Topics:
1. HTTP Protocol
2. Wireshark Software
3. SMTP
4. DNS
5. FTP

HTTP PROTOCOL:
1. HTTP stands for HyperText Transfer Protocol. The protocol is used to transfer hypertext between two computers.
2. HTTP provides standard between a web browser and web server to establish communication. It is set of rules for transferring data from one computer to another. Data such as text, images, and other multimedia files are shared on the World Wide Web.
3. Whenever a web user opens their web browser, user will indirectly uses HTTP.
4. HTTP is application layer protocol which is used to deliver data from server to client or vice-versa.
5. Clients and servers communicate by exchanging individual messages (as opposed to a stream of data). The messages sent by the client, usually a Web browser, are called requests and the messages sent by the server as an answer are called responses.
6. HTTP is a generic and stateless protocol which can be used for other purposes as well using extensions of its request methods, error codes, and headers.
7. Basically, HTTP is a TCP/IP based communication protocol, that is used to deliver data (HTML files, image files, query results, etc.) on the World Wide Web. The default port is TCP 80, but other ports can be used as well.
8. The HTTP protocol is a request/response protocol based on the client/server based architecture where web browsers, robots and search engines, etc. act like HTTP clients, and the Web server acts as a server.
9. The HTTP client sends a request to the server in the form of a request method, URI, and protocol version, followed by a MIME-like message containing request modifiers, client information, and possible body content over a TCP/IP connection
9. The HTTP server responds with a status line, including the message's protocol version and a success or error code, followed by a MIME-like message containing server information, entity meta information, and possible entity-body content.

WIRESHARK:
1. Wireshark is a network protocol analyzer, or an application that captures packets from a network connection, such as from your computer to your home office or the internet.
2. It is used to track the packets so that each one is filtered to meet our specific needs. It is commonly called as a sniffer, network protocol analyzer, and network analyzer. It is also used by network security engineers to examine security problems.
3. Wireshark is a free to use application which is used to apprehend the data back and forth. It is often called as a free packet sniffer computer application. It puts the network card into an unselective mode, i.e., to accept all the packets which it receives.

SMTP:
1. SMTP (Simple Mail Transfer Protocol) is a TCP/IP protocol used in sending and receiving email. SMTP is used most commonly by email clients, including Gmail, Outlook, Apple Mail and Yahoo Mail. The other two are Post Office Protocol v3 (POP3) and Internet
Message Access Protocol (IMAP). SMTP is one of several internet protocols that are designed to be plain text and ASCII printable. This means that traffic sent over SMTP is visible and easily readable by eavesdroppers. When running in plain text mode, SMTP uses port 25. Port 587 is the official port that should be used by SMTP clients submitting traffic to be routed by a mail server.
2. SMTP can send and receive email, but email clients typically use a program with SMTP for sending email. Because SMTP is limited in its ability to queue messages at the receiving end, it's usually used with either Post Office Protocol 3 (POP3) or Internet Message Access Protocol (IMAP), which lets the user save messages in a server mailbox and download them periodically from a server. SMTP is typically limited to and relied on to send messages from a sender to a recipient.
3. An SMTP server is an application or computer that sends, receives and relays email. These servers typically use TCP on port 25 or 587.
4. SMTP servers are set to an always-on listening mode and as soon as a server detects a TCP connection from a client, the SMTP process initiates a connection to port 25 to send the email.
5. Outgoing SMTP servers send messages for users. Email clients which are used to read and send emails, must also have the Internet Protocol (IP) address of the SMTP server. To handle issues such as spam, server administrators must control which clients can use the server. They can do this either by restricting users by their IP address or, more likely, by imposing a system or command that requires the authentication of clients.
6. It works as follows: An email server uses SMTP to send a message from an email client to another email server. The email server uses SMTP as a relay service to send the email to the receiving email server. The receiving email server uses an email client to download incoming mail via IMAP, for example, and places it in the recipient's inbox.
7. So, for example, when a user clicks the "send" button, a TCP connection is established that links to an SMTP server. When the SMTP server receives the TCP connection from a client, the SMTP process starts a connection through port 25 to send the email.
8. From here, the SMTP client tells the server what to do with information such as the sender's and recipient's email addresses and the email's content. A mail transfer agent (MTA) then checks if both email addresses are from the same email domain. If they're from the same domain, the email is sent; if not, the server uses the domain name system (DNS) to identify the recipient's domain and then sends it to the correct server. The recipient then uses IMAP or POP3 protocols to receive the email.

ESMTP: Extended Simple Mail Transfer Protocol

IMAP:
1. IMAP is a standard email retrieval protocol that manages and retrieves email messages. It keeps an email on a server and then synchronizes it across different devices. IMAP enables users to organize messages into folders, flag messages and save draft messages on the server.

2. Users can also have multiple email client applications that sync with the email server to consistently show which messages are read or unread.

POP:
1. Most email servers and clients support POP and use it to receive emails from remote servers and send them to a local client. POP is a one-way client-server protocol in which emails are received and held on the email server.
2. POP also enables users to download emails from a server to the client so the recipient can view the email offline.

DNS:
1. The domain name system (DNS) is a naming database in which internet domain names are located and translated into Internet Protocol (IP) addresses. The domain name system maps the name people use to locate a website to the IP address that a computer uses to locate that website.
2. For example, if someone types "example.com" into a web browser, a server behind the scenes maps that name to the corresponding IP address. An IP address is similar in structure to 203.0.113.72.
3. DNS servers convert URLs and domain names into IP addresses that computers can understand and use. They translate what a user types into a browser into something the machine can use to find a webpage. This process of translation and lookup is called DNS resolution.
4. How it works: When you type a domain name (such as www.example.com) into your web browser, the browser sends a request to a DNS resolver, which is a server that is responsible for resolving domain names into IP addresses. The resolver then checks its local cache to see if it has a record of the IP address for the requested domain name. If the record is not in the cache, the resolver sends a query to one or more DNS root servers to request the IP address of the top-level domain (such as .com or .net).
5. Then the root servers respond with a referral to a set of authoritative name servers that are responsible for the domain name in question. The resolver then sends a query to one of the authoritative name servers, which responds with the IP address for the requested domain name. The resolver then caches the IP address so that it can quickly respond to future requests for the same domain name.
6. Types Of DNS:
---Recursive DNS Server: This type of DNS server receives requests from client devices and searches for the IP address associated with the requested domain name. If the server does not have the IP address cached, it will recursively search the DNS hierarchy until it finds the IP address and returns it to the client.
---Authoritative DNS Server: This type of DNS server stores the IP addresses associated with domain names within a specific zone. When a recursive DNS server queries for the IP address of a domain name within that zone, the authoritative DNS server provides the IP address.
---Root DNS Server: This type of DNS server is the top-level server in the DNS hierarchy and is responsible for resolving requests for top-level domains, such as .com, .org, .net, etc. There are 13 root DNS servers distributed around the world.
---TLD (Top-Level Domain) DNS Server: This type of DNS server is responsible for resolving requests for domains within a specific top-level domain, such as .com or .org.
---Secondary DNS Server: This type of DNS server is a backup server that copies the zone information from a primary authoritative DNS server. If the primary server fails, the secondary server can respond to requests until the primary server is restored.

---Intermediate DNS Server: Also known as a forwarding DNS server, is a DNS server that is configured to forward DNS queries to other DNS servers. When a client device sends a DNS query to an intermediate DNS server, the intermediate server forwards the query to another DNS server, typically a recursive DNS server, to resolve the query.

FTP:
1. FTP (File Transfer Protocol) is a standard protocol used for transferring files over the internet. It is a client-server protocol, which means that there is a client program that connects to a server program to transfer files.
2. FTP is used for various purposes such as uploading and downloading files, managing files on a remote server, and backing up data. It is widely used by web developers to upload and manage website files on a remote server.
3. FTP uses a two-step process for transferring files: authentication and data transfer. In the authentication phase, the client connects to the server using a username and password, and if the credentials are valid, the connection is established. In the data transfer phase, the actual transfer of files occurs.

EXTRA QUESTIONS:
1. HTTP Version 1.0 VS 1.1?
A: HTTP 1.1 comes with persistent connections which define that we can have more than one request or response on the same HTTP connection. While in HTTP 1.0 you have to open a new connection for each request and response.

2. What is IP Address?
A: An IP address is a unique address that identifies a device on the internet or a local network. IP stands for "Internet Protocol," which is the set of rules governing the format of data sent via the internet or local network.

3. IPV4 VS IPV6?
A: ---IPv4 is composed of 32-bit address length and is the fourth version of the Internet Protocol (IP). IPv6 is composed of 128-bit address length and is the latest updated version of the Internet Protocol (IP).

--- IPv4 address is numerically based with 4 fields separated by a dot (.). IPv6 address is alphanumeric based consisting of 8 fields separated by a colon (:).

--- In the case of IPv4 addressing no encryption or authentication services are initiated. IPv6 provides proper encryption and authentication services for the address.

4. What is HTTP Response Status Code?
A: HTTP response status codes indicate whether a specific HTTP request has been successfully completed.

---Examples: 200-OK, 511-Network Authentication Required, 404-Not Found, 250-SMTP Ready, 354-SMTP Server Ready To Accept Messages, 221-SMTP Service Closing

5. What is ”IF-MODIFIED-SINCE”?

A: The If-Modified-Since request HTTP header makes the request conditional: the server sends back the requested resource, with a 200 status, only if it has been last modified after the given date.

6. What is Content-Length?
A: The Content-Length is the actual size of the HTTP response body in bytes.

7. What is a TCP Segment?
A: A TCP segment consists of a segment header and a data section.

8. What is "telnet smtp.gmail.com 25"?
A:
---Connecting to smtp.gmail.com on port 25 using Telnet command would attempt to establish a connection to Gmail's Simple Mail Transfer Protocol (SMTP) server. However, starting February 2021, Gmail no longer supports plain text authentication, which means that you cannot connect to Gmail's SMTP server using Telnet with the 25 port.

---Instead, you can use a secure connection with SSL/TLS to connect to Gmail's SMTP server using port 465 or 587. For example, you can use the following command to connect to Gmail's SMTP server using Telnet with SSL/TLS: "telnet smtp.gmail.com 465".

---Once the connection is established, you can then follow the SMTP protocol to send email messages. However, it's worth noting that using Telnet to send email messages is not a recommended method, as it is not secure and may expose your login credentials. It's better to use a dedicated email client or a library that supports secure authentication and encryption.

9. What is "Src Port" and "Dst Port"?
A:
---In Wireshark, "Src Port" stands for the source port and "Dst Port" stands for the destination port.
---When a device sends a packet over a network, it includes a source port and a destination port in the packet header. The source port is the port number on the sender's device that the packet is being sent from, while the destination port is the port number on the receiver's device that the packet is being sent to.

10. What is nslookup?
A: nslookup is the name of a program that lets an Internet server administrator or any computer user enter a host name (for example, "whatis.com") and find out the corresponding IP address or domain name system (DNS) record.

11. What is nslookup -type=ns google.com?
A: This causes nslookup to send a query for a type-NS record to the default local DNS server. In words, the query is saying, “Please send me the host names of the authoritative DNS for google.com”.

12. What does nslookup google.com 8.8.8.8 mean?

A: Running the command "nslookup google.com 8.8.8.8" means that you are sending a DNS query to the Google Public DNS server at IP address 8.8.8.8 to obtain the IP address associated with the domain name "google.com".

13. What is ipconfig command?
A: A command line utility that is used to display and manage the IP address assigned to the machine.

14. What is TTL in Answers Of Wireshark?
A: Time To Live, or TTL for short, is the sort of expiration date that is put on a DNS record. The TTL serves to tell the recursive server or local resolver how long it should keep said record in its cache. The longer the TTL, the longer the resolver holds that information in its cache.

15. What is TCP SYN packet?
A: SYN packets are normally generated when a client attempts to start a TCP connection to a server, and the client and server exchange a series of messages, which normally runs like this: The client requests a connection by sending a SYN (synchronize) message to the server.

16. What are HTTP Get Messages?
A:
---HTTP GET request messages are a type of HTTP request message used to retrieve a resource (such as a web page, image, or file) from a server over the Internet.
---HTTP GET requests are initiated by a client, such as a web browser, which sends a GET request message to the server containing the URL of the requested resource. The server then responds with an HTTP response message that contains the requested resource, typically in the form of HTML, CSS, JavaScript, or an image file.

17. What is a URL?
A: A URL (Uniform Resource Locator) is a unique identifier used to locate a resource on the Internet. It is also referred to as a web address. URLs consist of multiple parts -- including a protocol and domain name -- that tell a web browser how and where to retrieve a resource.

18. Images downloaded serially or parallel?
A:
---Serial image downloading means that the browser downloads images one after the other, in a sequential manner. This means that the browser will start downloading the first image and wait for it to finish downloading before moving on to the next image. This can result in slower image loading times, as the browser must wait for each image to finish downloading before it can display the page.
---Parallel image downloading, on the other hand, means that the browser downloads multiple images at the same time. This can result in faster image loading times, as the browser can download multiple images simultaneously. To achieve parallel image downloading, modern browsers can use multiple connections to the server or use a technique called "pipelining", where multiple requests are sent to the server before receiving responses.

CODE QUESTIONS:

Reliable UDP:
This is a C program that implements a simple reliable transport protocol over UDP. The program contains a client and a server. The client sends a message to the server using UDP, and the server sends an acknowledgement back to the client to confirm that the message was received. If the client does not receive an acknowledgement within a certain amount of time, it resends the message.

The protocol uses a sliding window mechanism to ensure reliable delivery of the message. The client sends a sequence of packets to the server, and the server sends an acknowledgement for each packet it receives. The client maintains a window of packets that have been sent but not yet acknowledged. If the window is full, the client stops sending packets until an acknowledgement is received and space in the window becomes available.

The protocol also includes a checksum mechanism to detect errors in the data transmission. Each packet includes a checksum value that is calculated from the contents of the packet. When a packet is received, the checksum is calculated again, and if it does not match the value in the packet, the packet is assumed to be corrupted and is discarded. The acknowledgement packets also include a checksum value.

The program is divided into several functions that perform different parts of the protocol. The main() function initializes the socket and server address, and then calls the shakeHand() function to initiate the handshake with the server. If the handshake is successful, the main() function calls the comunicate() function to send the message to the server. The comunicate() function sends a single packet and waits for an acknowledgement. If the acknowledgement is not received within a certain amount of time, the packet is resent.

The program also includes several helper functions. The noOfPck() function calculates the number of packets required to send the message. The createSum() functions calculate the checksum values for the packets and acknowledgement packets. The checkSum() functions check the checksum values of the packets and acknowledgement packets to detect errors.

The code implements a client-server communication protocol using UDP (User Datagram Protocol). Below is an explanation of all the functions in the client code:

noOfPck(int size) - calculates the number of packets required to send the message of size size in chunks of 100 bytes.

createSumControl(struct control_pkt * c_pkt) - calculates and returns the checksum for a control_pkt structure.

checkSumControl(struct control_pkt * c_pkt) - checks if the checksum of a control_pkt structure matches the calculated checksum.

createSumACK(struct ack_pkt* ack) - calculates and returns the checksum for an ack_pkt structure.

checkSumACK(struct ack_pkt * ack) - checks if the checksum of an ack_pkt structure matches the calculated checksum.

createSum(struct data_pkt* data) - calculates and returns the checksum for a data_pkt structure.

checkSum(struct data_pkt* data) - checks if the checksum of a data_pkt structure matches the calculated checksum.

shakeHand(int sock_fd,struct sockaddr_in server_addr,int sock_length,int size,struct control_pkt* c_pkt) - initiates the handshake process between the client and server. It sends a control_pkt structure to the server containing information about the number of packets to be sent and their sequence numbers. The function waits for an ack_pkt response from the server before returning.

comunicate(int sock_fd,struct sockaddr_in server_addr,int sock_length) - sends a message to the server in the form of a data_pkt structure. It waits for an ack_pkt response from the server before returning.

comunicatePkt(int sock_fd,struct sockaddr_in server_addr,int sock_length,char * msg,int sr_no) - sends a message to the server in the form of a data_pkt structure with a specified sequence number. It waits for an ack_pkt response from the server before returning.

main() - the main function of the client code. It initializes the socket, server address, and packet structures. It calls the shakeHand() function to initiate the handshake process and then enters a loop where the user can input messages to be sent to the server using the comunicate() or comunicatePkt() functions. The loop terminates when the user enters a blank message. Finally, it closes the socket and exits.

Based on the code provided, the size of the data being transmitted is 1024 bytes (data = file.read(1024)).

The number of packets that the data is being divided into would depend on the size of the file being read. Each packet is 1024 bytes, so the total number of packets would be the size of the file divided by 1024. For example, if the size of the file is 4096 bytes, then the data would be divided into 4 packets. However, since the code provided only shows the transmission of the first packet, it is not possible to determine the total number of packets without additional information.

FTP:
The given code consists of two C programs, one acting as a client and the other as a server. The client program sends the contents of a file ("test.txt") to the server program over a network connection, while the server program receives the data and saves it to another file ("get.txt").

The client program first creates a socket using the socket() function, and sets the server address and port number. It then attempts to connect to the server using the connect() function. If successful, it opens the file to be sent and calls the send_file() function to read the file data and send it to the server using the send() function. Finally, it closes the socket and file.

The server program first creates a socket using the socket() function, sets the server address and port number, and binds the socket to the server address using the bind() function. It then listens for incoming connections using the listen() function. When a client connects to the server, the accept() function is called to accept the connection and return a new socket for communication with the client. The server then calls the write_file() function to receive the data sent by the client using the recv() function and save it to a file using the fprintf() function. Finally, it closes the socket and file.

Overall, these programs demonstrate the basic process of sending and receiving data between a client and server over a network connection using sockets in C.

socket() function: This function creates a new socket and returns a file descriptor that can be used to reference the socket in other socket-related functions. It takes three arguments: the domain of the socket (e.g. AF_INET for IPv4), the type of socket (e.g. SOCK_STREAM for a TCP socket), and the protocol to be used (usually set to 0 to use the default protocol for the given domain and type).

htons() function: This function converts a short integer (16 bits) from host byte order to network byte order (big-endian).

inet_addr() function: This function converts a string containing an IPv4 address in dotted-decimal notation to a network byte order binary value.

connect() function: This function establishes a connection to a remote server over a network using the specified socket and server address. It takes three arguments: the socket file descriptor, a pointer to a struct sockaddr containing the server address and port, and the size of the sockaddr struct.

fopen() function: This function opens a file for reading or writing and returns a file pointer that can be used to reference the file in other file-related functions.

fgets() function: This function reads a line from a file into a character buffer.

send() function: This function sends data over a socket to a remote server. It takes four arguments: the socket file descriptor, a pointer to the data to be sent, the size of the data, and optional flags to modify the behavior of the function.

memset() function: This function fills a block of memory with a specified value. In this code, it is used to clear the character buffer used for reading file data before each call to fgets().

recv() function: This function receives data from a remote server over a socket. It takes four arguments: the socket file descriptor, a pointer to a buffer to store the received data, the size of the buffer, and optional flags to modify the behavior of the function.

fprintf() function: This function writes formatted data to a file. In this code, it is used to write the data received from the client to the output file.

accept() function: This function accepts an incoming connection from a remote client and returns a new socket file descriptor that can be used for communication with the client.

bind() function: This function binds a socket to a specific network address and port. It takes three arguments: the socket file descriptor, a pointer to a struct sockaddr containing the server address and port, and the size of the sockaddr struct.

listen() function: This function sets the socket to listen for incoming connections. It takes two arguments: the socket file descriptor and the maximum number of connections that can be queued for the socket.

setsockopt() function: This function sets various socket options. In this code, it is used to set the SO_REUSEADDR and SO_REUSEPORT options to allow multiple sockets to bind to the same address and port.

printf() function: This function writes formatted output to the standard output stream.

fprintf() function: This function writes formatted output to a file.

memset() function: This function fills a block of memory with a specified value. In this code, it is used to clear the character buffer used for receiving data from the client before each call to recv().

TIME CONTROL:
This code seems to be an implementation of a client-server communication using UDP. The code starts by importing the necessary libraries and defining some constants and structures that will be used in the program.

The program defines three structures:

control_pkt: contains information about the number of packets, sequence number and checksum
data_pkt: contains the message to be sent along with a sequence number and a checksum
ack_pkt: contains an acknowledgement number, sequence number, and a checksum
The noOfPck function calculates the number of packets required to send the entire message, assuming that each packet carries a maximum of 100 characters. The createSumControl, createSumACK, and createSum functions calculate the checksums for the control_pkt, ack_pkt, and data_pkt, respectively. The checkSumControl, checkSumACK, and checkSum functions check whether the received packet's checksum matches the expected checksum.

The shakeHand function handles the initial handshake between the client and server. The client sends a control_pkt to the server indicating the number of packets to be sent. The ack_pkt received from the server confirms the handshake.

The comunicate function sends the message from the client to the server in one or more packets and receives an acknowledgement for each packet. The comunicatePkt function sends individual packets to the server, and it is called multiple times by the comunicate function.

The main function initializes a socket, sets the server's IP address and port number, and calls the shakeHand and comunicate functions to communicate with the server. However, the code seems to be incomplete, as some parts of the code are missing, such as the socket creation and initialization of the server_addr variable.