Homework 2

Due: Solution

1 IP Prefixes

a. A: 143.112.0/19, B: 143.112.32/19, C: 143.112.64/18. [2 pt]

> *A common notation nit was to have A's prefix be 143.112/19, but you should include all of the bits that are fixed by the mask.*

b. With the assumption that X has other client ASes that use up the remaining of its allocation, it advertises its entire range, 143.112/16. [1 pt]

c. When we make C multi-homed, X's advertisements do not change. Y has to start advertising C's prefix. So, X advertises, 143.112/16 and Y advertises 222.10.192/18 and 143.112.64/18. [1 pt]

> *A common mistake was to have X advertise all of Y's addresses because of the new connection, and vice versa. However, C is* ***not providing transit service to X or Y****. C is a client of both X and Y, and a client does not advertise to a provider the routes it learns from other providers.*

d. Y continues to advertise 222.10.192/18 and 143.112.64/18. X, however, cannot advertise C's address range anymore, and so has to advertise A and B, and the other half above C, i.e., 143.112.0/18 (note that this aggregates A and B), and 143.112.128/17 (the other half). [2 pt]

It is undesirable to decouple the address allocation from the topology because it prevents ASes from aggregating prefixes, as we see in this example. This causes more prefixes to be announced, and increases the sizes of the routing tables for the upstream providers.

2 IP Forwarding

a. On AB, the 30k = 30720 byte packet will be divided up into frames that carry 1480 bytes (because of the 20 byte IP header). There will be 20 such frames, carrying 29600 bytes of the 30k packets. They will all have the MF bit set, and have offsets of 0, 185, 370, etc. (Recall that the offset is defined in multiples of 8.) The final frame will have MF = 0, and contain 1120 bytes. (If you interpretted 30k = 30000, it would have 400 bytes.) [2 pt]

On link BC, each of the first 20 frames will be split into three frames containing 376 (must be a multiple of 8) bytes of IP data, and one contaning 352 bytes (offsets are 0, 47, 94, and 141 higher than the packet they came from). All of these datagrams will have the MF bit = 1. The final 1120 bytes will be sent in 3 datagrams, containing 376, 376 and 368 bytes of IP data. Only the datagram containing 368 IP bytes will have MF = 0.

Link CD will carry exactly the same datagrams as BC.

b. You should break up the 2MB transfer into datagrams that are as large as possible, without fragmenting. Taking into account the IP header (20 bytes) AND the UDP header (8 bytes), the data should be send in 372 byte pieces to avoid framentation on even the "tightest" link. This [2 pt]

allows a one-to-one correspondence between acknowledgements and datagrams. If you choose a larger frame, then they will be fragmented, and a single loss would lead to retransmission of all fragments. If you choose a smaller frame, there will be more total frames, so the per-packet overhead will be paid more often than needed. In addition, a larger percentage of each transmission will be wasted in headers.

c. FastPath is planning to use cut-through routing. In cut-through routing, a router makes a forwarding decision while the the packet is still arriving on the wire. Once the decision is made, the packet can be emitted on the proper outgoing interface even as it is still arriving on the incoming interfaces. Since forwarding is based on the destination address, moving it up in the packet allows the decision to be made faster. If IPv6 had been specified as FastPath hoped, they would be able to decrease the latency of packet forwarding by $128bits \times \frac{1sec}{100*1024*1024bits}$ = 0.00000122 secs, or 1.22 μsecs. [2 pt]

3 Distance Vector Routing

a. Here's a possible sequence. Assuming that the next hop is the sender of the update message. [3 pt]

- B sets cost to ∞
- B receives [A, 3, D], changes its parent to D
- C receives [A,6,B], updates its cost via its current parent (B)
- D receives [A,7,C], updates its cost via its current parent (C)
- ...

This loop involves the three nodes and is broken when the count of one of the nodes reaches the number specified in the protocol to represent infinity.

> *A common mistake in this question was to form a loop with only two nodes (either B/C or C/D). The question asked for a loop involving the three nodes. It is also possible to form a loop in the other direction, with B updating its parent to C, C updating its parent to D, and D to B. This loop, however, would be harder to form and easier to break.*†

b. Poison reverse (or split horizon) would not help with this loop, because it only prevents a node from selecting its children as parents, i.e., it prevents loops with two nodes. [3 pt]

A path vector protocol would prevent the loop from forming, because if a node receives any message that would form a loop it would be able to see itself in the path, ignoring the message. Such a protocol has variable-length messages and is more costly than DV, but prevents loops.

DSDV would also prevent the loop, as any message from a descendent would not have a newer sequence number, and thus would not be chosen by a parent. B would ignore any update message from C or D. The protocol has a nice feature to make this more explicit: the sources only use even sequence numbers. When a node loses its route, such as B did above, it advertises to its descendants a new sequence number that is 1 more than the last know sequence number. This prevents it from accepting any outdated information it may hear from any descendants, who would have the older sequence number, and also makes the "bad news" spread quickly down the tree.

c. A packet caught in the loop would be transmitted among the involved nodes until the TTL of the packet reached 0, when the packet would be dropped. This usually happens much faster than the speed with which routing updates happen. [1 pt]