EECS 122: Introduction to Communication Networks

Homework 3 Solutions

Solution 1.

a) We find out that one-layer subnetting does not work: indeed, 3 departments need 5000 host addresses, so we need 13 bits ($2^{12} = 4096 < 5000$, so 12 bits are not enough) for the host part of the subnetwork. Since we have 16 bits available (it's a class B address), this leaves only $16 - 13 = 3$ bits for the subnetwork ID part, which can accommodate only $2^3 = 8$ departments.

So we have to use two-layer subnetting, that is, divide some subnetworks into subsubnetworks. At the first layer, we use 8 subnetworks of $2^{13} = 8192$ hosts each. Three of these subnetworks are allocated to the three departments that need 5000 hosts each. For example, department 1 gets the subnetwork with ID 001, department 2 gets the subnetwork with ID 010 and department 3 gets the subnetwork with ID 011. Department 1 can then number its hosts 001xxxxxxxxxxxxx, where the x's are either 1 or 0.

We are left with $8 - 3 = 5$ subnetworks of 8192 hosts each. We need $600 < 1024 = 2^{10}$ host addresses for each of the 7 small departments, so we can divide one of the 8192-host subnetworks into $2^3 = 8$ subsubnetworks of $2^{10} = 1024$ hosts each. For example, we can divide the subnetwork with ID 100, and let 100001 be the subsubnetwork ID of department 4, 100010 be the subsubnetwork ID of department 5, 100011 be the subsubnetwork ID of department 6, and so on. Department 10 can then name its hosts 100111xxxxxxxxxx, where the x's are either 1 or 0.

We are still left with 4 subnetwork IDs (101–111) for future use.

(In real life it's slightly more complicated: address fields that are all 0s or all 1s are reserved, so subnetwork 100 can have only 6 subsubnetworks, 100001 through 100110. For Department 10 we could divide subnetwork 101 the same way, and give it subsubnetwork 101001.)

b) If we are using class C network addresses, then for the departments with 5000 hosts we need $2^{(13-8)} = 32$ class C addresses per department. For the departments with 600 hosts we need $2^{(10-8)} = 4$ class C addresses per department. The total number of class C network addresses needed is $3 \cdot 32 + 7 \cdot 4 = 124$.

Solution 2. In this case packet format is $[s, d \mid S, D \mid data]$, where s, d are Ethernet addresses (of course, the Ethernet header also includes other fields not relevant to this problem) and S, D are IP addresses (of course, the IP header also includes other fields not relevant to this problem). Note that since B and C are switches, all nodes are in fact on one local network, so the link layer addresses remain the same throughout.

Solution 3.

a) Assuming full-duplex interfaces, the maximum total transmission rate is achieved when all the nodes are sending and receiving data at the line rate, 100 Mbps. This is achieved when no more than 100 Mbps of data is destined to a single node (for example, when distinct pairs of nodes are communicating to each other). The resulting maximum total transmission rate over the network is thus $16 \cdot 100$ Mbps $= 1.6$ Gbps. For half-duplex interfaces, data can only be flowing in one direction at a time on each link, so the answer is half as much (800 Mbps).

b) Consider the case where 15 nodes want to transmit data to the last one. In that case the total transmission rate is limited, in the best case, to the line rate of the last node, which is 100 Mbps.

Solution 4.

a) The maximum transmission rate from the server is 100 Mbps, but the maximum transmission rate to all the computers is $16 \cdot 10$ Mbps $= 160$ Mbps. Hence the achievable rate is simply $\min(100 \text{ Mbps}, 160 \text{ Mbps}) = 100$ Mbps.

b) In this case, the maximum transmission rate from the server is 10 Mbps, while the rate to the computers remains the same. Hence the achievable rate is $\min(10 \text{ Mbps}, 160 \text{ Mbps}) =$ 10 Mbps. We can see that in both cases the achievable rate is limited by the link to the server for this configuration.

c) Once the switch starts forwarding the packet to the server, it must always have bits to send, otherwise there is going to be a gap in the transmission of the packet, and the packet will be lost. The simplest (and in fact, best) solution is for the switch to completely receive the whole packet before it begins forwarding it to the server.

A more complicated, less robust, and thus worse solution would be the following. The switch figures out the length of the packet (from the header), say L bits. The switch waits until it has received at least $\frac{9}{10}L$ bits (nine tenths of the packet) before it starts forwarding the packet to the server. Since the incoming rate is $1/10$ the outgoing transmission rate to the server, the buffer of the switch will not be drained until the whole packet is received.

Solution 5. ARP tables translate the network address of a node to its Ethernet address. For computer t such a pair would be [(c.k:u)], which is the entry that appears in ARP1 for subnet c. Similarly, the entries in ARP2 for subnet b are [(b.i:z),(b.s:w)], corresponding to computers n and r respectively.

Solution 6. Suppose there are $K-1$ levels in the domain hierarchy, root domain being level 1, subroot being level 2, and so on until level K (the leaves of the tree, that is, hosts). Let every name domain at level i, $1 \leq i < K$, have L_i subdomains (domains at level $i+1$). Then each name server for a domain at level i, $1 \leq i < K$, needs L_i entries, one for the name of each subdomain.

The total number of names needed is

$$\prod_{i=1}^{K-1} L_i = L_1 \cdot L_2 \cdots L_{K-1}$$

This product must be equal to the total number of names, 10^9. Therefore, if $K=2$, then $L_1 = 10^9$. If $K=3$, $L_1 \cdot L_2 = 10^9$, and assuming $L_1 = L_2$, we get $L_1 = L_2 = 10^{4.5}$. If $K=10$, we only need 10 entries per server. We see that the greater K is (the deeper the tree, the more levels in the hierarchy) the smaller the number of entries each name server needs. On the other hand, the deeper the tree, the larger the look-up time of a name becomes: for a tree of depth K, the worst-case look-up time is on the order of $K+1$ (see problem 7).

Solution 7. Let p be the probability of a name to be found in the local server. Let K be the number of levels, L_i be the number of entries at level i, and $L = \prod_{i=1}^{K-1} L_i$ (see problem 6). Then the expected lookup time is:

$$p \cdot (\text{time for local lookup}) + (1-p) \cdot (\text{time for non-local lookup})$$

The time taken for a local lookup is $1 + \alpha \log(L_{K-1})$, since local servers are in level $K-1$ of the hierarchy (level K is hosts). A non-local lookup consults K servers: the local server, plus $K-1$ servers from the root to the $K-1^{\text{st}}$ server, which is responsible for the non-local name. So the time taken for a non-local lookup is

$$(1 + \alpha \log(L_{K-1})) + (K-1) \cdot (1 + \sum_{i=1}^{K-1} \alpha \log(L_i))$$

If we use caching, p increases, so the expected lookup time decreases. The price to pay is memory for the cache and time for cache operations.

Solution 8. Let N_A, N_B, and N_C be the number of networks that use class A, B, or C network addresses, respectively. A network with k hosts uses:

- a class A address if $2^{16} < k \leq 2^{24}$;
- a class B address if $2^{8} < k \leq 2^{16}$;
- a class C address if $1 \leq k \leq 2^{8}$.

We know that the number of networks with k hosts is:

$$N \cdot p \cdot (1-p)^{k-1}$$

Therefore, we have:

$$\begin{aligned}
N_A &= \sum_{k=2^{16}+1}^{2^{24}} N \cdot p \cdot (1-p)^{k-1} \approx N \cdot p \cdot \sum_{k=2^{16}+1}^{\infty} (1-p)^{k-1} = N \cdot (1-p)^{2^{16}} \\
N_B &= \sum_{k=2^{8}+1}^{2^{16}} N \cdot p \cdot (1-p)^{k-1} \approx N \cdot p \cdot \sum_{k=2^{8}+1}^{\infty} (1-p)^{k-1} = N \cdot (1-p)^{2^{8}} \\
N_C &= \sum_{k=1}^{2^{8}} N \cdot p \cdot (1-p)^{k-1} \approx N \cdot p \cdot \sum_{k=1}^{\infty} (1-p)^{k-1} = N \cdot (1-p)
\end{aligned}$$

Each class A network uses up 2^{24} addresses (see problem 9). Each class B network uses up 2^{16} addresses. Each class C network uses up 2^{8} addresses. Therefore, the total number of addresses used is:

$$U = N_A \cdot 2^{24} + N_B \cdot 2^{16} + N_C \cdot 2^{8}$$

Now, a network with k hosts assigns only k addresses, even if it has used up more than k addresses. Therefore, the number of addresses really assigned to hosts in class A, B, and C networks is:

$$V = \sum_{k=1}^{2^{24}} k \cdot N \cdot p \cdot (1-p)^{k-1} \approx N \cdot p \cdot \sum_{k=1}^{\infty} k \cdot (1-p)^{k-1} = N \cdot p \cdot \frac{1}{p^2} = \frac{N}{p}$$

That is, $U - V$ is the number of addresses wasted.

Knowing V and U, we can compute the efficiency of this addressing scheme. The efficiency is defined as:

$$\frac{\text{number of addresses really assigned}}{\text{number of addresses reserved (used up)}} = \frac{V}{U}$$

Notice that N will cancel out. Depending on the value of p, we find that the efficiency varies between about 2% and 12%.

Solution 9.

a) If the company uses three class B networks, then it will use up $3 \cdot 2^{16} = 196608$ IP addresses. since each class B network address comprises 2^{16} IP addresses. Note that “to use up” means to reserve so that others cannot use it, that is, to consume. So even if the company will actually assign to its 1500 hosts only 1500 of the 196608 IP addresses it has been given, it has still used up all of them.

b) If the company uses one class B network address with subnetting, it uses up only 2^{16} IP addresses.

c) If the company adopts CIDR addressing, it could get $2^3 = 8$ class C network addresses, which means it uses up $8 \cdot 2^8 = 2048$ IP addresses. The company could use a network ID of 21 bits to identify its supernetwork. The remaining 11 bits are used as follows: 2 bits identify the LAN (we actually have only 3 LANs, so one LAN ID is left available for future use) and the remaining 9 bits identify the host in the LAN ($2^9 = 512 > 500$).

Solution 10.

a) The known facts are the following:

- All hosts/routers belong to the same network (10xxxx).
- Host b and router interface r are both on subnet 1.
- Host c is on subnet 2.
- Host a is on neither subnet 1 nor 2.

From the first two facts, we know that subnet 1 must have a 3-bit mask (like interface r), so its ID is 101. Therefore host b must have the same mask (111000) and an address of 101xxx, where the x's are either 1 or 0 but xxx $\neq$ 111 (because that is host a's address).

For host c, things are a bit more complicated. Suppose the mask of subnet 2 is 3 bits long (111000). The ID of subnet 2 cannot be 100, because then hosts on subnet 2 would think that host a was on their local network, because its address begins with 100. The ID of subnet 2 cannot be 101 either, because then hosts on subnet 2 would think host b was on their local network, because its address begins with 101. Since the ID of subnet 2 must begin with 10, we have eliminated all possibilities for a 3-bit mask, so the mask must be at least 4 bits long (111100). The ID cannot be 1000, because again host a would appear to be on subnet 2, so the ID must be 1001. Therefore a possible mask for host c is 111100, in which case its address can be anything of the form 1001xx, like 100101.

b) 10xxxx

c) Since subnet 1 has a 3-bit mask, there are 3 bits left for distinguishing hosts, so there can be at most 8 nodes on subnet 1.