and a second router on B's Ethernet. The point-to-point link has MTU=600 bytes. Recall that MTU is the maximum amount of data that can be sent in a frame at the physical layer and thus includes all TCP and IP headers (each of which occupies 20 bytes). Also recall that IP fragmentation breaks data along 8 byte boundaries.

a. An application on host A passes 2000 bytes of data to TCP. Following the approach used to draw Figure 4.4 (page 243 of P&D) but including the TCP header, sketch the packets that cross each link in the route. How many bits are delivered to the network layer protocol at host B?

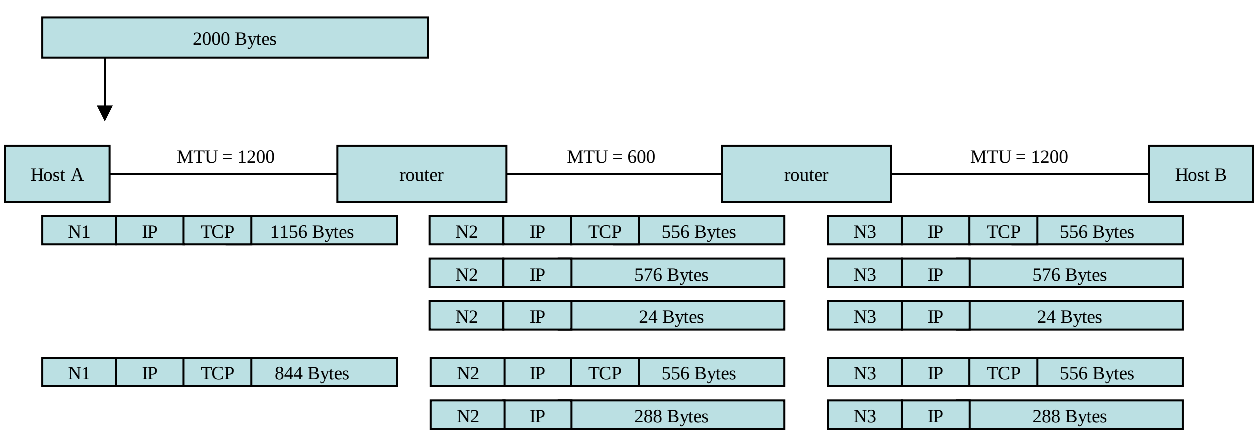
b. If the probability that any IP datagram crossing any link arrives intact (without error) is given by *p*, calculate the probability that the entire 2000 bytes sent in part (a) arrives without the need for retransmission.

c. Calculate the average amount of data, including TCP and IP headers, and including all transmissions and retransmissions, that must be sent by host A in order to successfully deliver the 2000 bytes to an application on host B given $p = 9/10$, where p is as defined in part (b). Assume that host B will buffer any data that arrives at the TCP level.

d. Most IP datagram reassembly algorithms have a timer to avoid having a lost fragment tie up reassembly buffers forever. Suppose a datagram is fragmented into four fragments. The first three fragments arrive, but the last one is delayed. Eventually the timer goes off and the three fragments in the receiver's memory are discarded. A little later, the last fragment stumbles in. What happens to this last fragment at the receiver?

Sol:

a.



The network layer receives 5 IP headers, 2 TCP headers and 2000 bytes of data = 2140 bytes.

b. The probability of success with no retransmissions is equal to the probability that none of the fragments gets lost = p^{12}.

c. When IP reassembly fails due to fragment loss, retransmission occurs at the TCP level. The first TCP packet has a probability of p^7 to be reassembled per transmission, so $1/p^7$ = 2.09 transmissions are necessary on average. The second TCP packet succeeds with probability p^5, so only 1.69 transmissions are necessary on average. Host A must send an average of (1196 bytes x 2.09) + (884 bytes x 1.69) = 2499.64 + 1493.96 bytes = 3993.6 bytes.

d. As far as the receiver is concerned, this is a part of new datagram, since no other parts of it are known. It will therefore be queued until the rest show up. If they do not, this one will time out too.

2. IP Fragmentation

Suppose an IP packet is fragmented into 15 fragments, each with a 1.5% (independent) probability of loss.

a. What is the probability of losing the whole packet due to loss of a fragment?

Based on the previous answer, what is the probability of net loss of the whole packet if the packet is transmitted twice,

b. assuming all fragments received must have been part of the same transmission?
c. assuming any given fragment may have been part of either transmission?
d. Explain how use of the *Ident* field might be applicable here.

Sol:

a. $\binom{1}{15} * 1.5\% = 0.225$

b. *The probability of losing both transmission of the packet would be* $0.225 * 0.225 = 0.050625$.

c. *The probability of loss is now the probability that for some pair of identical fragments, both are lost. For any particular fragment the probability of losing both instances* $1.5\% * 1.5\% = 0.000225$, *and the probability that this happens at least once for the 15 different fragments is thus 1-(1-0.000225)^15 = 0.00337.*

d. *An implementation might (though generally most do not) use the same value for Ident when a packet had to be retransmitted. If the retransmission timeout was less than the reassembly timeout, this might mean that case (c) applied and that a received packet might contain fragments from each transmission.*

Comments: Since "due to loss of a fragment" has some ambiguity, the other set of answers is also acceptable:

a. *1- 0.985^15 = 0.202844*
b. *0.202844^2 = 0.041146*
c. *Same with the above one*
d. *Same with the above one*

3. Forwarding and Classless Inter-domain Routing (CIDR)

a. Consider a router that interconnects three subnets: Subnet A, Subnet B and Subnet C. Suppose all of the interfaces in each of these three subnets are required to have the prefix 233.1.15/23. Also suppose that Subnet A is required to support up to 120 interfaces, and Subnets B and C are each required to support up to 61 interfaces. Provide three network addresses (of the form a.b.c.d/x) that satisfy these constraints.

Suppose a router has built up the routing table shown below. The first four lines are for CIDR addresses, with “/22” indicating a mask of 22 1's followed by 10 0's.

Net/Masklength	NextHop
128.174.240.0/20	R1
128.174.240.128/25	Interface 1
128.174.240.17	Interface 2
128.174.252.0/22	R3
128.174.240.16/29	Interface 3
128.174.248.0/22	R2
default	R4

b. How many individual IP addresses match each of the four Net/Masklength pairs? (For simplicity, ignore the fact that there is some overlap).

c. The router can deliver packets directly over interfaces 0, 1 or 2, or it can forward to routers R1,R2, R3 or R4. Specify the next hop for each of the following destinations. Remember that if a destination matches more than one line of the table, the longest match is used.

i. 128.174.240.17
ii. 128.174.245.17
iii. 128.174.250.17
iv. 128.174.254.17
v. 128.174.225.17
vi. 128.174.240.18

sol

a.

A: 233.1.15.0/25
B: 233.1.15.128/26
C: 233.1.15.192/26

b.

Net/Masklength	Number of Hosts
128.174.240.0/20	1910
128.174.240.128/25	126
128.174.240.17	1
128.174.252.0/22	1022
128.174.240.16/29	5
128.174.248.0/22	1022
default	-

c.

i. 128.174.240.17 ⇨ I2
ii. 128.174.245.17 ⇨ R1

iii. 128.174.250.17 ⇨ R2
iv. 128.174.254.17 ⇨ R3
v. 128.174.225.17 ⇨ R4
vi. 128.174.240.18 ⇨ I3

4. Networking Utilities

Use the EWS machines for this problem. Show the commands that you use to solve the problem and the output you get. You have to show your work to get full credit.

The **`ifconfig`** utility is used to assign and examine network interface parameters. Read the man page on ifconfig and study the different options. Use the command to determine the Internet broadcast address for a EWS machine that you have logged on.

The **`arp`** utility can be used to display the Internet-to-Ethernet address translation table in the host it is running on. Consult the man page for arp, and use it to determine the Ethernet address of 3 EWS machines. (Hint: Since entries in the address table time-out, the table may be empty sometimes. To fill out the table, send a ping to the broadcast address you obtained in part (a). The host will inspect the return ping packets to build up its address translation table.)

Sol:

a. Typing `ifconfig -a` on host `linux5`

```
eth0      Link encap:Ethernet  HWaddr A4:BA:DB:30:39:E9
          inet addr:130.126.112.148  Bcast:130.126.112.255  Mask:255.255.255.0
          inet6 addr: fe80::a6ba:dbff:fe30:39e9/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1058467718 errors:0 dropped:69 overruns:0 frame:0
          TX packets:1239927215 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:307274034895 (286.1 GiB)  TX bytes:799283125534 (744.3 GiB)
          Interrupt:36 Memory:d8000000-d8012800

eth1      Link encap:Ethernet  HWaddr A4:BA:DB:30:39:EB
          inet6 addr: fe80::a6ba:dbff:fe30:39eb/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:7494153 errors:0 dropped:0 overruns:0 frame:0
          TX packets:10 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:806393494 (769.0 MiB)  TX bytes:1876 (1.8 KiB)
          Interrupt:48 Memory:da000000-da012800

eth2      Link encap:Ethernet  HWaddr A4:BA:DB:30:39:ED
          inet6 addr: fe80::a6ba:dbff:fe30:39ed/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:478434 errors:0 dropped:0 overruns:0 frame:0
          TX packets:10 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:61447405 (58.6 MiB)  TX bytes:1876 (1.8 KiB)
          Interrupt:38 Memory:c7000000-c77fffff

eth3      Link encap:Ethernet  HWaddr A4:BA:DB:30:39:EF
          inet6 addr: fe80::a6ba:dbff:fe30:39ef/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:478432 errors:0 dropped:0 overruns:0 frame:0
          TX packets:12 errors:0 dropped:0 overruns:0 carrier:0
```

```
          collisions:0 txqueuelen:1000
          RX bytes:61446713 (58.6 MiB)  TX bytes:2568 (2.5 KiB)
          Interrupt:45 Memory:c7800000-c7ffffff

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:1278946375 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1278946375 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:305103863398 (284.1 GiB)  TX bytes:305103863398 (284.1 GiB)
```

b. arp -a

```
engr-cpanel-mysql.engr.illinois.edu (130.126.112.113) at 00:50:56:bd:00:8b [ether] on eth0
engr-linux.engr.illinois.edu (130.126.112.126) at 00:50:56:bd:43:c3 [ether] on eth0
engr-wiki-prod.engr.illinois.edu (130.126.112.135) at 00:50:56:bd:6d:8a [ether] on eth0
engr-subversion.engr.illinois.edu (130.126.112.120) at 00:50:56:bd:00:3c [ether] on eth0
engr-courses-li.engr.illinois.edu (130.126.112.115) at 00:50:56:bd:00:28 [ether] on eth0
0015-enghdata-net.gw.uiuc.edu (130.126.112.1) at 00:12:f2:b2:b6:00 [ether] on eth0
```