Zain Al Abidin

21L-6260

Computer Networks

Assignment-A2

93/100

Q#1

(a) 1 device can have multiple applications running at the same time. Each application is assigned a port number for effective communication. At the sender's side, multiplexing allows multiple application to share the same network connection. and at the receiver's side, demultiplexing enables the transport layer to identify which data belongs to which application. The main role of port numbers is to ensure that the right data reaches the right application. 4

(b) Port 80 and 8080 is used for HTTP web servers while port 443 is generally used for HTTPS.

In Segment 1:

→ Source IP: 10.1.1.1 → Source port: 55000. DestPort: 80

This tells us that the transport layer will deliver this segment to the HTTP server that is running on the destination

In Segment 2: 4

→ Source IP: 10.1.1.2 → Source port: 55001 DestPort: 80

Other same request but from another user. The destination IP and destination port are the same but the source IP and source port are different which means that another user and a application is trying to access the server at port 80

In Segment 3:

→ Source IP: 10.1.1.1 Source port: 55000 destPort: 8080

This is the same as Segment 1 except for the port that it is trying to access, 8080 is another alternative for HTTP servers. The same device and the same application as Segment 1 is trying to access.

In Segment 4:

The same device and same application trying to establish a HTTPS connection (secure).

Q#2

UDP is called connectionless because there is no connection established between the sender and the receiver. The sender simply adds the destination to the data packets and then they are sent, each packet might take different routes throughout the network, (some might not even reach the destination).

UDP is not reliable while TCP is. In case the data packets are corrupted or do not reach the destination, TCP ensures that it does by transmitting it again while UDP does nothing of this sort and has no error recovery. Second service that TCP provide is that it ensures that the data packets are received in the correct order while UDP does not provide this service.

An application developer might prefer UDP over TCP if he needs lower latency and simplicity at the cost of slight data loss. UDP is also far more effecient in the sense that it requires relatively less bandwidth and processing costs.

Q#3

ACK : Signal sent by receiver to sender after successful data transmission.

NAK : signal sent by receiver to sender if the data hasn't been received or has been corrupted.

When the sender receives an ACK, it knows that data has been successfully received, so it sends the next packet. If the Sender receives a NAK, then it knows that there has been some kind of issue and the receiver has received that data packet properly so it sends the same packet ~~agg~~ again. Therefore ACKs and NAKs result is reliable data transmission.

✓ 10

A protocol that only uses ACKs generally peforms better because less messages are exchanged between the sender and the receiver which results in less congestion on the network. The NAKs can easily be replaced by timeouts and duplicate ACKs.

Q#4

A timer at the sender's side is necessary so that if the packet is lost in transmission, the receiver doesn't have to ask again for re-transmission. Also if the data was transmitted successfully and the receivers sends an Ack but the ACK itself doesn't reach the sender, then the sender would be stuck waiting indefinitely

The ACK might be a little delayed and if the timer expires before ACK then the Sender would send the same packet again thinking that the packet wasn't delivered the first time.

10

Q#5

Prob of packet loss ⇒ 0.2
Prob of successful transmission ⇒ 0.8

P (Packet | send) ⇒ 0.8
P (Ack | recv) ⇒ 0.8

10

= 0.8 × 0.8 ⇒ 0.64

Therefore the probability for successful transmission and the successful receival of ACK is 64%

Q#6

In case there is some error and the sender needs to send the same packet again, then how will the receiver identify wheter it is the next packet or the previous packet transmitted again. Having a sequence number allows the receiver to rearrange them and get the message/data the way it was intended. If the receiver has sequence numbers then they can arrange them in the correct order, no matter in whatever order they arrive.

For example let's assume that the Sender sent some data and it was received successfully. The receiver sent an ACK but the ACK got lost in transmission

and the Sender didn't receive the ACK. The sender would then Send the same packet again and the receiver will receive and think that this is the new packet ~~sent~~ Since There are no Sequence numbers. Also when data is sent in multiple packets, there is a very high chance that the packets will arrive at the receiver out-of-order. If there are no sequence numbers, then there is no way for the receiver to re-assemble the original message.

Q#7

(a) If an ACK packet is lost, the sender thinks that there may have been some kind of issue and transmits the same message again. The issue that arises is that the receiver receives the same packet again while expecting ~~the~~ the next packet. It's also highly ineffecient since it uses the bandwidth when re-transmitting. The easiest way to fix this issue is to include a packet/sequence number in the header of each packet.

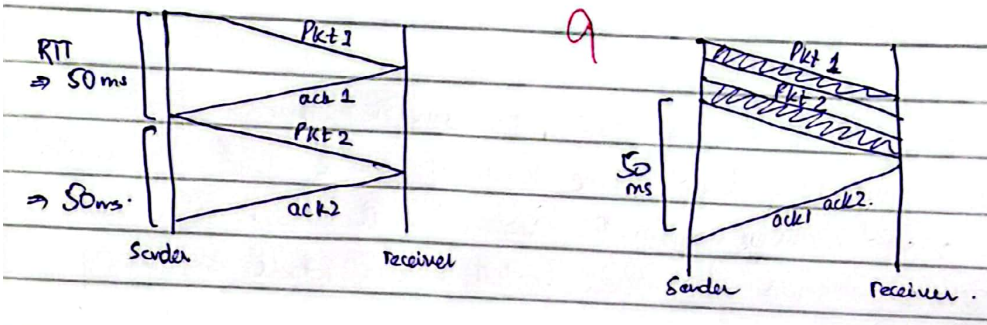
(b) The effeciency is determined by 3 factors (i) Packet size (ii) Bandwidth (iii) RTT

These 3 can be expressed as: $\frac{L/B}{L/B + RTT}$

The effeciency for a stop-and-wait protocol is generally low because of the transmission time.

Q#8

Instead of sending an ACK for each packet, TCP uses cumulative acknowledgements which is that it sends multiple packets and then after successful transmission the receiver sends multiple ACKs for multiple packets packaged in a single packet. This allows for more effecient transmission, since less bandwidth is used.



Total time ⇒ 100ms.

Total Time ≈ max 70ms.

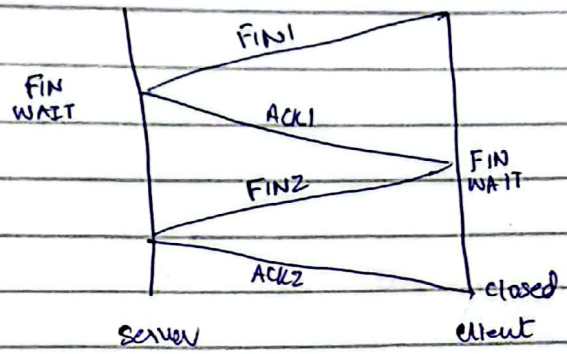
Therefor when transmission time is high, sending cumulative ACKs can help improve effeciency.

Q#9

(a) The client sends a SYN to check if the server is active. If it's active, it sends back a SYN-ACK telling the client that it's active. The client then sends an ACK back to the server to establish the connection.

It is necessary to have 3 steps because if the SYN-ACK from server gets lost in transmission then the ~~receiver~~ client might think that the server is not up. Therefore 3-way handshake is necessary so that all parties involved that they are live and want to establish a connection

(b) The graceful termination process ensures that both the client and server are done transmitting the data and want to terminate the connection



9

Gets acknowledgement from both sides by ACK that they're done transmitting data and want to terminate.

Q#10

(a)

$$\text{throughput} \Rightarrow \frac{\text{window size}}{RTT} \Rightarrow \frac{10 \times 1500}{0.1} \Rightarrow 150,000$$

$\Rightarrow$ 150 kb/sec

(b) If a packet loss occurs when fast retransmit and fast recovery are used then the Congestion window is Reduced which affects the throughput of the next RTT. When the receiver finds out that a packet has been lost, it sends the lost packet again without waiting for timeout.

(c) If RTT increases while everything else remains same the the throughput would be affected in a negative way.

1