Wyatt Kormick
CSCI 4211
Homework 2
11/17/17

1.

a. ACK packets tell their receiver what their sender's current window size is. One host (A) sends a packet that it knows will fit inside the buffer of the other host (B). Host B will send an ACK packet back to host A saying what the new current window size is after receiving the new packet and maybe processing some. Once again, host A will send a data packet that it knows will fit inside host B's buffer. If the buffer is full, the ACK packet will say the window size is 0. Host A will see that its full, and will send a packet of just 1 byte, for the sole purpose of eliciting a new ACK packet, with hopefully a new window size. It does this until an ACK packet says the buffer can hold more data, and continues the process. If this is happening on a router, the router that has a full buffer will scan the network for a router that has space in its buffer, and will send a different type of packet called an ICMP redirect, that tells host A to try to send to the better router for the job instead.

B. a) Rounds 1-6 and 23-26 indicate slow start operation

Slow start operation is indicated by exponential growth of the congestion window

C. b) Rounds 6-22 indicate congestion avoidance operation

Congestion avoidance is indicated by linear growth

D. c) Triple duplicate ACKs

Triple duplicate ACKs result in a congestion window size drop of only half the current window size

E. d) Timeout

Timeouts result in a congestion window size drop down to a size of 1, followed by slow start operation

2.

a.

	First Destination Host Address in Range	Last Destination Host Address in Range	Number of addresses in range
0	00000000	00111111	2^6 = 64
1	01000000	01011111	2^5 = 32
2	01100000 10000000	01111111 10111111	2^5 + 2^6 = 96
3	1100000	11111111	2^6 = 64

64+32+96+64 = 256 = 8 bits

c.

	First Destination Host Address in Range	Last Destination Host Address in Range	Number of addresses in range
0	11000000	11011111	2^5 = 32
1	10000000	10111111	2^6 = 64
2	11100000	11111111	2^5 = 32
3	00000000	01111111	2^7 = 128

32 + 64 + 32 + 128 = 256 = 8 bits

3.

a.

i. X's gives its ARP module Z's IP address. If Z is in the ARP module, it gives X Z's MAC address. If it is not, the ARP module sends a broadcast ARP packet to S. S broadcasts the ARP packet to all hosts on the subnet. The hosts that do not have the address specified in the ARP packet discard it, but the host that does (Z) sends back a standard ARP packet to S with its MAC address in it, which sends that back to X's ARP module, which puts it in the table, and gives it to X. When the switch receives the packets from X and Z, it adds their entries (the MAC address, the interface they arrived on, and the time of arrival) to the switch table.

ii. R will receive the broadcast ARP packet from X because S broadcasts it to all connected adapters. R will see that it's destination IP is not its own and will discard the packet. R will not receive the response from Z because it has the destination MAC address of X in it, and that address is in S's switch table so S will send it to X and not broadcast it

iii. X encapsulates the IP datagram in an ethernet frame and sends it out without connecting to Z first, because the frame itself has the destination MAC address. It gets sent to S. If the destination address of Z is not in the switch table, the frame gets broadcasted to the entire subnet, and everything that isn't Z (including R) discards it. If the Z is in the switch table, S forwards the frame to Z. Z receives the ethernet frame and extracts the IP datagram.

B.

i. X sends the IP address of H to its ARP modules. The ARP modules checks its cache for H's MAC address, sees that it is there, and gives it to X. X doesn't need to send out an ARP packet, so the switch doesn't do anything, and its switch table doesn't change.

ii. Since no ARP packets were sent, R doesn't receive any ARP packets from H or X, and doesn't take any action.

iii. X encapsulates the IP datagram in an ethernet frame and sends it out without connecting to H first, because the frame itself has the destination MAC address. It gets sent to S. Since H is already in the switch table, S forwards the frame to H. H receives the ethernet frame and extracts the IP datagram.

C.

i. X would not send an ARP for W's MAC address because W has a different subnet mask, and the other adapters on X's subnet would just let the ARP die.

ii. X knows that W is not on the same subnet and thus the datagram must be sent over the internet because the IP address for W doesn't have the same subnet mask. W's IP starts with 72, while X's subnet starts with 128.101.1.

iii. X encapsulates the IP datagram in an ethernet frame, with the destination IP being W's IP, the destination MAC being R's MAC on X's subnet's side (which X got from its ARP cache), the source IP being X's IP and the source MAC address being X's MAC. X sends this frame out, and S forwards it on to R because the destination MAC address is in S's switch table. R receives this frame and creates a new frame with all of the same information except it would have R's MAC address as the source MAC.

4. Virtual Circuits

a. Virtual circuits use packet switching to accomplish what appears to be circuit switching. Because it is packet switched however, it has variable latency because it has to share the network paths with other users, and because it has variable queue lengths.

b.

i. Router 1

In Port	In VCI	Out Port	Out VCI
0	1	1	1
3	1	1	2
3	2	1	3
1	4	2	4
1	5	2	5

ii. Router 2

In Port	In VCI	Out Port	Out VCI
3	1	1	1
3	2	1	2
3	3	2	3
2	1	3	4
0	1	3	5

iii. Router 3

In Port	In VCI	Out Port	Out VCI
0	4	2	1
0	5	1	2

iv. Router 4

In Port	In VCI	Out Port	Out VCI
3	1	2	1
3	2	1	2

v. Router 5

In Port	In VCI	Out Port	Out VCI
3	2	0	1

4. Link State Routing

a. X to all

step	N'	D(t), p(t)	D(u), p(u)	D(v), p(v)	D(w), p(w)	D(y), p(y)	D(z), p(z)
0	x	inf	inf	3,x	6,x	6,x	8,x
1	xv	7,v	6,v		6,x	6,x	8,x
2	xvu	7,v			6,x	6,x	8,x
3	xvut				6,x	6,x	8,x
4	xvuty				6,x		8,x
5	xvutyz						8,x
6	xvutyzw						

b.

i. t to all

step	N'	D(u), p(u)	D(v), p(v)	D(w), p(w)	D(x), p(x)	D(y), p(y)	D(z), p(z)
0	t	2, t	4, t	inf	inf	7, t	inf
1	tu		4, t	3, u	inf	7, t	inf
2	tuw		4, t		6, w	7, t	inf
3	tuwv				6, v	7, t	inf
4	tuwvx					7, t	8, x
5	tuwvxy						8, x
6	tuwvxyz						

li. u to all

step	N'	D(t), p(t)	D(v), p(v)	D(w), p(w)	D(x), p(x)	D(y), p(y)	D(z), p(z)
0	u	2, u	3, u	3, u	inf	inf	inf
1	ut		3, u	3, u	inf	7, t	inf
2	utv			3, u	3, v	7, t	inf
3	utvw				3, v	7, t	inf
4	utvwx					7, t	8, x
5	utvwxy						8, x
6	utvwxyz						

lii. v to all

step	N'	D(t), p(t)	D(u), p(u)	D(w), p(w)	D(x), p(x)	D(y), p(y)	D(z), p(z)

0	v	4, v	3, v	4, v	3, v	8, v	inf
1	vx	4, v	3, v	4,v		8,v	8,x
2	vxw	4,v	3,v			8,v	8,x
3	vxwu	4,v				8,v	8,x
4	vxwut					8,v	8,x
5	vxwuty						8,x
6	vxwutyz						

Vxwutyz. Fix this

Iv. w to all

step	N’	D(t), p(t)	D(u), p(u)	D(v), p(v)	D(x), p(x)	D(y), p(y)	D(z), p(z)
0	w	inf	3, w	4, w	6, w	inf	inf
1	wu	5, u		4, w	6, w	inf	inf
2	wut			4, w	6, w	12, t	inf
3	wutv				6, w	12, t	inf
4	wutvx					12, t	14, x
5	wutvxy						14, x
6	wutvxyz						

Wutvxyz

V. y to all

step	N’	D(t), p(t)	D(u), p(u)	D(v), p(v)	D(w), p(w)	D(x), p(x)	D(z), p(z)
0	y	7,y	inf	8,y	inf	6,y	12,y
1	yx	7,y	inf	8,y	6,x		12,y
2	yxv	7,y	12,v		6,x		12,y
3	yxvu	7,y			6,x		12,y

4	yxvut				6,x		12,y
5	yxvutw						12,y
6	yxvutwz						

VI. z to all

step	N'	D(t), p(t)	D(u), p(u)	D(v), p(v)	D(w), p(w)	D(x), p(x)	D(y), p(y)
0	z	inf	inf	inf	inf	8,z	12,z
1	zx	inf	inf	11,x	14,,x		12,z
2	zxv	15,v	14,v		14,x		12,z
3	zxvu	15,v			14,x		12,z
4	zxvut				14,x		12,z
5	zxvuty				14,x		
6	zxvutyw						

5.

a.

Initialization

Dx	X	Y	Z
x	0	3	4
Y	inf	inf	inf
Z	inf	inf	inf

Dy	X	Y	Z
x	inf	inf	inf
Y	3	0	6
Z	inf	inf	inf

Dz	X	Y	Z
x	inf	inf	inf
Y	inf	inf	inf
Z	4	6	0

Iteration 1

Dx	X	Y	Z
x	0	3	4
Y	3	0	6
Z	4	6	0

Dy	X	Y	Z
x	0	3	4
Y	3	0	6
Z	4	6	0

Dz	X	Y	Z
x	0	3	4
Y	3	0	6
Z	4	6	0

b. The count-to-infinity problem is when the distance-vector algorithm gets itself into an ‘infinite’ loop because the distance value that a node uses in its calculation is dependent on the previous distance value for the same link, so that after it is calculated, the value used to calculate it changes, so it changes again, and again, and so on.
c. Split horizon with poisonous reverse does not work when there are more than 2 nodes involved in the loop.