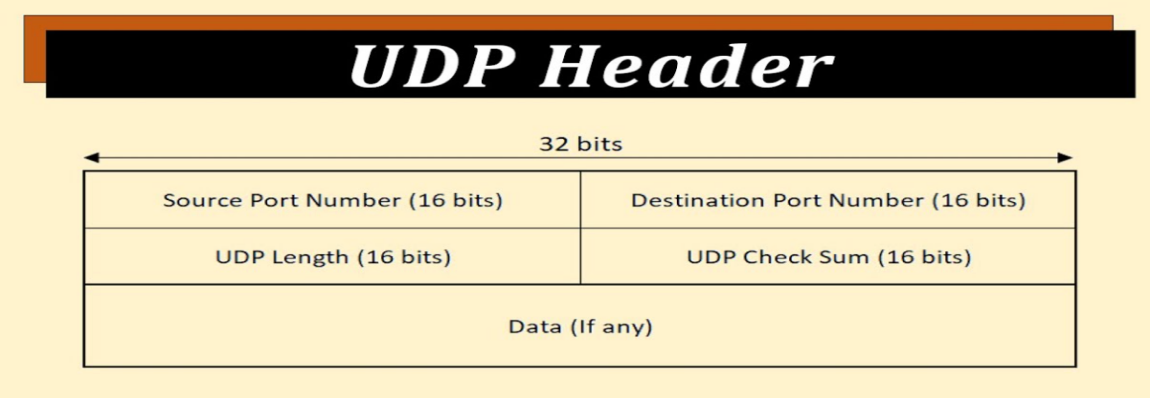
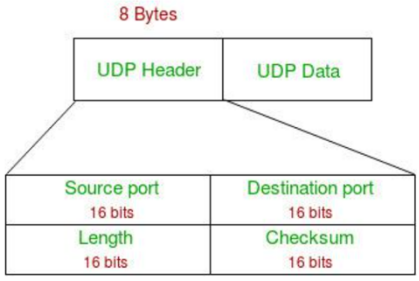
Port Number	Protocol
20, 21	File Transfer Protocol (FTP)
22	Secure Shell (SSH)
23	Telnet Protocol
25	Simple Mail Transfer Protocol (SMTP)
53	Domain Name System (DNS)
67, 68	Dynamic Host Configuration Protocol (DHCP)
80	HyperText Transfer Protocol (HTTP)
110	Post Office Protocol (POP3)
137	NetBIOS Name Service
143	Internet Message Access Protocol (IMAP4)
443	Secure HTTP (HTTPS)
445	Microsoft-DS (Active Directory)



UDP Header

UDP header is an **8-byte** fixed and simple header, while for TCP it may vary from 20 bytes to 60 bytes. The first 8 Bytes contain all necessary header information and the remaining part consists of data. UDP port number fields are each 16 bits long, therefore the range for port numbers is defined from 0 to 65535; port number 0 is reserved. Port numbers help to distinguish different user requests or processes.

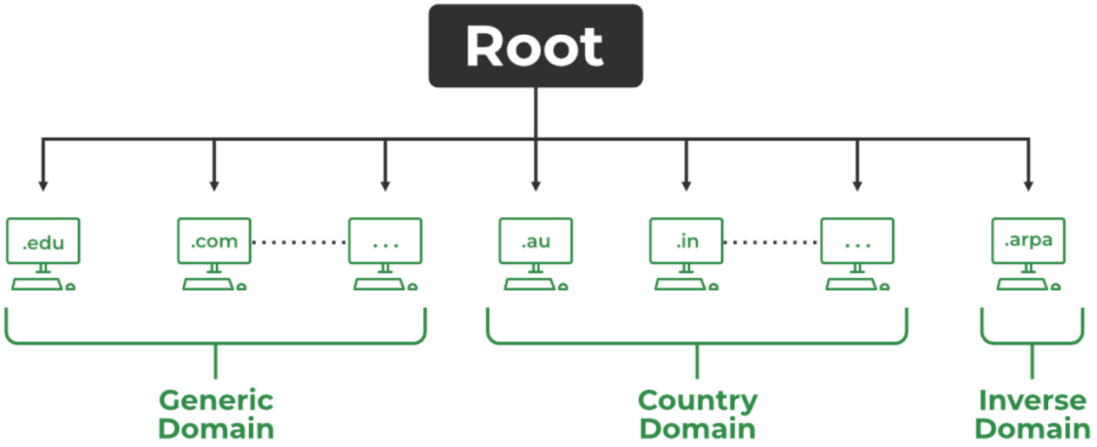


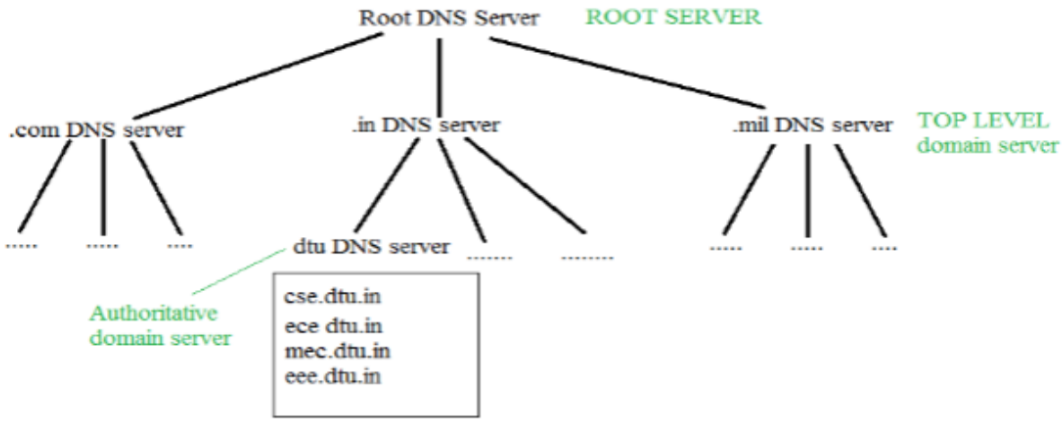
UDP Header

- **Source Port:** Source Port is a 2 Byte long field used to identify the port number of the source.
- **Destination Port:** It is a 2 Byte long field, used to identify the port of the destined packet.
- **Length:** Length is the length of UDP including the header and the data. It is a 16-bits field.
- **Checksum:** Checksum is 2 Bytes long field. It is the 16-bit one's complement of the one's complement sum of the UDP header, the pseudo-header of information from the IP header, and the data, padded with zero octets at the end (if necessary) to make a multiple of two octets.

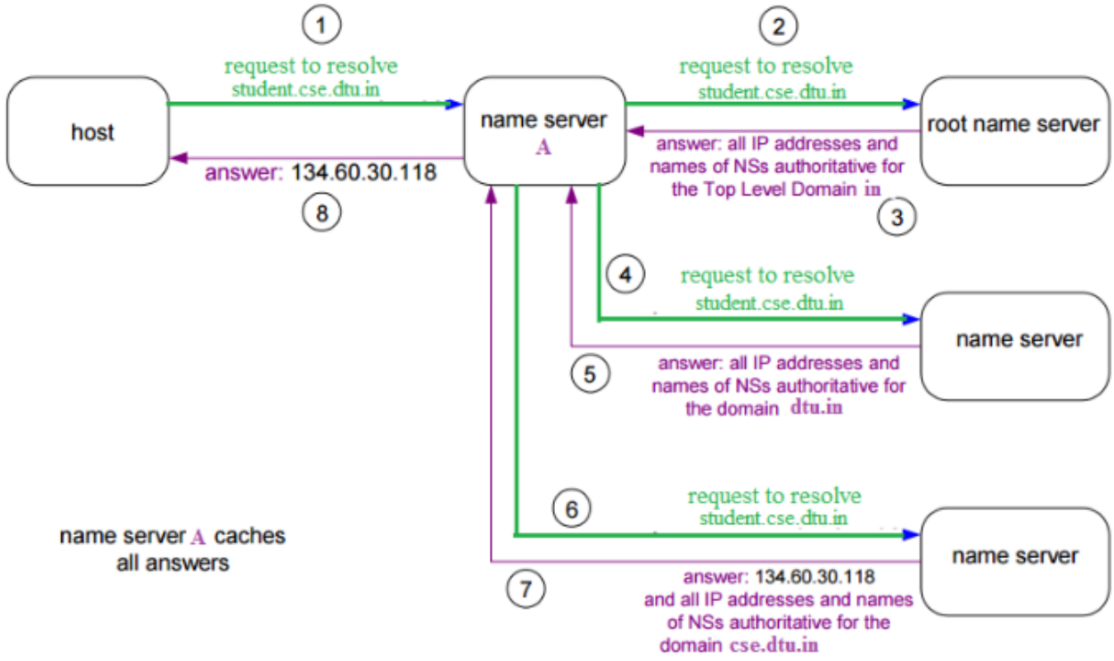
Notes – Unlike TCP, the Checksum calculation is not mandatory in UDP. No Error control or flow control is provided by UDP. Hence UDP depends on IP and ICMP for error reporting. Also UDP provides port numbers so that is can differentiate between users requests.

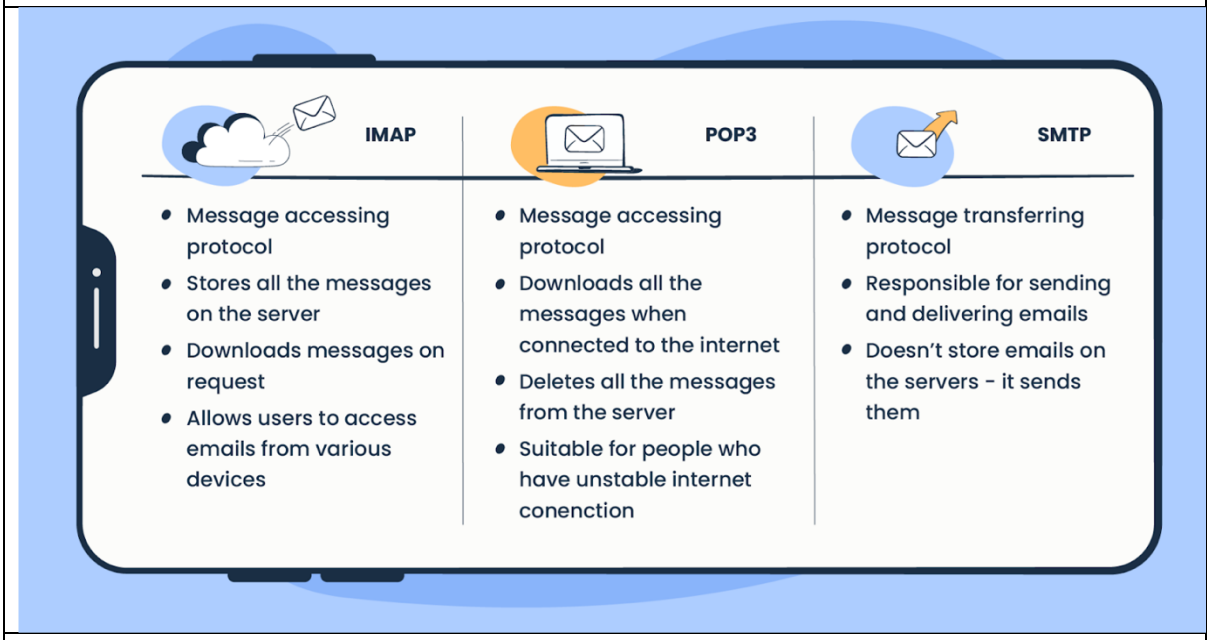
DNS





A host wants the IP address of cse.dtu.in





POP3 downloads emails from a server to a single computer, making those emails only accessible on that specific computer. IMAP stores emails on a server and then syncs them across multiple devices. IMAP is more advanced than POP3 and allows you to access your email from anywhere, and on any device.

What is WPA?

Wi-Fi Protected Access (WPA) is a security standard for computing devices equipped with wireless internet connections. WPA was developed by the Wi-Fi Alliance to provide more sophisticated data encryption and better user authentication than Wired Equivalent Privacy (WEP), the original Wi-Fi security standard.

WPA was initially released in 2003. The Wi-Fi Alliance defined WPA as a response to serious weaknesses found in the WEP protocol. A more secure version, WPA2, was released in 2004. In 2018, the Wi-Fi Alliance announced the release of WPA's third and current version, WPA3.

What is the Secure Shell (SSH) protocol?

The Secure Shell (SSH) protocol is a method for securely sending commands to a computer over an unsecured network. SSH uses cryptography to authenticate and encrypt connections between devices.

SSH is commonly implemented using the **client-server model**. One computer is called the **SSH client** and another machine acts as the **SSH server**, or **host**.

HTTPS, or Hyper Text Transfer Protocol Secure, is also another protocol that encrypts data. So what is the difference between SSH and HTTPS? HTTPS allows web browsers to communicate with servers to display websites. SSH allows for shells to enable data exchange or communication between two devices, not just browsers and a server. Shells allow you to talk to operating systems.

When you connect to an SSH server, you are dropped into a shell. This shell can be a Linux terminal shell or a Windows command prompt shell where you can execute commands on the machine you are connected to. When you use terminal or command line, you are talking to your operating system. With SSH, you can talk to remote operating systems too.

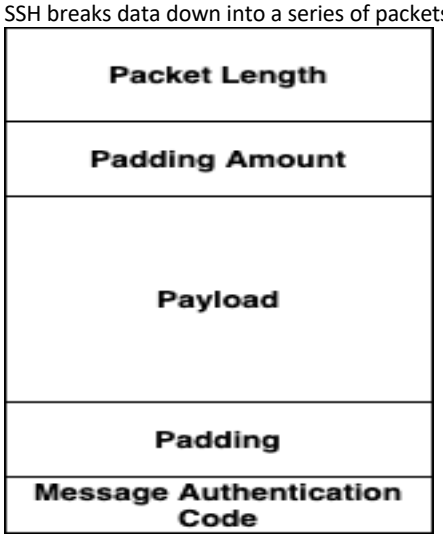
What Can We Transfer With SSH?

SSH can be used to transmit:

- Data
- Commands
- Text
- Files (Using SFTP: Secure File Transfer Protocol, basically an encrypted version of FTP that makes it that man-in-the-middle attacks are not possible)

How does SSH work?

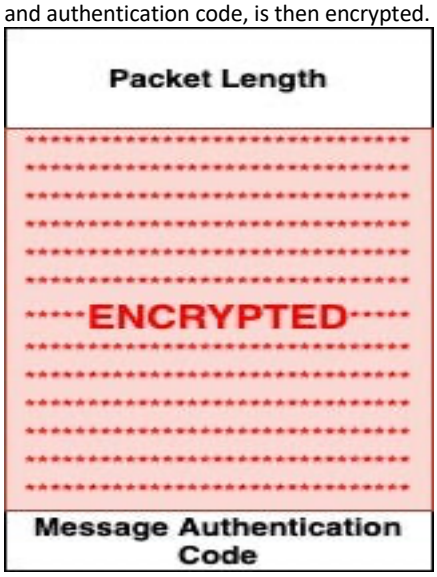
SSH breaks data down into a series of packets. Like any packet transfer, there are a few fields at the beginning.



A Packet

At the top, the **Packet Length** tells you how big the packet is. After that, you have another byte, **Padding Amount**, tells you how much padding there is. Then you have your data, the **Payload**. Following the payload, you have **Padding**. That padding is random bytes that don't mean anything but are encrypted along with the payload to make it even harder to detect the data because you've thrown in this random extra data. Finally, you have a **Message Authentication Code** so that you can be sure the data has not been tampered with.

The payload can also be compressed using standard compression algorithms. The whole packet, excluding the length and authentication code, is then encrypted.



An Encrypted Packet

The packet is then sent to the server. The server decrypts the packet and decompresses the payload to extract the data. The same process is done for every packet sent over the connection.

To keep SSH secure, SSH uses three different types of data manipulation techniques at various points during a transmission. The three techniques used in SSH are:

1. Symmetrical Encryption
2. Asymmetrical Encryption
3. Hashing

Symmetrical Encryption

Symmetric encryption is the type of encryption where one key can be used to encrypt messages sent to the destination and also decrypt messages received at the destination. This encryption scheme is also known as a ***shared secret encryption***, or a ***shared key encryption***.

Diagram 1: Symmetrical Encryption

Both devices use the same key to encrypt data they are sending over and decrypt data they receive. A secret key is specific to each SSH session. This is the type of encryption that is used to encrypt the entire SSH connection to stop man-in-the-middle attacks from being able to read the data because they do not have this secret key.

One problem that arises is the initial key exchange. If a third party is listening during the key exchange, they would now know the key and be able to decrypt all our messages. One way to prevent this is by using a Key Exchange Algorithm.

A **Key Exchange Algorithm** is a secure way to exchange secret keys without interception. This is done by two computers exchanging public data and then manipulating that data independently to derive the secret key. In order to implement a key exchange algorithm, we need Asymmetrical Encryption.

Asymmetrical Encryption

Asymmetrical encryption is encryption through the use of two separate keys for encryption and decryption, a public key and a private key. The public key can be can be shared with anyone but the private key is never shared. A public key and a private key form a ***key pair***. A message that is encrypted with a machine's public key can only be decrypted by its private key. The public key is stored on the SSH server and the private key is stored locally on the SSH client.

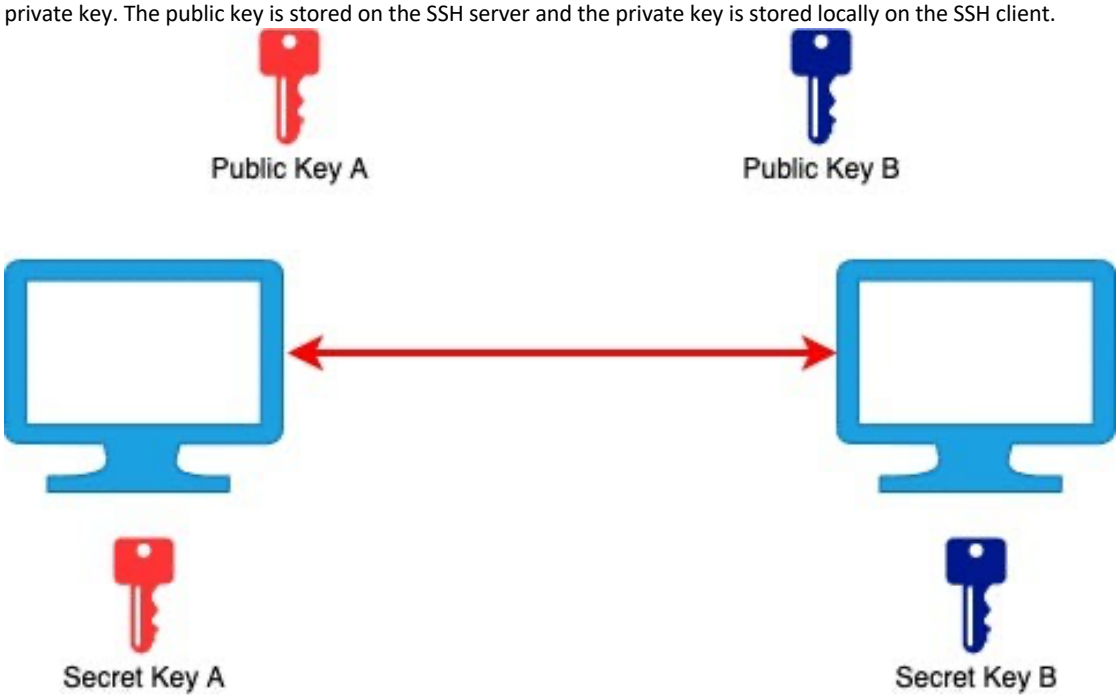


Diagram 2: Asymmetrical Encryption

If I give you my public key, you can send me a message by encrypting it with my public key. I will then be able to read it by decrypting it using my private key.

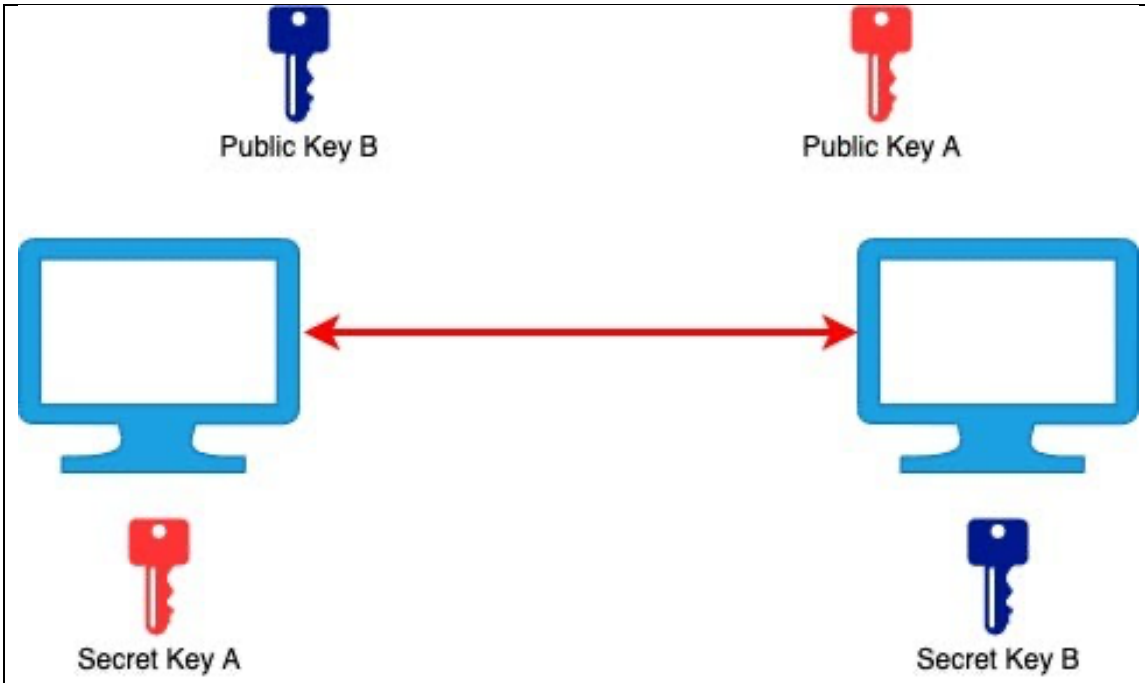


Diagram 3: Asymmetrical Encryption key exchange

Even if a third party manages to obtain a public key, they will not be able to decrypt any messages because they don't have the private key. So long as the private key is never sent over and is secure on your device, your messages can't be decrypted.

SSH uses asymmetrical encryption in a few places such as the key exchange algorithm used to set up the symmetrical encryption. Asymmetrical encryption is also used as the key that can be used to SSH into a server without the use of a password. We exchange the algorithm to generate the keys (the yellow keys in diagram 1) used to encrypt and decrypt messages.

Both devices generate temporary public and private keys and share their respective public keys. They then independently generate a new symmetric key that both devices will use to encrypt and decrypt messages. This generation is done using the **Diffie Hellman key exchange**.

At the start of a Diffie Hellman key exchange, the two devices need to agree on a few parameters that they will use for the key exchange

HTTP Non-Persistent & Persistent Connection

The Hypertext Transfer Protocol (HTTP) is an application-level protocol that uses TCP as an underlying transport and typically runs on port 80. HTTP is a stateless protocol i.e. server maintains no information about past client requests.

HTTP Connections

1. Non-Persistent
2. Persistent

Basic Pre-Requisite

The terminology which we must know before going deep into Persistent & Non-Persistent Connections is

1. RTT(Round Trip Time)
2. TCP 3-Way Handshake

1. RTT: Time for a small packet to travel from client to server and back.

RTT = 2 X propagation time

1. For a connection Persistent or Non-persistent it is sure that to initiate a TCP connection one RTT is used.

2. One RTT is used for the HTTP request and the first few bytes to the HTTP response to return. So to know the total file transmission time.

Total = 2RTT + transmit time

2. TCP 3-Way Handshake: TCP Connection establishes in 3 ways, that's why it is called a 3-way Handshake.

- Requesting the server for the connection.
- The server responds to whether the connection can be established or not.
- Acknowledgment by the client on the response sent by the server.

Difference between Persistent and Non-Persistent Connections

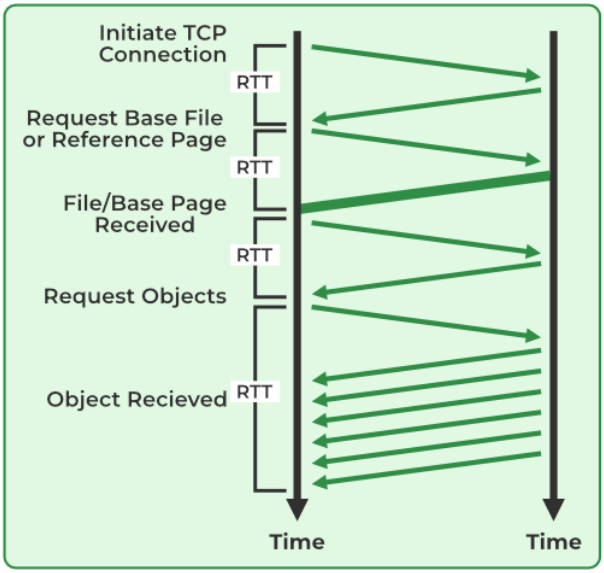
Persistent HTTP	**Non-Persistent HTTP**
The server leaves the connection open after sending a response.	Requires 2 RTTs per object.
Subsequent HTTP messages between the same client/server are sent over an open connection.	OS overhead for each TCP connection
The client sends requests as soon as it encounters a referenced object.	Browsers often open parallel TCP connections to fetch referenced objects.
As little as one RTT for all the referenced objects.	Here, at most one object can be sent over one TCP Connection.

Non-Persistent Connection

Non-Persistent Connections are those connections in which for each object we have to create a new connection for sending that object from source to destination. Here, we can send a maximum of one object from one TCP connection. There are two types:

1. Non-Persistent-Without parallel connection: Each objection takes two RTTs (assuming no window limit) one for TCP connection and the other for HTTP image/text file.

2. Non-Persistent-With parallel connection: Non-Persistent with a parallel connection requires extra overhead in transferring data.



Non-Persistent & Parallel Connection

Advantages of Non-Persistent Connection

1. Wastage of Resources is very less because the connection opens only when there is some data to be sent.
2. Non-Persistent Connection is more secure because after sending the data, the connection gets terminated and nothing can be shared thereafter.

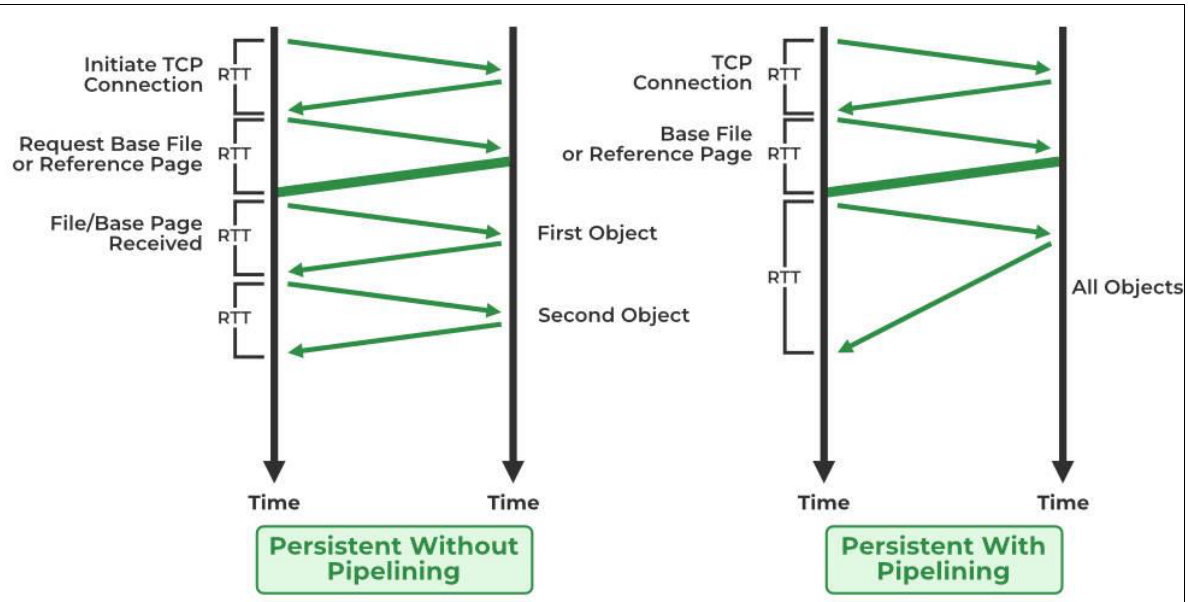
Disadvantages of Non-Persistent Connection

1. In Non-Persistent Connection, it requires a greater CPU overhead for the transmission of data

Persistent Connection

1. Non-Pipelined Persistent Connection: In a Non-pipeline connection, we first establish a connection that takes two RTTs then we send all the object's images/text files which take 1 RTT each (TCP for each object is not required).

2. Pipelined Persistent Connection: In Pipelined connection, 2RTT is for connection establishment and then 1RTT(assuming no window limit) for all the objects i.e. images/text.



Persistent Without Pipelining and with Pipelining

Advantages of Persistent Connections

- Lower CPU and memory usage because there is less number of connections.
- Allows HTTP pipelining of requests and responses.
- Reduced network congestion (fewer TCP connections).
- Reduced latency in subsequent requests (no handshaking).
- Errors can be reported without the penalty of closing the TCP connection.

Disadvantages of Persistent Connections

- Resources may be kept occupied even when not needed and may not be available to others.
- Most modern browsers like Chrome, Firefox, and Internet Explorer use persistent connections.

Consider the resolution of the domain name $www.gate.org.in$ by a DNS resolver. Assume that no resource records are cached anywhere across the DNS servers and that iterative query mechanism is used in the resolution. The number of DNS query-response pairs involved in completely resolving the domain name is .

A 2

B 3

C 4

D 5

Which of the following protocol pairs can be used to send and retrieve e-mails (in that order)?

A IMAP POP3

B SMTP, POP3

C SMTP, MIME

D IMAP, SMTP

Identify the correct sequence in which the following packets are transmitted on the network by a host when a browser requests a webpage from a remote server,assuming that the host has just been restarted.

A. HTTP GET request, DNS query, TCP SYN

B. DNS query, HTTP GET request, TCP SYN

C. DNS query,TCP SYN, HTTP GET request

D. TCP SYN, DNS query, HTTP GET request

Assume that you have made a request for a web page through your web browser to a web server. Initially the browser cache is empty. Further, the browser is configured to send HTTP requests in non-persistent mode. The web page contains text and five very small images.The minimum number of TCP connections required to display the web page completely in your browser is_____.

A. 1

B. 6

C. 5

D. 2

A user starts browsing a webpage hosted at a remote server. The browser opens a single TCP connection to fetch the entire webpage from the server. The webpage consists of a top-level index page with multiple embedded image objects. Assume that all caches (e.g., DNS cache, browser cache) are all initially empty. The following packets leave the user's computer in some order.

(i) HTTP GET request for the index page
(ii) DNS request to resolve the web server's name to its IP address
(iii) HTTP GET request for an image object
(iv) TCP SYN to open a connection to the web server

Which one of the following is the CORRECT chronological order (earliest in time to latest) of the packets leaving the computer ?

A. (iv), (ii), (iii), (i)

B. (ii), (iv), (iii), (i)

C. (ii), (iv), (i), (iii)

D. (iv), (ii), (i), (iii)

Suppose in a web browser, you click on the www.gate-2023.in URL. The browser cache is empty. The IP address for this URL is not cached in your local host, so a DNS lookup is triggered (by the local DNS server deployed on your local host) over the 3-tier DNS hierarchy in an iterative mode. No resource records are cached anywhere across all DNS servers.
Let RTT denote the round trip time between your local host and DNS servers in the DNS hierarchy. The round trip time between the local host and the web server hosting www.gate-2023.in is also equal to RTT. The HTML file associated with the URL is small enough to have negligible transmission time and negligible rendering time by your web browser, which references 10 equally small objects on the same web server.
Which of the following statements is/are CORRECT about the minimum elapsed time between clicking on the URL and your browser fully rendering it?

A. 7 RTTs, in case of non-persistent HTTP with 5 parallel TCP connections.

B. 5 RTTs, in case of persistent HTTP with pipelining.

C. 9 RTTs, in case of non-persistent HTTP with 5 parallel TCP connections.

D. 6 RTTs, in case of persistent HTTP with pipelining.