# 50.012 Networks (2021 Term 6)

# Homework 1

Hand-out: 31 Jan

Due: 14 Feb 23:59

Name: Lim Boon Han Melvin Student ID: 1005288

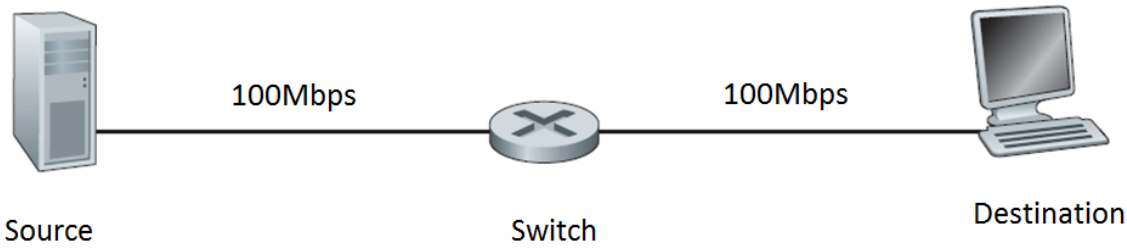
**1.** Why two ISPs peer with each other? (Hint: consider the different types of peering: Regional ISP with Regional ISP, Tier 1 with Tier 1, and Regional ISP with content provider) What is the role of an IXP? How does an IXP generate revenue? (Hint: study some IXP, e.g., https://www.sgix.sg/)

Peering reduces costs where peers can request or provide services to one another and communicate without an always-on server and increases scalability when new peers bring in new services. However, since peering can only be done with ISPs on the same level, we would still require the link between ISPs, Tier 1 networks, and content providers. Peering reduces the access to the links which reduces costs.

IXPs are Internet exchange points, where ISPs and content providers of different networks can connect with each other and exchange data. They generate revenue selling their port-based services where content providers and ISPs can have a port for them to hold their services.

**2.** (2019's mid-term exam question): Calculate the end-to-end delay (i.e., the duration from the first bit sent by the source to the last bit received by the destination) for a packet with size of 1500 bytes (12,000 bits) for the following settings:

**2.1** The source and the destination are connected via a store-and-forward switch (as in the figure below). Assume that each link in the figure has a propagation delay of 12 µs (1 µs = $10^{-6}$s) and operates at 100Mbps (consider 1M = $10^6$). The switch begins forwarding immediately after it finishes receiving the whole packet. Assume zero processing and queueing delay.




**Single switch case**

$d_{prop} = 12\,\mu s = 12 \times 10^{-6}\,s = 0.012\,ms$,

Throughput = 100 Mbps = $100 \times 10^6$ bps

Packet size = 12 000 bits

$d_{nodal} = d_{proc} + d_{queue} + d_{trans} + d_{prop}$

$d_{trans} = \frac{\text{Packet size}}{\text{Throughput}}$

$= \frac{12\,000}{100 \times 10^6}$

$= 1.2 \times 10^{-4}$

$d_{nodal} = 0 + 0 + 1.2 \times 10^{-4} + 12 \times 10^{-6}$

$= 1.32 \times 10^{-4}$,

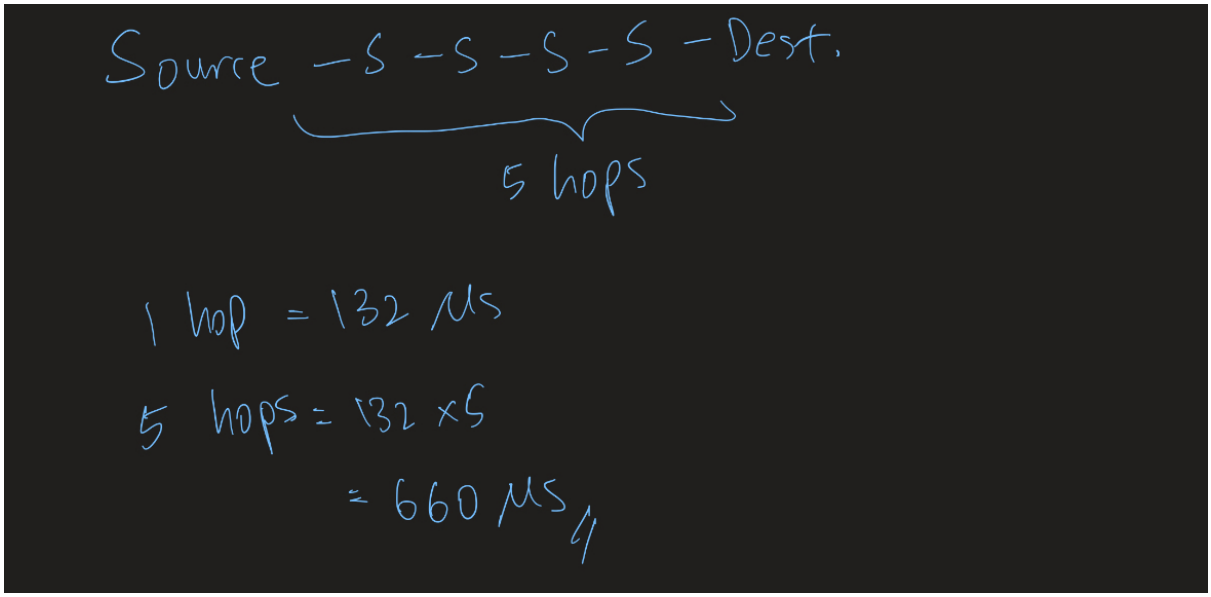
end - end delay = $d_{nodal} \times 2$

$= (1.32 \times 10^{-6}) \times 2$

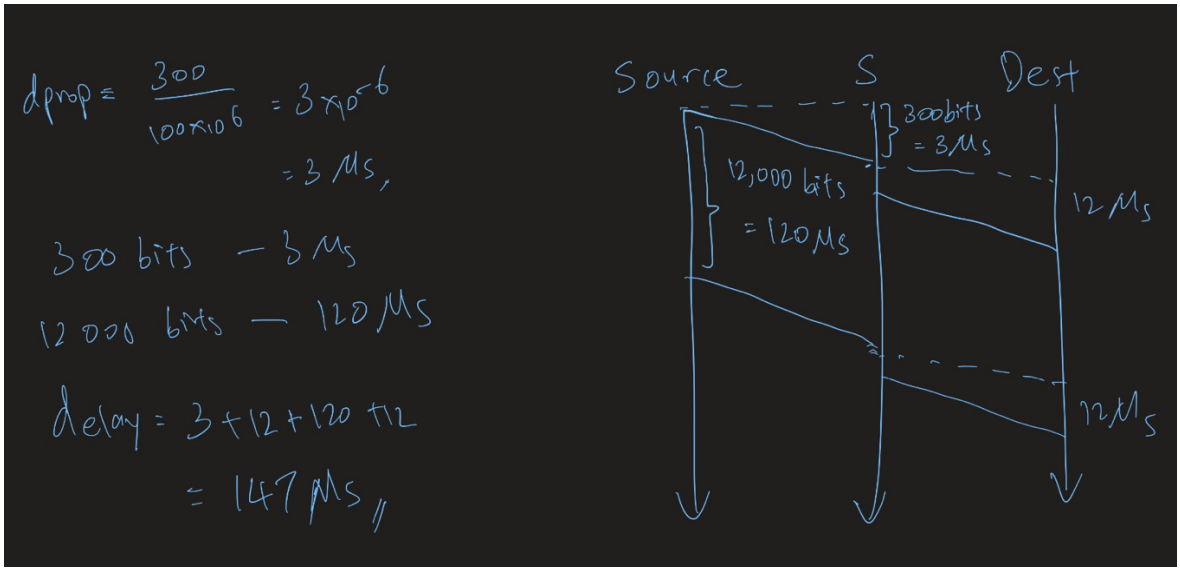
$= 264 \times 10^{-6}\,s$

$= 264\,\mu s$

**2.2** Same scenario as **2.1** above, calculate the end-to-end delay when there are four switches chained together (each switch and each link has the same setting as in scenario **2.1**) in the path between the source and the destination.




**2.3** Same scenario as **2.1** above, i.e., there is only a single switch between the source and the destination, but assume the switch implements "cut-through" switching, i.e., the switch begins to forward the packet after the first 300 bits of the packet have been received. Calculate the end-to-end delay. (Hint: Draw a space-time diagram.)



**3.** (adapted from textbook problem chapter 2, problem 4) Consider the following string of ASCII characters that were captured by Wireshark when the browser sent an HTTP GET message (i.e., this is the actual content of an HTTP GET message). The characters *<cr><lf>* are carriage return and line-feed characters (that is, the italized character string *<cr>* in the text below represents the single carriage-return character that was contained at that point in the HTTP header). Answer the following questions, indicating where in the HTTP GET message below you find the answer.

GET /cs453/index.html HTTP/1.1*<cr><lf>*Host: gaia.cs.umass.edu*<cr><lf>*User-Agent: Mozilla/5.0 (Windows;U; Windows NT 5.1; en-US; rv:1.7.2) Gecko/20040804 Netscape/7.2 (ax) *<cr><lf>*Accept: ext/xml, application/xml, application/xhtml+xml, text/html;q=0.9, text/plain;q=0.8, image/png,*/*;q=0.5*<cr><lf>*Accept-Language: en-us,en;q=0.5*<cr><lf>*Accept-Encoding: zip,deflate*<cr><lf>*Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7*<cr><lf>*Keep-Alive: 300*<cr><lf>*Connection: keep-alive*<cr><lf><cr><lf>*

**3.1** What is the URL of the document requested by the browser? (Hint: please look up the syntax rules of URL and includes all parts)

URL: gaia.cs.umass.edu, specifically the /cs453/index.html page. The "host" field corresponds to the server name, which is gaia.cs.umass.edu while the fields before is the HTTP version and specific html page requested.

Found at: sHost: gaia.cs.umass.edu and /cs453/index.html



**3.2** What version of HTTP is the browser running?

HTTP version 1.1

Found at: HTTP/1.1

**3.3** Does the browser request a non-persistent or a persistent connection? What are the pros and cons of using a non-persistent connection as compared to using a persistent connection?

The browser requested a persistent connection, as denoted by the "connection" field. This means that once a connection is established, both end systems can send data over this connection without having to open a new port for the same use.

Found at: Connection: keep-alive

Pros (non-persistent): No wastage of connection ports as they are only open if needed.

Cons (non-persistent): Unreliable as data may be delivered out-of-order / lost if connection is lost halfway. There is also a resource overhead on both systems to open a port whenever a connection is requested.

Pros (persistent): Saves resources that are used to constantly open ports for connection, lower latency.

Cons (persistent): Waste of network resources if the browser is idle (though newer browsers are smart enough to use these resources on other pages if not needed), may cause deadlocks if too many connections established.

**3.4** What is the IP address of the host on which the browser is running?

The IP address would be the address of gaia.cs.mass.edu, the GET message header and values does not have the exact IP address.

**3.5** What type of browser initiates this message? Why is the browser type needed in an HTTP request message?

Mozilla version 5.0. This is to identify the browser to adapt the webpage or throw errors accordingly. This is found under the "User-Agent" field.

Found at: User-Agent: Mozilla/5.0 (Windows;U; Windows NT 5.1; en-US; rv:1.7.2) Gecko/20040804 Netscape/7.2 (ax)

**3.6** How many bytes are there for the request line (i.e., the first line in a request message, including *<cr><lf>*) in this message?

Header size is 32 bits = 4 bytes.

**4.** (textbook problem chapter 2, problem 5) The text below shows the reply sent from the server in response to the HTTP GET message in the question above. Answer the following questions, indicating where in the message below you find the answer.

HTTP/1.1 200 OK*<cr><lf>*Date: Tue, 07 Mar 2008 12:39:45GMT*<cr><lf>*Server: Apache/2.0.52 (Fedora)*<cr><lf>*Last-Modified: Sat, 10 Dec2005 18:27:46 GMT*<cr><lf>*ETag: "526c3-f22-a88a4c80"*<cr><lf>*Accept-Ranges: bytes*<cr><lf>*Content-Length: 3874*<cr><lf>*Keep-Alive: timeout=max=100*<cr><lf>*Connection: Keep-Alive*<cr><lf>*Content-Type: text/html; charset=ISO-8859-1*<cr><lf><cr><lf>*<!doctype html public "//w3c//dtd html 4.0 transitional//en">*<lf>*<html>*<lf>*<head>*<lf>* <meta http-equiv="Content-Type"content="text/html; charset=iso-8859-1"><lf> <meta name="GENERATOR" content="Mozilla/4.79 [en] (Windows NT 5.0; U) Netscape]">*<lf>* <title>CMPSCI 453 / 591 / NTU-ST550A Spring 2005 homepage</title>*<lf>*</head>*<lf>* <much more document text following here (not shown)>

**4.1** Was the server able to successfully find the document or not? What time was the document reply provided?

Status code of 200 and phrase "OK" implies it was successful. Reply was on Tuesday, 7 March 2008, 12:39:45 GMT. As stated under the "Date" header.

Found at: HTTP/1.1 200 OK and Date: Tue, 07 Mar 2008 12:39:45GMT.

**4.2** When was the document last modified?

Saturday, 10 December 2005 18:27:46 GMT, found under the field "Last-Modified".

Found at: Last-Modified: Sat, 10 Dec2005 18:27:46 GMT

**4.3** How many bytes are there in the document being returned?

This can be found under the header "Content-Length", it stated a total of 3874 bytes.

Found at: Content-Length: 3874

**4.4** What are the first 5 bytes of the document being returned? Did the server agree to a persistent connection?

Since data is after the header lines, the first 5 bytes returned were <!doct after the "Content-Type:" header. The server agreed to the persistent connection as well.

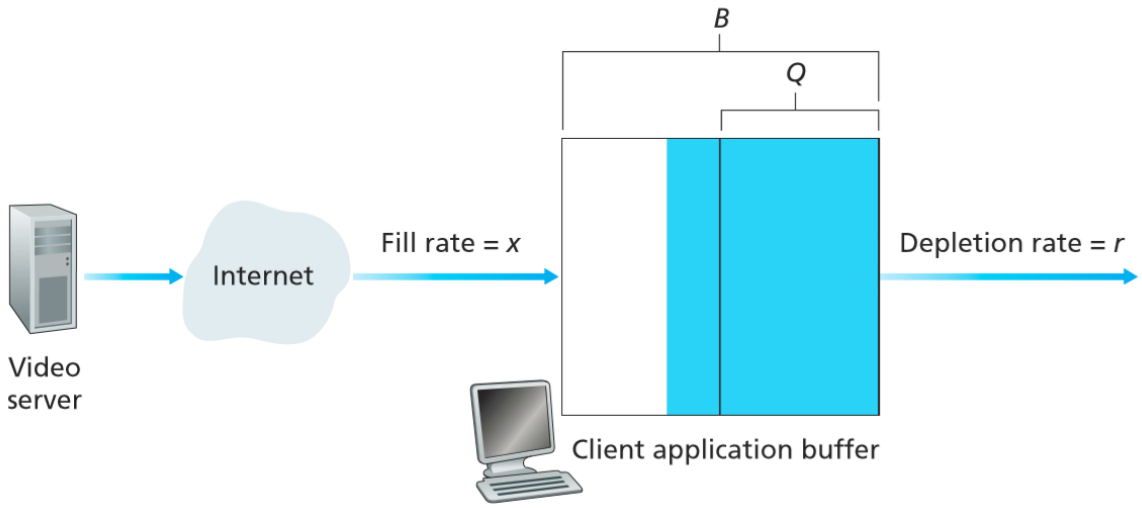
Found at : Content-Type: text/html; charset=ISO-8859-1*<cr><lf><cr><lf>*<!doctype… and Connection: Keep-Alive

**4.5** Explain how the "Last-Modified" and "ETag" information in the header lines can be useful.

Being able to know when the page was last modified allows users to test its integrity and validate if the resource is the same as the previously stored one.

"ETag" is the entity tag header, it identifies the version of a resource, allowing caches to be more efficient as the web server does not need to resend a full response when the content is unchanged.

**5.** Recall the simple model for HTTP streaming shown in the Figure below. Let B denote the size of the client's application buffer, and let Q denote the number of bits that must be buffered before the client application begins playout. Also, let r denote the video consumption rate. Assume that the server sends bits at a constant rate x whenever the client buffer is not full.




**5.1** Suppose that x < r. In this case playout will alternate between periods of continuous playout and periods of freezing. Determine the length of each continuous playout and freezing period as a function of Q, r, and x.

Fill at $x$ bps, deplete at $r$ bps

$Q$ = min. bits needed, $B$ = max. bits load

Continuous playout $= \frac{Q}{\text{net fill rate}}$

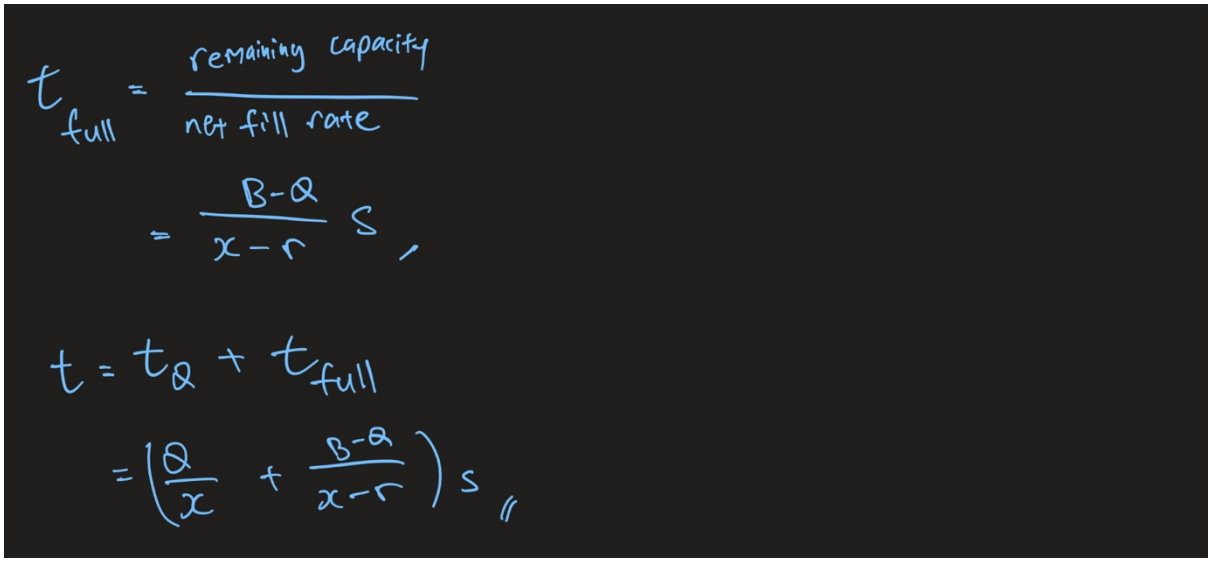
$= \frac{Q}{(r-x)\text{ bps}} = \frac{Q}{r-x}$ s,

freezing = loading (must fill to at least $Q$)

$= \frac{Q}{\text{net fill rate}}$

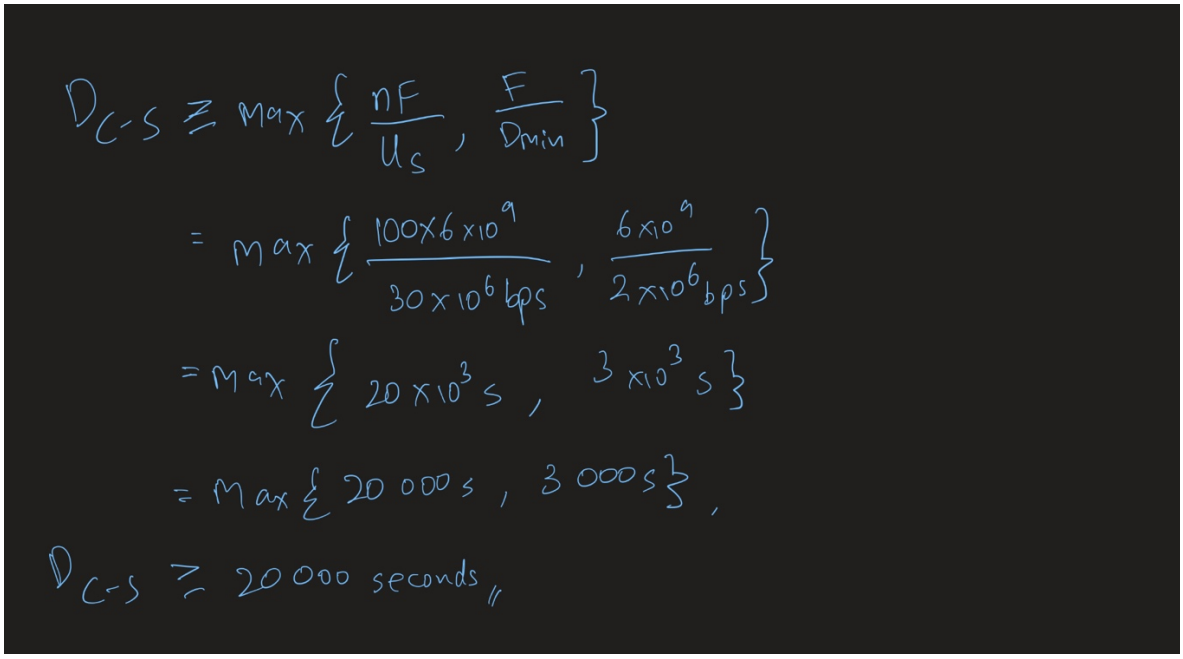
$= \frac{Q}{x}$ s //

**5.2** Now suppose that x > r. At what time does the client application buffer become full?

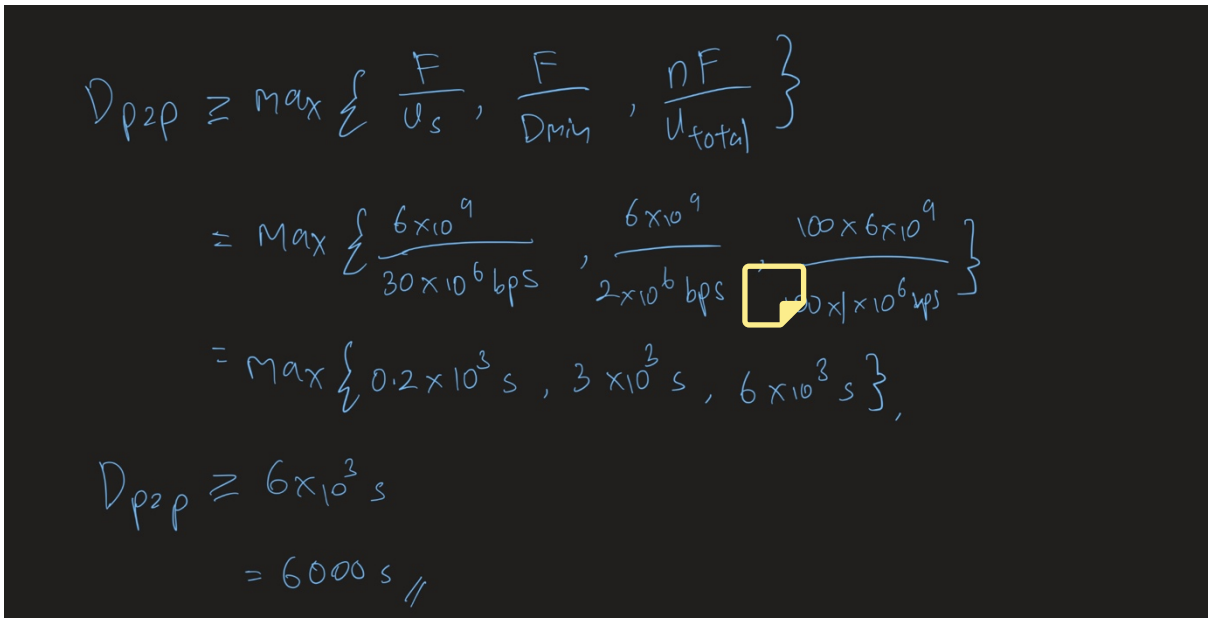


**6.** (2019's midterm exam question) Consider distributing a file of F = 6 x $10^9$ bits to N=100 peers. The server has an upload rate of $u_s$ = 30 Mbps, and each peer has a download rate of $d_i$ = 2 Mbps and an upload rate of $u_i$=1 Mbps. Assume 1M = $10^6$. Calculate the minimum distribution time (i.e., to let every peer have a copy of the file) for:

**6.1** the client-server distribution mode, and




**6.2** the P2P distribution mode.



**7.** Compare the basic design of the following application layer protocols: HTTP (you can focus on HTTP/1.1), MQTT, WebSocket, and SMTP.

HTTP: A client-server model, with a client opening a connection to make a request and waiting until it receives a response. Mainly for transmitting hypermedia documents like html, in other words, a unidirectional communication.

MQTT: Common in IoT devices, suitable for point-to-point communication and transmitting data using a MQTT broker. Designed for communication in remote locations with limited bandwidth, where both end points are able to communicate with each other. Hence it is a bi-directional model.

WebSocket: A client-server model, but it is a bi-directional communication, which is one of the differences from HTTP. The socket will always stay alive until either ends of the connection is closed. Breaks up large chunks of data packets into smaller ones for ease of transmission.

SMTP: Mail processing model, where the mail sent by a user will be transferred to other forwarders along the way (e.g Mail server agents, mail exchanger, mail delivery agent) before reaching its destination.

In conclusion, all of these commonly use TCP protocols, with some differences in how communication is done and other data transfer protocols supported. HTTP also supports UDP whereas MQTT does not due to its ordering requirements.

**7.1** List one common design aspect that all these application protocols adopt. Briefly explain the reason behind why they all choose this design.

They generally support TCP as their transport layer due to it being more secure, ordered, reliable and error checked. These qualities will ensure the integrity and authenticity of the data transmitted to each end points.

**7.2** List a main design difference between HTTP and SMTP protocol, and briefly explain the reason behind this design difference.

HTTP uses a pull model whereas SMTP uses a push model. This is due to the needs of users; HTTP is for users to pull information off servers while SMTP is for users to push information in the form of a mail into the mail server for it to reach its destination.

**7.3** List a main design difference between MQTT and SMTP protocol, and briefly explain the reason behind.

MQTT requires a broker between two devices, thus it is a three-way handshake. However, SMTP requires many steps of pushing (to exchange servers, mail agents) as mail might be sent to someone out of the sender's regional network.

**7.4** List a main design difference between HTTP and WebSocket protocol, and briefly explain the reason behind.

HTTP uses a unidirectional model whereas WebSocket is a bi-directional implementation. HTTP is used in applications that have static data and only the client initiates the communication, and thus receives a response from the server. However, WebSocket is better for real time data and is able to communicate without a user request as the server needs to send data constantly.

**=== END ===**