CSC358H5: Principles of Computer Networking — Winter 2025

Worksheet 10 Solution: HTTP, Transport Layer, UDP, TCP

Q0 Knowledge Check

0.a List one advantage and one disadvantage of using a text-based header (as in HTTP) instead of a binary format. Why does IP use a binary format?

Answer. A text-based header is easier for human beings to read and debug. Text-based headers are also extensible. However, they are verbose (*i.e.*, waste bandwidth) and harder to parse. IP uses a binary format to limit the bandwidth consumed by the header, and to simplify parsing at the router, which much process packets very quickly (*e.g.*, within a few nanoseconds on high-speed links).

0.b Which of the following statements are True? Choose all that apply.

- ☐ HTTP is a stateful protocol that remembers user sessions by default.
- ☐ HTTP operates at the transport layer of the OSI model.
- [✓] HTTP uses request and response messages to facilitate communication between clients and servers.
- ☐ HTTP always ensures data encryption without the need for additional protocols.

0.c In your Wireshark labs, you worked with two application layer protocols that operate over UDP as their transport layer protocols. Name those application layer protocols.

Answer. DHCP and DNS.

0.d We had seen many protocols and devices in this course that use Soft State. In the following part, specify why using the Soft State is necessary.

0.d. i **DHCP Offer Lease Time:** In your Wireshark lab, you notices that an offer message from a DHCP server includes a lease time, along with other fields like an IP address, subnet mask, DNS server address, and so on. Why is a lease time necessary?

Answer. The lease time allows the DHCP server to reclaim the IP address, even if the client never explicitly releases the address (*e.g.*, if the client crashes or has a buggy DHCP implementation).

0.d. ii **DNS Response Time-to-Live:** A DNS response message from a DNS server includes a time-to-live field. Why is this necessary?

Answer. The time-to-live field determines how long the client can safely cache the response. Otherwise, a client might cache a name-to-address (or address-to-name) mapping indefinitely. Yet, the mapping may need to change over time. For example, a Web site may move from one hosting provider to another, forcing a change of IP address. Even if a site does not change providers, DNS may be used to control how Web clients are directed to different replicas (*e.g.*, for load balancing), so the ability to adjust the mapping over time is important, requiring a mechanism for flushing the DNS cache. Using a TTL places the responsibility for flushing the cache at the client, rather than requiring the server to remember (and contact) all past clients to notify them when the mapping information has changed.

0.d. iii **IP TTL Field:** The IP packet header includes a time-to-live field that is decremented by each router along the path. Why is the time-to-live field necessary?

Answer. A packet may get stuck in a forwarding loop (*e.g.*, due to a router configuration mistake). By decrementing the TTL field at each hop, and discarding the packet when the TTL reaches 0, the network prevents the packet from cycling in a loop indefinitely. Otherwise, the packet would consume excessive resources, or even escape the loop eventually and reach the destination much later (running the risk that the packet is mistakenly viewed as part of a more recent transmission with the same IP addresses and TCP/UDP port numbers).

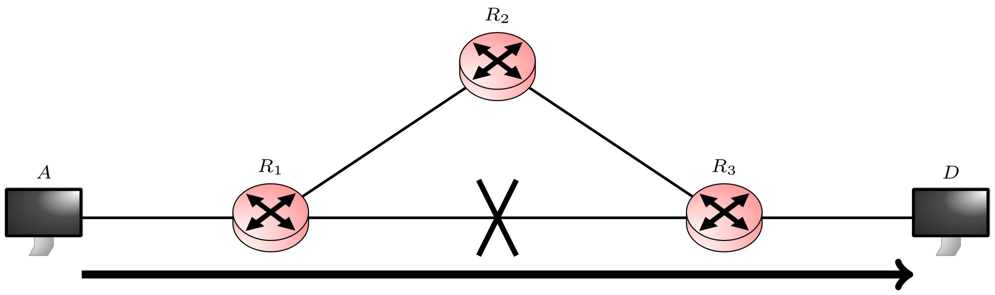
0.e Why does a TCP sender use a very large retransmission timeout (*e.g.*, several seconds) to detect and retransmit a lost SYN packet?

Answer. The TCP sender does not have any initial estimate of the round-trip time (RTT). Starting with a conservative retransmission timeout (RTO) prevents the excessive retransmissions that would result from using an RTO that is smaller than the actual RTT.

0.f Considering the TCP implementation that was described in lecture, when does fast retransmit fail to improve the performance?

Answer. Fast Retransmissoin fails to improve the performance if triple duplicate ACKs are less likely to "kick in" and timeout occurs instead. This will happen when TCP has a very small window size, *i.e.*, there are not enough ACKs to trigger the Fast Retransmission. Furthermore, when we have "bursty packet loss," multiple packets are lost together and there won't be enough ACks to trigger the Fast Retarnsmission.

0.g Suppose two hosts have a long-lived TCP session over a path with a 100 msec round-trip time (RTT). Then, a link fails, causing the traffic to flow over a longer path with a 500 msec RTT. Suppose the router on the left recognizes the failure immediately and starts forwarding data packets over the new path, without losing any packets. (Assume also that the router on the right recognizes the failure immediately and starts directing ACKs over the new path, without losing any ACK packets.) Why might the TCP sender retransmit some of the data packets anyway?



Answer. TCP bases its retransmission timeout (RTO) on an estimate of the round-trip time between the sending and receiving hosts. In this example, the RTT is 100 msec before the failure. As this connection has been active for some time, the sender's RTT estimate should pretty accurate. The RTO is typically (say) twice the RTT estimate. When the failure occurs, the increase in the actual round-trip time implies that the ACK packets will not arrive before the RTO expires. This causes the sender to presume the data packets have been lost, leading to retransmissions, despite the fact that no packets were actually lost.

0.h When starting a new TCP connection, why do the sender and receiver each pick a random initial sequence number (ISN)? Why not start every TCP transfer with a sequence number of 0?

Answer. The port numbers in TCP connections come from a finite range and, as such, are reused over time. As such, it is possible that two communicating hosts are using a pair of port numbers that were used in the past. It is conceivable that a packet from the earlier connection is still in flight and might reach the receiver. To reduce the likelihood that the old packet is viewed as part of the ongoing transfer, the starting sequence number changes over time.

0.i In the three-way handshake to open a TCP connection, host A sends a SYN, host B sends a SYN-ACK, and host A sends an ACK. When can the TCP implementation at host A start sending data packets? When can the TCP implementation at host B start sending data packets? Explain why they cannot start sending any sooner.

Answer. Host A can start sending upon receiving the SYN-ACK, since that signals to host A that B is willing to accept traffic and knows A's initial sequence number. Host B can start sending traffic upon receiving the ACK, since that signals to host B that A is willing to accept traffic and knows B's initial sequence number.

0.j What happens, at the socket level, when the user clicks "reload" on a Web browser when the data transfer is proceeding slowly? Why does this often lead to a faster download?

Answer. Clicking "reload" aborts the existing socket (and the associated underlying TCP connection), and triggers the browser to initiate a new socket (and the OS to initiate a new underlying TCP connection). If the TCP connection is temporarily stalled due a packet loss (such as a lost SYN packet), starting a new connection effectively leads to an immediate "retransmission" of a SYN packet (albeit for a new socket, rather than the old one).

Q1 An HTTP response message can include a "Last Modified" time, indicating the last time the requested object was modified. When a user requests an object that resides in the browser cache, the browser generates an HTTP request message with an "If Modified Since" header to ask the server to return a fresh copy only if the object has been modified since the previous request. How is it possible for the "If Modified Since" cache-validation technique to work, even if the two machines have vastly different notions of time *(i.e.*, no clock synchronization)?

Answer. Both timestamps are from the viewpoint of the server, making the client's notion of time irrelevant. That is, the server transmits the "Last Modified" time, and the client returns this timestamp as the argument of "If-Modified-Since" in a subsequent request.

Q2 To make an HTTP request, the client establishes a TCP connection and sends an HTTP request message. An HTTP request message has a "Host" header than includes the name of the server (*e.g.*, "Host: www.erfanmeskar.com"). Why is the Host necessary? Wouldn't the server already know its own name?

Answer. A single server might host multiple Web sites, making the Host header necessary to distinguish between the sites.[1]

Q3 Your boss asked you to customize TCP to make it suitable for the specific network of your organization. You were told that the maximum possible rount-trip-time (RTT) in this network is 100 ms. Moreover, the links in this network have a capacity of 1 Gbps. Furthermore, you were told that the **Maximum Segment Lifetime (MSL)** in this network is 30 seconds. MSL defines the maximum time a TCP segment can remain valid in the network. It represents the longest time a TCP segment could be in transit before being discarded.[2]

3.a What is the minimum number of bits that you need for Sequence Number and Advertised Window field in the header of the TCP that you customized for this network?

Answer. To be able to provide the maximum possible throughput, the receiver's window size must be sufficiently large to allow the sender to keep the data pipeline fully utilized.

Thus, the receiver's window size must be at least $(d^{\text{tran}} + \text{RTT}) \times \text{BW}$. As in most realistic networks, it is safe to assume that d^{tran} is significantly smaller than RTT. Thus, the minimum receiver's window size can be approximated as $\text{RTT} \times \text{BW} = 100$ Mbit, which is equivalent to 12.5 Mbyte. We should have enough number of bits in the Advertised Window field to be able to represent 12.5 Mbyte. Thus, we need to have at least 24 bits in the Advertised Window field.[3]

MSL of 30 s indicates that in this network, it is possible for packets to be deliver with 30 s delay. Hence, the range for the sequence numbers should be large enough to not wrap during MSL. Otherwise, there's a potential for delayed packets to be mistaken as new ones, leading to issues in reliable data transmission. During a MSL, the sender can send up to $\text{MSL} \times \text{BW} = 30$ Gb, which is equivalent to 3.75 Gbyte. Thus, we need to have at least 32 bits in the Sequence Number field.[4]

3.b Among RTT and MSL, which one is more difficult to estimate?

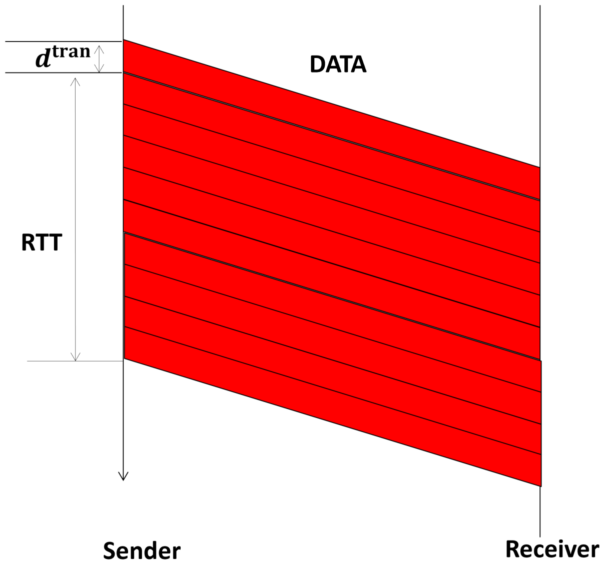
Answer. RTT is relatively accurate to measure at the moment, but can change in the future. MSL is more dificult to estimate as it depends on the size complexity of the network and routing loops.

[1]The Host header is also important when the client connects to a server through a proxy, so the proxy knows which site to contact. [**NOTE:** This footnote won't be covered in your exam.]

[2]In addition to detirmining the number of bits you need for the Sequence Number filed, MSL is used in TIME-WAIT mechanism of TCP, too. When a TCP connection is closed, the endpoint that performs an active close (sending the final FIN packet) enters the TIME-WAIT state for a duration of 2 $\times$ MSL. This ensures that all packets related to the closed connection are either delivered or discarded by the network. [**NOTE:** This footnote won't be covered in your exam]

[3]with 2^{24} bits in the Advertised Window field, we can represent any receiver's window of size 0 to about 16 million bytes.

[4]with 2^{32} bits in the Sequence Number field, we can represent any sequence number between 0 to about 4 billion bytes.



Q4 Suppose two hosts are about to open a TCP connection. The TCP headers used in the communication are only 20 bytes long and regular (*i.e.*, no options) IPv4 is being used.

4.a If the MTU of the link is 1260 bytes, what is the MSS?

Answer. MSS = MTU − TCP header size − IP header size $= 1260 - 20 - 20 = 1220$ bytes.

4.b When this connection starts, the sender starts with an ISN 19. The initial window for the sender is set to 10 MSS.

4.b. i Given the previously calculated MSS, what ACK number does the sender receive **as part of the TCP handshake**?

Answer. The ACK number received during the TCP handshake will be 20 as the receiver is ACKing having received the TCP SYN packet and is confirming that it is waiting for first data carrying byte which has a sequence number of ISN $+ 1 = 20$.

4.b. ii After sending the first window of data, what is the first and last ACK number the sender receives for this initial window? (Assume no packets were lost or reordered).

Answer. The first ACK number corresponds to the first sent segment. This segment starts with sequence number 20 and its last byte is numbered as 1239 as the segment has a size of MSS $= 1220$ bytes. Therefore, the first ACK number would be $1239 + 1 = 1240$, which refers to the next byte that the receiver is waiting for.
Similarly, the last ACK number corresponds to the last sent segment. The last byte of this segment is numbered as ISN $+ 10 \times MSS$. Therefore, the last ACK number would be ISN $+ 10 \times MSS + 1 = 12220$, which refers to the next byte that the receiver is waiting for.

Q5 Consider a Web server that generates and sends HTTP response messages to clients over sockets. The header of an HTTP response message consists of a collection of lines, each ending with a carriage return and line feed. For example,

```
HTTP/1.1 200 OK
Server: Apache/1.2.7-dev
Date: Tue, 07 Jul 1998 18:21:41 GMT
Content-Type: text/html
...
```

Some early Web-server software generated the lines one at a time, and used a separate system call to write (or send) each line to the socket.

5.a Why is this approach inefficient for the end host?

Answer. The server makes a separate system call for each line, leading to a high overhead for switching between the user-space process and the operating system. The smaller packets arriving at the client may also require the browser to perform many system calls to receive the full header, if the packets do not arrive close together in time.

5.b Why is this approach inefficient for the network?

Answer. The resulting TCP segments are very small, in the same ballpark as the link-layer, IP, and TCP headers themselves. A large fraction of the network bandwidth is consumed by the headers. Also, the receiver must transmit ACK packets in response to receiving the data packets, leading to a large number of ACK packets.

5.c Describe how a programmer implementing the Web server software could fix this problem.

Answer. The programmer could create a large user-space buffer (*e.g.*, at least 1460 bytes long, if not longer) and create the response header there. Then, the programmer could make a single socket call to send the full header. The operating system will then generate large packets containing multiple lines of data and send these larger packets into the network.

Q6

This problem is inspired by a question raised by one of the students in class during last week's lecture.

Many of the protocols underlying the Internet rely on end hosts to faithfully implement the protocols correctly, for the greater good. This question explores what happens when they don't.

6.a Consider a multi-access Ethernet where one host has an adapter that does not back off after detecting a collision. Describe how this would affect the communication performance for the other hosts.

Answer. When a collision occurs, the other (well-behaved) adapters would back off, while the rogue adapter would continue sending; the well-behaved adapters would not try to send again because the link is already busy. Ultimately, the rogue adapter would have nearly complete control of the link, leaving the other adapters will little or no bandwidth.

6.b Consider a TCP implementation that does not reduce its sending rate in response to packet loss. Describe how this would affect the other TCP traffic sharing the congested link. Also, explain whether/how this non-compliant behavior might hurt the performance of the offending TCP connection.

Answer. By not reducing the sending rate in response to packet loss, the rogue TCP sender would not do its part to reduce congestion, forcing other packets to be dropped. Eventually, well-behaved TCP senders would reduce their sending rates and share whatever bandwidth the rogue sender is not using. This would lead to unfairness. It may also hurt the rogue TCP sender because driving the link beyond its capacity may cause loss of the rogue sender's packets, requiring retransmisisons.