Ans1:

A.)

The port number of the destination host will be used as a host to identify the processes running on the host. Different port numbers associated with different processes, for example, port numbers 20,21 associate with FTP, so if a host finds port 20,21 open at the destination host, they can assume the destination host has FTP service running.

B.)

1.) It would be pretty much impossible to manage a centralized host-address table with millions of websites and Billions of DNS requests every minute. The processing power needed to establish such a server would not be cost-effective at all

2.) It would be a single point of failure; any failure could have the potential to bring the entire internet down.

3.) The server would be terrible at scalability. Let's say in the next 15 years the number of host requests double, it wouldn't be easy to simply add more servers to deal with the extra load like we would be able to in a decentralized DNS system.

C.)

For long-distance network paths, I would prefer to use Persistent HTTP over Non-Persistent HTTP. Having a long-distance path means a high propagation delay and using Non-Persistent HTTP would result in having to spend twice the amount of propagation time on every object we send. By using a persistent HTTP connection, we can cut that time down to just 1x the propagation delay on each object, thus taking less time overall.

D.)

1.) There can exist certain scenarios, where receiving some data but immediately is more important than receiving everything but with a delay. TCP in some cases might cause some delays while guarantying delivery but when it comes to systems like online gaming delays are receiving data late can be way worse than receiving corrupted data.

2.) UDP allows for broadcasting, while TCP doesn't. Many applications might use the broadcasting capabilities of UDP for their proper functioning, which can be way better for system resources than forming multiple TCP connections to establish the same. DNS, for example, uses UDP to broadcast to multiple servers.

E.)

1.) IMAP allows the mail to be stored in a remote server where you could access the mail online unlike POP3 where you had to download the email to view it.

2.) IMAP also made it possible for E-mail clients to exist in the form they do, with POP3 you could only access your email at a single place, but IMAP made it possible for modern email clients to work the way they do.

3.) IMAP allows the user to create hierarchies and folders to manage their E-mail while POP3 does not offer any kind of functionality like it.

F.)

1.) SMTP is a protocol used by E-Mail servers to communicate with each other while HTTPS is a protocol that defines how web servers communicate with a client.

2.) HTTP is a pull protocol; SMTP is a push protocol.

3.) HTTP encapsulates each object on a web page in its own HTTP response message while SMTP encapsulates the entire message/mail in a single response.

4.) SMTP uses port 25 and HTTP uses port 80.

<u>Ans2:</u>

File sizes = L bits

Total Objects = M + 1 Base File

Bandwidth = R

RTT = T

Propagation delay (for each object) = RTT/2 = T/2

Transmission delay (for each object) = L/R

Initiation Time refers to propagation delay of initiating a connection and receiving confirmation

Base File Time refers to propagation delay of requesting base file + receiving the first bit of the file

a) *Non-Persistent Connection*
The connection is non persistent so the connection will have to be re-established for every single object after the initial file.
Time one object takes = Initiation Time + Base File Time + Transmission Delay
= T + T + L/R
= 2T + L/R
Now this will be repeated M more times hence
(M+1)(2T + L/R)

b) *Persistent*
The connection is persistent, so the connection won't have to be re-established for
Time = Initiation Time + Base File Time + propagation time of each object + number of objects x Transmission Delay
= T + T + M*T + (M+1)(L/R)
= **(M+2)T + (M+1)(L/R)**

c) *Persistent with Pipelining*
In Pipelining with a persistent connection, we can send request all the data in 1RTT.
Hence,
Time = Initiation Time + Base File Time + Time to request all the data + Transmission Delay of all Objects
= T + T + T + (M+1)(L/R)
= **3T + (M+1)(L/R)**

d) *Non-Persistent with Parallel Connections*
Assuming 5 Parallel Connections and M = 10
Here, 10 objects can be sent parallelly after receiving the base file
Hence the time can be written as
= Initiation Time + Base File Time + Transmission and propagation Delay of 10 objects with 5 Parallel Connections
Browser with 5 parallel Connections, each connection will get 1/5th of the Bandwidth = R/5
= T + T + L/R + (10/5) (T + T + (5L/R)
= 2T + L/R + 4T + 10L/R
= **6T + 11L/R**

Ans3:

Given that UDP 'is fully identified by' a two-tuple design; Destination IP and Destination Port number[1] we can conclude that even if we receive two UDP segments from different IP addresses they'll be received by the same destination socket because we cannot differentiate between the two hosts using UDP datagram.

Because we cannot differentiate between the two hosts using a UDP datagram there is no apparent way for the packets being received to be distinguished. UDP is a connection-less protocol hence the socket will receive datagrams from both the clients in whatever way the server machine receives them.

Ans4:

Yes, a sperate connection socket would be created for both the requests. Given that it's a persistent connection hence it's a TCP connection, Host C will examine the IP Address, Destination Port, Destination IP address and Source Port when it receives a datagram to decide which socket does the datagram it received belongs to. This is how requests from Host A and Host B are differentiated and passed through different sockets.

Ans5:

Request

a) Index.html
b) www-net.cs.umass.edu/index.html
c) Application Layer doesn't require an IP Address.
d) Persistent
e) HTTP1.1, there are 5 versions of HTTP(0.9, 1.0, 1.1, 2.0, 3.0)
f) Firefox. Browser information is provided for statistics and to load browser-compatible functions.

Response

a) Yes. The time the document was provided was 20:09:20 GMT
b) Document was last modified on 30 Oct 2007 17:00:02 GMT
c) 2652 Bytes.
d) Yes, the server agrees to a persistent connection.

Ans6:

1. DNS TYPE here is **NS**. When DNS loop up TYPE is NS, the performed function delegates to use the specified authoritative name server.

2. DNS TYPE here is **A**. Type here returns aN IPv4 address, which maps the domain name to an IP Address.

3. DNS TYPE here is **CNAME**. CNAME is used to get alias of the entered name, the data received will point to another canonical name of the entered domain.

4. DNS TYPE here is **MX.** MX is used to map the domain to a list of message transfer agents for that domain, a.k.a return a list of mail servers responsible for managing E-Mails for that domain.

5. DNS TYPE here is **PTR.** Also called pointer to a canonical name, PTR returns the domain name of an entered IP Address.

Ans7:

C. max(S/D, S/C, S/(C/K + U))

Let's start with the file size = S

Server Upload Rate = C

Uniform Download Rate = D

Total Clients = K

Time it takes for each file to upload from server = **S/C**

Time it takes for each file to download on clients = **S/D**

Total data to be transmitted = KS

Total Max Bandwidth = C + KD

Client Upload Speed = U

Hence Total time it would take on the network = KS / C + UK

= S/(C/K + U)

Hence maximum would be the maximum of **S/C , S/D , S/(C/K + U)**

References:

[1] James Kurose, Keith Rose – A Top-Down Approach 7th Edition, pg. 232.