פרק 6א'

1. האם ניתן לאחד בין שכבת האפלקציה ושכבת התעבורה לשכבה יחידה?

2. מדוע אנו נדרשים לסקוטים פנימיים שיקשרו בין שכבת האפלקיציה לשכבת התעבורה

פרק 6 ב'

1. מהו ההבדל בין TCP לUDP?

3. מדוע משחק רב משתתפים משתמש בפרוטוקול TCP ולא בפרוטוקול UDP?

4. מי מחליט מתי להשתמש בפרוטוקול TCP ומתי להשתמש בפרוטוקול UDP?

5. האם פרוטוקול HTTP משתמש ב TCPאו בUDP?

6. האם ניתן להשתמש ב פרוטוקול UDPהכמוס בתוך פרוטוקול TCP?

7. מנקודת מבטו של מפתח, מהו ההבדל בין פרוטוקול TCP ל UDP, מהם הAPI-ים שנצטרך לשנות כשאנחנו באים לשנות את התכנות מתכנות TCP לUDP?

8. באם לקוח מדבר אך ורק בפרוטוקול TCP , והשרת מדבר אך ורק בפרוטוקול UDP, האם יתאפשר חיבור ביניהם (בין הלקוח לשרת)?

9. מדוע פרוטוקול TCPפופולרי יותר מפרוטוקול UDP?

10 (שאלת ידע כללי) מדוע המערכת פרטוקלים קרויה "TCP/IP " בעוד UDP נקרא רק "UDP"

11. מהם היתרונות בשימוש בפרוטוקולי TCP וUDP יחד?

12. מדוע אפליקציות שונות משתמשות בפרוטוקולי TCP וUDP שונים בכדי לתקשר ברשת (האינטרנט)?

13. כמה לקוחות יכולים להתחבר לפורט TCPיחיד?

14. כיצד זה אפשרי לשרת יותר מ65,535 חיבורים סימולטניים כאשר פורט TCP , הוא ממספר 16 ביט.

Chapter 6A

1. Is it possible to merge the transport layer and the application layer of the tcp ip model into one layer?
2. Why do we need a socket interface between application layer and transport layer?

Chapter 6B

1. What is the difference between TCP and UDP?

2. When is it better to prefer TCP instead of UDP connections?

3. Why do multiplayer games use UDP and not TCP?

4. Who decides whether to use TCP or UDP?

5. Does HTTP use TCP or UDP? Why?

6. Can we use the UDP protocol encapsulated in a TCP connection?

7. Purely from a programming point of view, what is the difference between TCP and UDP socket programming? What are the APIs which need changes between TCP and UDP?

8. If a client can speak only TCP and a server can speak only UDP, will communication between them be possible or not?

9. Why is TCP more popular than UDP?

10. Why is the Internet protocol suite called "TCP/IP" if UDP is also common?

11. What's the benefit of using TCP and UDP protocols together?

12. Why do different applications use different TCP or UDP ports to communicate on the Internet?

13. How many clients can connect to a single TCP port?

14. How is it possible to serve more than 65535 connections simultaneously, when the the TCP port number is a 16-bit number?

Chapter 6 A

1. Is it possible to merge the transport layer and the application layer of the tcp ip model into one layer?

Your question is open to interpretation. You may want to include more details about what you are looking to do. The layers in the Open System Interconnect (OSI) model are reference layers. There isn't an exact mapping between the layers in the OSI model and the TCP/IP stack. For example, there is no session layer in TCP/IP.

The application layer can bypass TCP (transport layer) and IP (network layer) using a user-land TCP/IP stack such as OpenOnload. Keep in mind the kernel is optimized for efficiently handling TCP segments. Anyone who moves this processing into user-land likely has very specific latency needs that drive this. I'm not sure if I'd call this merging layers. You've moved the processing from the kernel to the application layer.

2. Why do we need a socket interface between application layer and transport layer?

There is always an interface between layers. Within the kernel it can be as simple as a function call, but that still constitutes an interface.

To prove this point, what would you replace the socket interface with?

Yes, you can do simpler and more elegant interfaces. Sockets are actually quite klunky, but they are what folks came up with.

Chapter 6 B

1. What is the difference between TCP and UDP?

Transmission Control Protocol is a connection-oriented protocol, which means that it requires handshaking to set up end-to-end communications. Once a connection is set up, user data may be sent bi-directionally over the connection.

- Reliable – TCP manages message acknowledgment, retransmission and timeout. Multiple attempts to deliver the message are made. If it gets lost along the way, the server will re-request the lost part. In TCP, there's either no missing data, or, in case of multiple timeouts, the connection is dropped.
- Ordered – If two messages are sent over a connection in sequence, the first message will reach the receiving application first. When data segments arrive in

the wrong order, TCP buffers delay the out-of-order data until all data can be properly re-ordered and delivered to the application.

- Heavyweight – TCP requires three packets to set up a socket connection, before any user data can be sent. TCP handles reliability and congestion control.
- Streaming – Data is read as a byte stream, no distinguishing indications are transmitted to signal message (segment) boundaries.
User Datagram Protocol is a simpler message-based connectionless protocol. Connectionless protocols do not set up a dedicated end-to-end connection. Communication is achieved by transmitting information in one direction from source to destination without verifying the readiness or state of the receiver.

- Unreliable – When a UDP message is sent, it cannot be known if it will reach its destination; it could get lost along the way. There is no concept of acknowledgment, retransmission, or timeout.
- Not ordered – If two messages are sent to the same recipient, the order in which they arrive cannot be predicted.
- Lightweight – There is no ordering of messages, no tracking connections, etc. It is a small transport layer designed on top of IP.
- Datagrams – Packets are sent individually and are checked for integrity only if they arrive. Packets have definite boundaries which are honored upon receipt, meaning a read operation at the receiver socket will yield an entire message as it was originally sent.
- No congestion control – UDP itself does not avoid congestion, unless they implement congestion control measures at the application level.
- Broadcasts - being connectionless, UDP can broadcast - sent packets can be addressed to be receivable by all devices on the subnet.

2. When is it better to prefer TCP instead of UDP connections?

TCP and UDP both are transport layer protocols.Main difference between two is TCP is connection oriented and UDP is connectionless.

In a connection-oriented service, a connection is first established between the sender and the receiver. Data are transferred. At the end, the connection is released.TCP creates a virtual connection between two TCPs to send data. In addition, TCP uses flow and error control mechanisms at the transport level.

In connection-less service,the packets are sent from one party to another with no need for connection establishment or connection release. The packets are not numbered; they may be delayed or lost or may arrive out of sequence. There is no acknowledgment either.

So when we don't have to bother about error and flow control , we use UDP such as in multicasting ,management processes such as SNMP.

But when we require to have a reliable transmission we use TCP. TCP is used when client connect to servers on the World Wide Web and it is used to deliver email and transfer files from one location to another. HTTP,FTP,Telnet,SMTP,POP3,IMAP also use TCP.

3. Why do multiplayer games use UDP and not TCP?

Real time games generally use UDP because retransmissions aren't particularly useful. If a packet is delayed or lost, it's simply not useful.

4. Who decides whether to use TCP or UDP?

HTTP always uses TCP - it was a design decision by Berners-Lee. Most protocols that require reliability use TCP since it is handshaked and delivery is guaranteed by re-sending missing packets. Web, email, SSH, IRC all use TCP.

UDP is used when timing is more important than delivery.If a packet of audio data is lost during an internet phone call, there's no point re-sending it, it's too late, the sound has already been played. So RTP uses UDP.
UDP was used in more innocent times for protocols such as DNS and NTP, but this is becoming a liability because of the ease of spoofing the source address in UDP packets, and consequent use of UDP in traffic amplification DDOS attacks.

The upcoming WebRTC protocol implemented directly in browsers uses RTP and hence, I presume, UDP.

5. Does HTTP use TCP or UDP? Why?

HTTP uses TCP because the files, images, web pages which we get from the remote host should not be dropped on the way and it should be delivered in order to the HTTP client.

HTTP could also use UDP but usually not, if a UDP packet containing the first part of a web page is lost, then its not retransmitted. We need to write some code in the application layer to handle these retransmissions of lost packets when we use UDP.

Since, we don't want to burden the web browsers (Application Layer) in doing this work rather than doing the actual work of rendering pages, we use TCP.

6. Can we use the UDP protocol encapsulated in a TCP connection?

Of course. And it would be totally pointless.

UDP is a lightweight datagram protocol.

TCP is a heavier weight connection stream protocol that guarantees order of delivery and provides automatic retransmission.

You can send anything you like in a TCP stream, including UDP packets. But adding the TCP overhead to a stream of UDP packets completely negates the advantages of UDP.

7. Purely from a programming point of view, what is the difference between TCP and UDP socket programming? What are the APIs which need changes between TCP and UDP?

The only two differences I could think of are :
1. In UDP, for creating a socket, you use socket(AF_INET, SOCK_DGRAM, 0

whereas in TCP, you would use sockfd = socket(PF_INET, SOCK_STREAM, 0);
2. In TCP, you use the APIs send() and recv() for sending and receiving messages between the server and client, but since UDP doesn't establish a connection, you need to give the source and destination, and so you would use sendto() and recvfrom()

In TCP you have to use bind(), connect(), listen(), accept(). You have a lot of things to set/reset in socket options.

8. If a client can speak only TCP and a server can speak only UDP, will communication between them be possible or not?

If client can understand UDP and server can understand TCP (i.e keep the ports open, does not send out data is fine but atleast send acks in case of TCP) then yes otherwise no

9. Why is TCP more popular than UDP?

TCP is more reliable compared to UDP as it is an acknowledgement based protocol.

After each packet sent, the sender/originator of that packet expects the receiver to send back an acknowledgement in a fixed time interval. If receiver didn't receive the packet or failed to send an acknowledgement back to the designated originator within the timeout period, Originator re-transmits the concerned packet again. A buffer is used to stored copy of packets in the originator in case acknowledgement is not received and re-transmission is required.

TCP also takes care of network traffic by implementing protocols like flow control(sliding window protocol) and congestion control.

Hence, basically UDP is used when one doesn't care if all data is transmitted like media streaming(loss of some frames can be acknowledged),gaming(doesn't care if user input is missed some times) etc.
Whereas TCP is used for more serious concerns where loss of data can erupt the entire application performance. Eg. Web, SMTP(sending e-mails), IMAP/POP(receiving e-mails),etc.

10. Why is the Internet protocol suite called "TCP/IP" if UDP is also common?

TCP and IP were standardized together in 1974 in RFC 675, Specification of Internet Transmission Control Program.

TCP provided connection services, and IP was the connectionless datagram protocol carrying the TCP traffic, providing addressing and fragmentation. They were standardized together, but in 1980 RFC 760 specified IP separately.

UDP was subsequently standardized later in 1980 in RFC 768 - User Datagram Protocol, to allow more capabilities in a datagram protocol than IP by itself afforded (e.g., port numbers, packet length, checksum).

TL;DR: TCP was there first.

11. What's the benefit of using TCP and UDP protocols together?

UDP and TCP are two protocols for network communication.

TCP provides a reliable byte stream, meaning that any packet loss is handled automatically by the protocol. The procotol guarentees that each byte received is exactly the same as each byte sent.

On the other hand UDP does not have this guarantee. So UDP messages may get lost in transit and never show up at the receiver.

In a game (e.g. chess), the game moves should be sent using TCP.

On the other hand, status messages that are sent regularly could be sent using UDP. For instance, the current value of the clocks, assuming that this information is sent at a regular interval.

The real benefit is to make the program simpler. This is mainly due to assigning different ports to different types of information. So instead of a single big receive function having to decode various message types (a move, a clock update, etc.), you have several smaller functions dealing with only a single message type.

12. Why do different applications use different TCP or UDP ports to communicate on the Internet?

Port numbers are basically how transport and application layers interact.

As you are aware that multiple applications might be running on your PC (IM, torrent, web browser etc) and a lot of them are sending data packets at same time, it's not like all other applications will stop when you are operating browser, that'd suck frankly. So, your data from multiple applications is multiplexed at lower layers.

Now, when you get a reply back, there must be way to send the received data received to correct applications. You wouldn't want the packets meant for you IM being delivered to you browser or to torrent software or any other program. This is what ports do, they tell the transport layer which application a particular packet is meant for! Thus avoiding chaos.

13. How many clients can connect to a single TCP port?

It's only limited by the TCP implementation.

14. How is it possible to serve more than 65535 connections simultaneously, when the the TCP port number is a 16-bit number?

That's 65k alplication ports per IP address. So you simply bind your listener to different IP addresses - or do the same for other uses that require 2+ multiples of 65 sockets.

What is UDP?

Re-Ask