Solution of Tutorial Set 4

Problem 1.

There is a need for at least 950 subnetworks. Theoretically, however, we need at least 952 subnetworks to allow for the all –1s and all –0s subnetids. This means that the minimum number of bits to be allocated for subnetting should be 10 ($2^9 < 952 < 2^{10}$). Six bits are left to define the hostids. The subnet mask is 255.255.255.192.

Problem 2.

IP address: 01111000 00001110 01111010 00010000

prefix address Hostid

Mask : 11111111 11111111 11000000 00000000

So, subnetwork address: 120.14.64.0 Host id: 58.16

Problem 3.

a) In order to give each department a single subnet, the nominal subnet sizes are 2^7, 2^6, 2^5, 2^5 respectively; we obtain these by rounding up the host numbers to the nearest power of 2. A possible arrangement of subnet numbers is as follows. Subnet numbers are in binary format and represent an initial segment of bits of the last byte of the IP address; anything to the right of the / represents host bits. The / thus represents the subnet mask. Any individual bit can, by symmetry, be flipped throughout; there are thus several possible bit assignments.

A	0/	one subnet bit, with value 0; seven host bits
B	10/	
C	110/	
D	111/	

The essential requirement is that any two distinct subnet numbers remain distinct when the longer one is truncated to the length of the shorter.

(b) We have two choices: either assign multiple subnets to single departments, or abandon subnets and buy a bridge. Here is a solution giving A two subnets, of sizes 64 and 32; every other department gets a single subnet of size the next highest power of 2:

A	01/
	001/
B	10/
C	000/
D	11/

Problem 4.

a) Note: There are more than one solutions, here is one:

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 these subnetworks. Since we have 16 bits available (it's a class B address), this would leave only 16 - 13 = 3 bits for the subnetwork ID part, which could then accommodate only $2^3 = 8$ departments.
So we have to use two-layer subnetting, that is, divide some subnetworks into sub-subnetworks. At the first layer, we use 8 subnetworks of $2^{13} = 8192$ hosts each. Three of these subnetworks are completely allocated to the three large departments, 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 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 subnetwork 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.

b) If we are using class C network addresses, then for the department 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*32+7*3=124.

Problem 5.

(a) If the company uses three class B networks, then it will use 196,608 address spaces, since each class B network consumes 65,536 address spaces.
(b) If the company uses one class B network with subnetting, it uses only 65,536 address spaces.
(c) If the company adopts the CIDR addressing, by renumbering the hosts, it requires 1536 (512(=9 bit)*3) address spaces which is the number of nodes the company needs.

Problem 6.

(2) Note: IP header size is 20 bytes.

Problem 7.

a) The probability of receiving the datagram correctly is the probability of receiving all fragments correctly. The probability of receiving one fragment correctly is 0.99. The probability of receiving all fragments correctly is the product of the probabilities of receiving the individual fragments correctly. Thus:

$$P(\text{correct datagram}) = \prod_{n=1}^{10} P(\text{correct } n^{\text{th}} \text{ fragment}) = 0.99^{10} \doteq 0.9044$$

b) If we send each fragment twice, then the probability of receiving a fragment correctly is the probability that at least one of the two copies is received correctly. This probability is equal to one minus the probability that both copies are received incorrectly. Therefore, the probability of a fragment being received correctly is 1- $0.01^2 = 0.9999$. Thus, the probability of receiving the datagram correctly is:

$$P(\text{correct datagram}) = \prod_{n=1}^{10} P(\text{correct } n^{\text{th}} \text{ fragment}) = 0.9999^{10} \doteq 0.9990$$

We see that transmitting each fragment twice significantly increases the probability of correct transmission for the datagram, as expected.

Problem 8.

a) B
b) A
c) E
d) F
e) C
f) D

Problem 9.

a) The answer is shown in Figure 1. A routing table maps destinations to next hops, where a next hop simply tells the router how to forward the packet to the next router. A next hop does not include the entire path that the packet will follow. In real life, a next hop is an interface ID (like "my 3d interface"), plus the address of the next router on the path if that interface is attached to a multipoint link. The picture below shows no interface IDs or link IDs, so the problem asked for the next node.
b) To illustrate the answer each step is shown below in Figure 2.

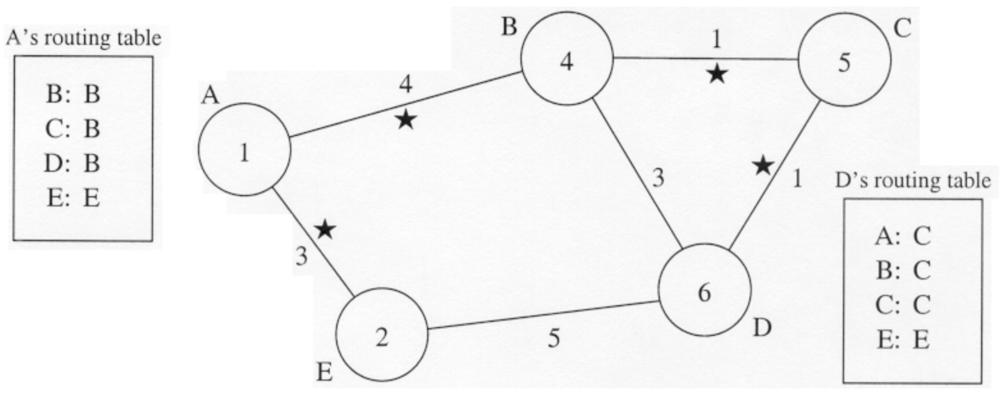

Figure 1

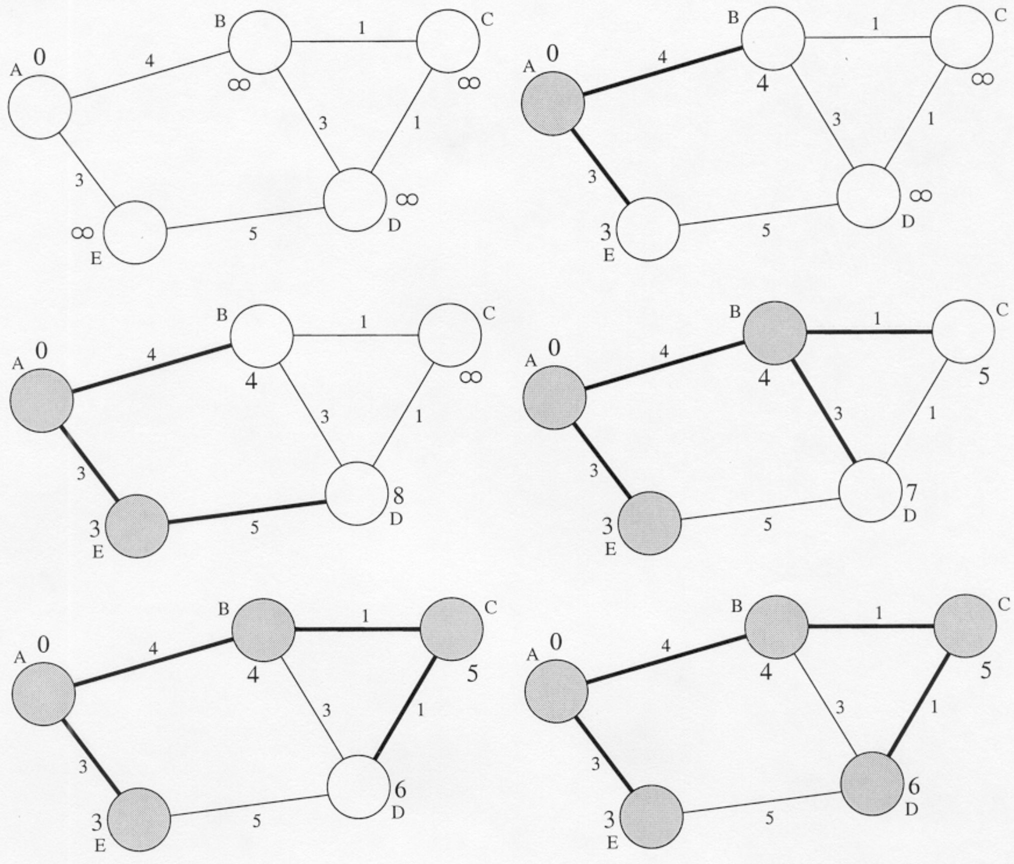

Figure 2

Problem 10.

The Bellman-Ford Algorithm: By convention, $D_1^{(h)} = 0$, for all h. Initially, $D_1^{(i)} = d_{1i}$, for all $i \neq 1$. For each successive $h \geq 1$, we compute $D_i^{(h+1)} = \min_j[D_j^{(h)} + d_{ji}]$, for all $i \neq 1$. The results are summarized in the following table.

i	D_i^1	D_i^2	D_i^3	D_i^4	D_i^5	Shortest path arcs
1	0	0	0	0	0	
2	4	4	4	4	4	(1,2)
3	5	5	5	5	5	(1,3)
4	∞	7	7	7	7	(2,4)
5	∞	14	13	12	12	(6,5)
6	∞	14	10	10	10	(4,6)
7	∞	∞	16	12	12	(6,7)

The arcs on the shortest path tree are computed after running the Bellman-Ford algorithm. For each $i \neq 1$, we include in the shortest path tree one arc (j, i) that minimizes Bellman's equation.

Dijkstra's Algorithm: Refer to the algorithm description in the text. Initially: $D_1 = 0$; $D_i = d_{1i}$ for $i \neq 1$; $P = \{1\}$. The state after each iteration is shown in the table below. P is not shown but can be inferred from i. Only the D_j's which are updated at each step are shown.

Iteration	i	D_1	D_2	D_3	D_4	D_5	D_6	D_7	Arc added
initial		0	4	5	∞	∞	∞	∞	
1	2			5	7	14	∞	∞	(1,2)
2	3				7	14	14	∞	(1,3)
3	4					13	10	∞	(2,4)
4	6					12		12	(4,6)
5	5							12	(6,5)
6	7								(6,7)

P8

a)

Prefix Match	**Link Interface**
11100000	0
11100001 00000000	1
11100001	2
otherwise	3

b) Prefix match for first address is 4th entry: link interface 3
Prefix match for second address is 2nd entry: link interface 1
Prefix match for first address is 3rd entry: link interface 2

P9

Destination Address Range	**Link Interface**
00000000 through 00111111	0
01000000 through 01111111	1
10000000 through 10111111	2
11000000 through 11111111	3

number of addresses in each range = $2^6 = 64$

P10

Destination Address Range	**Link Interface**
10000000 through (64 addresses) 10111111	0
11000000 through(32 addresses) 11011111	1
11100000 through (32 addresses) 11111111	2
00000000 through (128 addresses) 01111111	3

P11

223.1.17.0/25
223.1.17.128/26
223.1.17.192/26

P14

Any IP address in range 101.101.101.65 to 101.101.101.127

Four equal size subnets: 101.101.128.0/19, 101.101.160.0/19, 101.101.192.0/19, 101.101.224.0/19

P16

The maximum size of data field in each fragment = 480 (20 bytes IP header). Thus the number of required fragments $= \left\lceil \frac{3000-20}{480} \right\rceil = 7$

Each fragment will have Identification number 422. Each fragment except the last one will be of size 500 bytes (including IP header). The last datagram will be of size 120 bytes (including IP header). The offsets of the 7 fragments will be 0, 60, 120, 180, 240, 300, 360. Each of the first 6 fragments will have flag=1; the last fragment will have flag=0.