CSCI1680
SEUNGJI LEE
sl136

HW2

PROBLEM 1: ADDRESSING, AGGREGATION, FORWARDIG

1. IP 1

a) So if we need to assign one IP address to one computer, I would have to get class B from IPMart.
The problem with this purchase is that since we only need 560 ip addresses, we are going to waste 65536 - 560 address.
Also buying this class will be expensive.

b) There should be /22 bits in the mask. 1024 - 560 = 464 addresses will be wasted.

c) I should buy /22 and /26 blocks. (512 + 64) - 560 = 16 addresses will be wasted.

2. IP 2

a) The best CIDR block would be /19, which contains 8192 hosts.

b) It is the best to minimize the number of CIDR blocks I allocate to minimize the size of forwarding table.
The more the CIDR blocks I assign, the more entries in the forwarding table.

c) It is the best to minimize the size of the address blcoks since smaller the address blocks allows you to assign more accurate number of addresses to clients. This way, you can reduce number of addresses that are wasted.

3. IP 3

a) 128.148.34.129
b) 128.148.32.1
c) 128.148.0.1
d) 100.10.1.1
e) 128.148.0.1
f) 100.10.1.1

PROBLEM 2: IP FRAGMENTATION

1. IP FRAG 1

a) A: 4 packets
B: 10 packets
C: 10 packets

b) The fragmentation field, hdr checksum, and ttl in the IP header are different.

c) So total of 10 packets goes through the link B. Since other links do not drop packets at all,
we simply consider the probability of successful transmission in the link B. That is, (.999)^10, which is about 0.990

d) if MTU in link B is 1500 bytes, only 4 packets go through link B. Then we have the probability (.999)^4, which is about .996

PROBLEM 3: Distance Vector Routing

a) Before B sends an update, D sends D(A, 2) since D still thinks that D can reach A through B with cost of 2.
Receiving D's update, B now sets B(A, D, 3), and send the update. D receiving B's update, resets' its entry to D(A, B, 4), and so on.
This will cause count-to-infinity loop.

b) Now, assume that the designers added split-horizon to the protocol.
B broadcasts its updated entry B(A,-,inf), and C sends its table including C(A, 2) to D.
D, receiving B's update and then receiving C's update, sets the entry to D(A, C, 3), and C, receiving B's update, sets the entry to B(A,-,inf).
Now, D sends D(A, 3) to B. C sends C(A, -) to D.
B, receiving D's update, sets the entry to B(A, D, 4). D, receiving C's update, sets the entry to D(A,-,inf).
Now, B sends B(A, 4) to C. D sends D(A, -) to B.
C, receiving B's update, sets the entry to C(A, B, 5). B, receiving D's update, sets the entry to B(A,-,inf).
And so on.
This creates the count-to-infinity loop.

c) split horizon:

3. On the next updates, B will update its entry to B(A, C, 4) or B(A, D, 4). Also, B and C do not sent C(A,D,3) and D(A,C,3) to D and C, respectively, because of split horizon.
4. Now, then the entry C(A,D,3) and D(A,C,3) and B(A, C, 4) [or B(A, D, 4)] will be on the table until they timeout.

split horizon with poisoned reverse:

3. Now, C will send C(A,-) to D and D will send D(A, -) to C.
4. Then, now C and D will have C(A,-,inf) and D(A,-,inf).

d) Since path vector protocol learns all the nodes in the path to get to the destination, we can examine the path to find if the path creates a loop.

The examination should simply search if the current node is the part of the path or if there is two same node in the path.
Then, surely, the path creates a loop. This way, path vector protocol can prevent a loop from forming.