Homework Assignment 2

Swati Sisodia (04065889)

Chapter 3 :

1) *A sender wants to send 10 segments to a receiver. The round trip delay between the sender and the receiver is 1 second. The sender's timeout period is 2 seconds. We ignore any transmission delay. Suppose the 3rd segment and the 7th segment are each lost once. What is the total time for delivering all the segments and how many segments does the sender actually transmit/retransmit under each of the following scenarios? (12 points) a. Stop-and-Wait b. Go-back-N with window size 5 c. Selective Repeat with window size 5*

Solution:

a. 14

b. 4

c. 4

2) *Explain every field in the TCP header*.

Solution:

Following are the details for every field in TCP Header :

1. Source Port
   The source port number.
2. Destination Port
   The destination port number.
3. Sequence Number
   The sequence number of the first data octet in this segment (except when SYN is present). If SYN is present the sequence number is the initial sequence number (ISN) and the first data octet is ISN+1.
4. Acknowledgment Number:
   If the ACK control bit is set this field contains the value of the next sequence number the sender of the segment is expecting to receive. Once a connection is established this is always sent.
5. Data Offset:
   The number of 32 bit words in the TCP Header. This indicates where the data begins. The TCP header (even one including options) is an integral number of 32 bits long
6. Reserved: Reserved for future use. Must be zero.
7. Control Bits: (from left to right):

URG: Urgent Pointer field significant
ACK: Acknowledgment field significant
PSH: Push Function
RST: Reset the connection
SYN: Synchronize sequence numbers
FIN: No more data from sender

8. Window: The number of data octets beginning with the one indicated in the acknowledgment field which the sender of this segment is willing to accept
9. Checksum: The checksum field is the 16 bit one's complement of the one's complement sum of all 16 bit words in the header and text. If a segment contains an odd number of header and text octets to be checksummed, the last octet is padded on the right with zeros to from a 16 bit word for checksum purposes. The pad is not transmitted as part of the segment. While computing the checksum, the checksum field itself is replaced with zeros.

3) Explain TCP connection establishment (three-way handshaking).

Solution:

To establish a connection, each device must send a *SYN* and receive an *ACK* for it from the other device. Thus, conceptually, we need to have four control messages pass between the devices. However, it's inefficient to send a *SYN* and an *ACK* in separate messages when one could communicate both simultaneously. Thus, in the normal sequence of events in connection establishment, one of the *SYNs* and one of the *ACKs* is sent together by setting both of the relevant bits (a message sometimes called a *SYN+ACK*). This makes a total of three messages, and for this reason the connection procedure is called a *three-way handshake*.

4) Explain TCP congestion control (slow start and AIMD).

Solution:

TCP Congestion Control Window-based end-to-end flow control, where destination sends ACK for correctly received packets and source updates window size (which is proportional to allowed transmission rate).Basic idea is to determine each source determines how much capacity is available to a given flow in the network. Congestion Window (cwnd) is a variable held by the TCP source for each connection. cwnd is set based on the perceived level of congestion. cwnd is set based on the perceived level of congestion. The Host receives *implicit* (packet drop) or *explicit* (packet mark) indications of internal congestion.

The following are details for the components of the application :

a) AIMD (Adaptive Increase Multiplicative Decrease)

- Adaptive Increase stands for linear Increase i.e. for each "cwnd's worth" of packets sent, cwnd is increased by 1 packet. In practice, cwnd is incremented fractionally for each arriving ACK
- Multiplicative Decrease inTCP reacts to a timeout by halving cwnd. cwnd is not allowed below the size of a single packet.

b) Slow

Linear additive increase takes too long to ramp up a new TCP connection from cold start. the slow start mechanism was added to provide an initial exponential increase in the size of cwnd. The source starts with cwnd = 1. Every time an ACK arrives, cwnd is incremented (doubled per RTT "epoch").There are two slow start situations :

- At the very beginning of a connection
- When the connection goes dead waiting for a timeout to occur

5) Consider a reliable data transfer protocol that uses only negative acknowledgements. Suppose the sender sends data only infrequently. Would a NAK-only protocol be preferable to a protocol that uses ACKs? Why? Now suppose the sender has a lot of data to send and the end-to-end connection experiences few losses. In this second case, would a NAK-only protocol be preferable to a protocol that uses ACKs? Why?

Solution:

In a NAK only protocol, the loss of packet x is only detected by the receiver when packet x+1 is received. That is, the receiver receives x-1 and then x+1, only when x+1 is received does the receiver realizes that x was missed. If there is a long delay between the transmission of x and the transmission of x+1, then it will be a long time until x can be recovered, under a NAK only protocol. On the other hand, if data is being sent often, then recovery under a NAK-only scheme could happen quickly. Moreover, if errors are infrequent, then NAKs are only occasionally sent (when needed), and ACK are never sent – a significant reduction in feedback in the NAK-only case over the ACK-only case.

6) *Host A and B are directly connected with a 100 Mbps link. There is one TCP connection between the two hosts, and Host A is sending to Host B an enormous file over this connection. Host A can send its application data into its TCP socket at a rate as high as 120 Mbps but Host B can read out of its TPC receive socket at a maximum rate of 50 Mbps. Describe the effect of TCP flow control.*

Solution:

The receive buffer will begin to fill up, due to the fact that Host A will be sending data into it faster than Host B can remove the data from it. Once the buffer has completely filled, Host B will send a message to Host A to stop sending data until Host B can remove data from the buffer. Host B will then send a TCP segment to Host A, informing it to continue sending data. The buffer will fill up again and this process will repeat until all of the data has been sent from Host A to Host B.

Chapter 4 :

1) Suppose two packets arrive to two different input ports of a router at exactly the same time. Also suppose there are no other packets anywhere in the router. a. Suppose the two packets are to be forwarded to two different output ports. Is it possible to forward the two packets through the switch fabric at the same time when the fabric uses a shared bus? b. Suppose the two packets are to be forwarded to two different output ports. Is it possible to forward the two packets through the switch fabric at the same time when the fabric uses a crossbar? c. Suppose the two packets are to be forwarded to the same output port. Is it possible to forward the two packets through the switch fabric at the same time when the fabric uses a crossbar?

Solution:

a) No, it is impossible to forward both packets through the switch at the same time. When a shared bus topology is used only one packet is allowed through at a time.
b) It is possible to forward both packets through the switch at the same time as long as both have different destinations. That's the whole idea about crossbars.
c) No, it is not possible to forward two packets at the same time through the router to one output port. This is because although the crossbar allows multiple packets through, they cannot be at the same place at the same time

2) Consider a datagram network using 32-bit host addresses. Suppose a router has four links, numbered 0 through 3, and packets are to be forwarded to the link interfaces as follows: a. Provide a forwarding table that has five entries, uses longest prefix matching, and forwards packets to the correct link interfaces. b. Describe how your forwarding table determines the appropriate link interface for datagrams with destination addresses: 11001000 10010001 01010001 01010101 11100001 01000000 11000011 00111100 11100001 10000000 00010001 01110111

Solution:

a)

Address	Link

11100000 00 (/10)	0
11100000 01(/10)	1
11100000 01000001 (/16)	2
11100001(/8)	2
Otherwise	3

b)

Prefix match for first address is 5th entry: link interface 3
Prefix match for second address is 3nd entry: link interface 1
Prefix match for third address is 4th entry: link interface 2

3) Consider a router that interconnects three subnets: Subnet 1, Subnet 2, and Subnet 3. Suppose all of the interfaces in each of these three subnets are required to have the prefix 223.1.17/24. Also suppose that Subnet 1 is required to support at least 60 interfaces, Subnet 2 is to support at least 90 interfaces, and Subnet 3 is to support at least 12 interfaces. Provide three network addresses (of the form a.b.c.d/x) that satisfy these constraints.

Solution:

223.1.17.0/26
223.1.17.128/25
223.1.17.192/28

4) *Consider the topology shown in Figure 4.17. Denote the three subnets with hosts (starting clockwise at 12:00) as Networks A, B, and C. Denote the subnets without hosts as Networks D, E, and F. a. Assign network addresses to each of these six subnets, with the following constraints: All addresses must be allocated from 214.97.254/23; Subnet A should have enough addresses to support 250 interfaces; Subnet B should have enough addresses to support 120 interfaces; and Subnet C should have enough addresses to support 120 interfaces. Of course, subnets D, E and F should each be able to support two interfaces. For each subnet, the assignment should take the form a.b.c.d/x or a.b.c.d/x – e.f.g.h/y. b. Using your answer to part (a), provide the forwarding tables (using longest prefix matching) for each of the three routers.*

Solution:

a) Subnet A: 214.97.255/24 (256 addresses)
   Subnet B: 214.97.254.0/25 - 214.97.254.0/29 (128-8 = 120 addresses)
   Subnet C: 214.97.254.128/25 (128 addresses)
   Subnet D: 214.97.254.0/31 (2 addresses)
   Subnet E: 214.97.254.2/31 (2 addresses)
   Subnet F: 214.97.254.4/30 (4 addresses)

b) To simplify the solution, assume that no datagrams have router interfaces as ultimate destinations. Also, label D, E, F for the upper-right, bottom, and upper-left interior subnets, respectively.

5*) Consider the following network. With the indicated link costs, use Dijkstra's shortest-path algorithm to compute the shortest path from x to all network nodes. Show how the algorithm works, you may want to use a table.*

Solution:

x to y = 6
x to z = 8
x to v = 3
x to w = 6
x to u = 6 (via v)
x to t = 7 (via v)

Dijikstra's algorithm:

```
2  N' = {u} // this is x for the above diagram
3  for all nodes v
4    if v adjacent to u
5       then D(v) = c(u,v)
6    else D(v) = ∞
7
8
```

6) Argue that for the distance-vector algorithm in the following figure, each value in the distance vector D(x) is non-increasing and will eventually stabilize in a finite number of steps.

Solution:

At each step, each updating of a node's distance vectors is based on the Bellman-Ford equation, i.e., only decreasing those values in its distance vector. There is no inc r eas ing in values. If no

updating, then no message will b e sent out. Thus, D(x) is non-increasing. Since those costs are finite, then eventually distance vectors will be stabilized in finite steps