CSE 123 Fall 2010 Homework 2

Due 10/21/2010 beginning of class.

Exercise 1

1. A reasonable timeout value for this network is 200ms - assuming negligible processing time between when a data frame is received and an ack frame is sent, this will be the expected time between when a frame is sent and its acknowledgement is received. If we assume that ACKs can only be sent 25ms apart (like data frames), then 225ms would also be appropriate, as the receiver may not be able to send its ack due to being in the middle of sending a different ACK or DATA frame.

2. Timing diagram is attached.

3. The bandwidth delay product for this link is 4 frames - thus a reasonable window size to "keep the pipe full" would be 8 - 4 frames to send for the forward delay, and 4 frames to send for the ACK's delay.

Exercise 2

Pseudocode for recovery algorithm:

1. Send the distress call on all ports.
2. Serialize processing of each response. This is not necessary for full credit in the student solution. However, serializing response processing means that we must finish one before we begin the next - this is because each iteration of the following steps both reads from and writes to the forwarding table. A race condition could occur if both reads happen before both writes, resulting in an ambiguous forwarding table.
3. For each response from host with address a on port x: add the (Destination, Port) tuple (a, x) to the forwarding table.
4. For each response from a switch on port x with text "we are communicating via port y and my forwarding table consists of tuples (d_0, p_0) through (d_n, p_n), do:
   (a) for each table entry (d_i, p_i), do:
      i. if $p_i = y$, that is, the switch is telling us that it forwards packets to d_i through the port we are talking on (through the recovering switch), ignore the message - we will be able to reach the destination either directly, or through a different switch.
      ii. for entries such that $p_i \neq y$, check whether an entry in our recovered table already exists for d_i. If an entry exists, ignore the message. If an entry does not exist, add an entry (d_i, x) - as this switch knows how to forward packets to d_i, we will allow him to do that work for us. If a second switch later also tells us it can forward to d_i, even if that was the original entry in the forwarding table, it is currently not needed.

Exercise 3

Answers for each sub question:

1. subnets:

Department name	Subnet Number	Subnet Mask
A	137.110.0.0	255.255.255.128
B	137.110.0.128	255.255.255.128
C	137.110.1.0	255.255.255.192
D	137.110.1.64	255.255.255.192
E	137.110.2.0	255.255.254.0

2. B:

Subnet Number	Subnet Mask	Next Hop
137.110.0.0	255.255.255.128	Router A
137.110.0.128	255.255.255.128	Interface 0
default		Router C

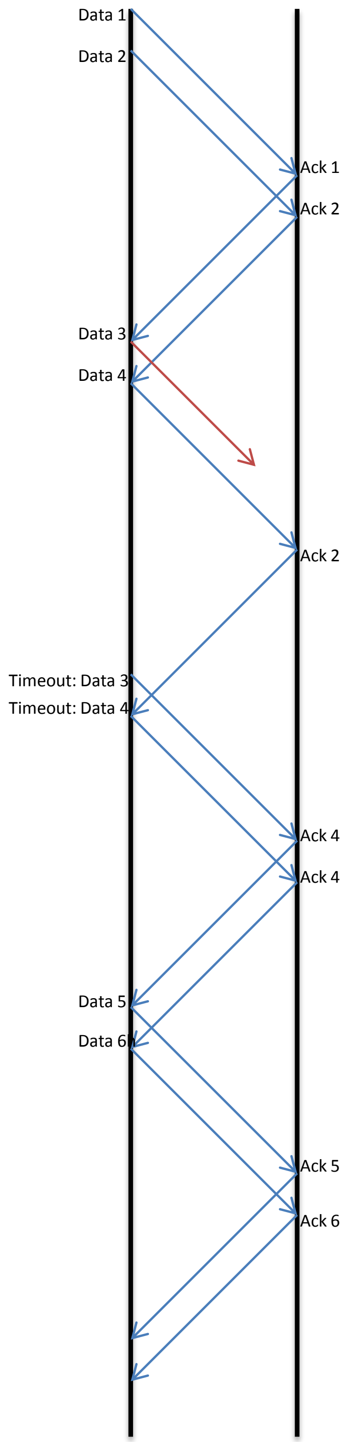
Note here that the only real entries needed are for A and B's subnets - everything else, including traffic to the rest of the internet, flows through C, and thus doesn't need a specific route.

F:

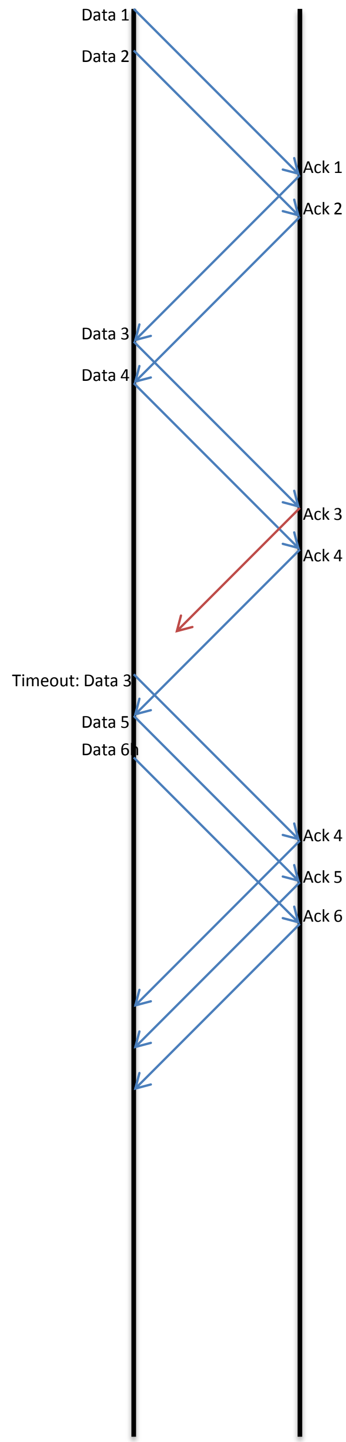
Subnet Number	Subnet Mask	Next Hop
137.110.0.0	255.255.255.0	Router C
137.110.1.0	255.255.255.192	Router C
137.110.1.64	255.255.255.192	Router D
137.110.2.0	255.255.254.0	Router D
default		Interface 0

Here, the routes for A and B are merged into the first entry - since they occupy the "bottom" and "top" halves of the 137.110.0.0/24 space and are addressed through the same router, they can be merged into one entry. This is not required for full credit.

3. Subnets A and B can grow from 100 to 126 nodes. Subnets C and D can grow from 40 to 62 nodes. Subnet E can grow from 400 to 510 nodes. As all address space has been allocated so far from low to high, all addresses from 137.110.4.0 to 137.110.255.254 are available. A new large department could be allocated at 137.110.4.0. Smaller departments could also be allocated starting at 137.110.4.0, and two small departments could also fill the two available 62 address blocks between subnets D and E.



Data 1
Data 2
Ack 1
Ack 2
Data 3
Data 4
Ack 2
Timeout: Data 3
Timeout: Data 4
Ack 4
Ack 4
Data 5
Data 6
Ack 5
Ack 6



Data 1
Data 2
Ack 1
Ack 2
Data 3
Data 4
Ack 3
Ack 4
Timeout: Data 3
Data 5
Data 6
Ack 4
Ack 5
Ack 6