Question 1

In this question, we consider some of the pros and cons of virtual-circuit and datagram networks.

a. Suppose that routers were subjected to conditions that might cause them to fail fairly often. Would this argue in favor of a VC or datagram architecture? Why?

A **datagram architecture** is preferred because it is **more resilient to failures**. It does not rely on fixed paths, so packets can be **rerouted dynamically** if a router fails. In contrast, VC networks depend on pre-established paths, and a failure would **break the virtual circuit**, requiring re-establishment.

b. Suppose that a source node and a destination require that a fixed amount of capacity always be available at all routers on the path between the source and destination node, for the exclusive use of traffic flowing between this source and destination node. Would this argue in favor of a VC or datagram architecture? Why?

A **virtual-circuit (VC) architecture** is preferred because it allows **resource reservation** during connection setup. This guarantees a **fixed bandwidth/capacity** along the path. Datagram networks cannot provide such guarantees since packets are routed independently without reservation.

c. Suppose that the links and routers in the network never fail and that routing paths used between all source/destination pairs remains constant. In this scenario, does a VC or datagram architecture have more control traffic overhead? Why?

A **datagram architecture** has **more control overhead** because it requires **continuous routing updates and per-packet routing decisions**. In contrast, VC networks have control overhead mainly during **connection setup**, after which data follows a fixed path with minimal additional control.

Question 2

UDP and TCP use 1s complement for their checksums. Suppose you have the following three 8-bit bytes: 01010011, 01100110, and 01110100.

a. [6] What is the 1s complement of the sum of these 8-bit bytes? (Note that although UDP and TCP use 16-bit words in computing the checksum, for this problem you are being asked to consider 8-bit sums.) Show all work.
b. [2] Why is it that UDP takes the 1's complement of the sum; that is, why not just use the sum?
c. [2] With the 1s complement scheme, how does the receiver detect errors?
d. [2] Is it possible that a 1-bit error will go undetected?
e. [2] How about a 2-bit error?

Solution:

a. Note, wrap around if overflow.

$$\begin{array}{r} 0\ 1\ 0\ 1\ 0\ 0\ 1\ 1 \\ +\ 0\ 1\ 1\ 0\ 0\ 1\ 1\ 0 \\ \hline 1\ 0\ 1\ 1\ 1\ 0\ 0\ 1 \end{array} \text{ and then } \begin{array}{r} 1\ 0\ 1\ 1\ 1\ 0\ 0\ 1 \\ +\ 0\ 1\ 1\ 1\ 0\ 1\ 0\ 0 \\ \hline 0\ 0\ 1\ 0\ 1\ 1\ 1\ 0 \end{array}$$

One's complement = 1 1 0 1 0 0 0 1.

b. To detect errors, the receiver adds the four words (the three original words and the checksum).

c. If the sum contains a zero, the receiver knows there has been an error.

d. All one-bit errors will be detected.

e. Two-bit errors can be undetected (e.g., if the last digit of the first word is converted to a 0 and the last digit of the second word is converted to a 1).

Question 3

Suppose the network layer provides the following service. The network layer in the source host accepts a segment of maximum size 1,200 bytes and a destination host address from the transport layer. The network layer then guarantees to deliver the segment to the transport layer at the destination host. Suppose many network application processes can be running at the destination host.

a) Design the simplest possible transport-layer protocol that will get application data to the desired process at the destination host. Assume the operating system in the destination host has assigned a 4-byte port number to each running application process.

The simplest transport-layer protocol performs only **multiplexing and demultiplexing**. It adds a **4-byte destination port number** to each segment header to identify the target application. Since the maximum segment size is 1,200 bytes, the protocol can carry **1,196 bytes of data**. At the receiver, the destination port is used to deliver the segment to the correct application process.

b) Modify this protocol so that it provides a "return address" to the destination process.

The protocol is modified by adding a **4-byte source port number** to the header. This acts as a **return address**, allowing the receiving application to identify the sender and send responses back to the correct process.

(c) In your protocols, does the transport layer "have to do anything" in the core of the computer network?

No. The transport layer **does not need to do anything in the core of the network**. All transport-layer functions (such as adding and reading port numbers for multiplexing/demultiplexing) are performed **only at the end hosts**, while the network core simply forwards packets.

Question 4

(a) (4 points) Consider distributing a file of F bits to N peers using a client-server architecture. Assume a fluid model where the server can simultaneously transmit to multiple peers, transmitting to each peer at different rates, as long as the combined rate does not exceed u_s.
a. Suppose that $u_s/N \leq d_{min}$. Specify a distribution scheme that has a distribution time of NF/u_s.
b. Suppose that $u_s/N \geq d_{min}$. Specify a distribution scheme that has a distribution time of F/d_{min}.
c. Conclude that the minimum distribution time is in general given by $\max\{NF/u_s, F/d_{min}\}$.

(a) If $u_s/N \leq d_{min}$, the server sends the file to all N clients in parallel at rate u_s/N each. Since each client can download at least this rate, all clients receive the full file in:

$$D_{CS} = \frac{F}{u_s/N} = \frac{NF}{u_s}$$

(b) If $u_s/N \geq d_{min}$, the server sends to each client at rate d_{min}. Since the total rate $Nd_{min} \leq u_s$, the server can support all clients at that rate. The distribution time is:

$$D_{CS} = \frac{F}{d_{min}}$$

(c) Therefore, the minimum client-server distribution time is controlled by the larger bottleneck:

$$D_{CS} = \max\left\{\frac{NF}{u_s}, \frac{F}{d_{min}}\right\}$$

Question 5

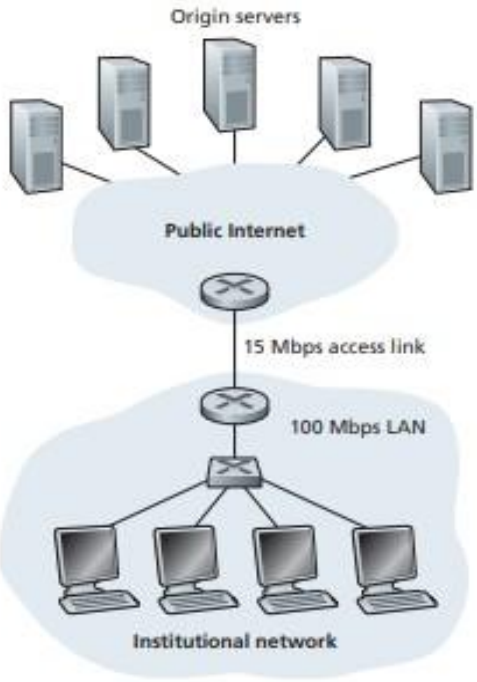

Figure 1: TCP window size as a funtion of time

1. Consider the Figure in the above, for which there is an institutional network connected to the Internet. Suppose that the average object size is 850,000 bits and that the average request rate from the institutions browsers to the origin servers is 16 requests per second. Also suppose that the amount of time it takes from when the router on the Internet side of the access link forwards an HTTP request until it receives the response is 3 seconds on average. Model the total average response time as the sum of the average access delay (that is, the delay from Internet router to institution router) and the average Internet delay. For the average access

$$\text{Access delay} = \frac{\Delta}{1 - \Delta\beta}$$

(a) Without Cache

$$\Delta = \frac{850000}{15000000} = 0.0567$$

$$\Delta\beta = 0.0567 \times 16 = 0.907$$

$$\text{Access delay} = \frac{0.0567}{1 - 0.907} = 0.61s$$

$$\text{Total response time} = 3 + 0.61$$

$$\boxed{3.61 \text{ seconds}}$$

(b) With Cache, Miss Rate = 0.4

Only **40%** of requests go through access link:

$$\beta = 0.4 \times 16 = 6.4$$

$$\Delta\beta = 0.0567 \times 6.4 = 0.363$$

$$\text{Access delay} = \frac{0.0567}{1 - 0.363} = 0.089s$$

For missed requests:

$$3 + 0.089 = 3.089s$$

Average response time:

$$0.4 \times 3.089 = 1.24s$$

$$\boxed{1.24 \text{ seconds}}$$

Question 6

Suppose Alice, with a Web-based e-mail account (such as Hotmail or gmail), sends a message to Bob, who accesses his mail from his mail server using POP3. Discuss how the message gets from Alice's host to Bob's host. Be sure to list the series of application-layer protocols that are used to move the message between the two hosts.

The message is first sent from Alice's ***host to her mail server over HTTP***. Alice's mail server then sends the message to Bob's mail server over SMTP. Bob then transfers the message from his mail server to his host over POP3. **OR**

The email is first sent from Alice's host to her webmail server using **HTTP/HTTPS**. The webmail server then forwards the message to Bob's mail server using **SMTP**. Finally, Bob downloads the message from his mail server to his host using **POP3**.

Question 7

(a) (2.5 points, 0.5 point for each question) 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>*

a. What is the URL of the document requested by the browser?
b. What version of HTTP is the browser running?
c. Does the browser request a non-persistent or a persistent connection?
d. What is the IP address of the host on which the browser is running?
e. What type of browser initiates this message? Why is the browser type needed in an HTTP request message?

a. The document request was http://gaia.cs.umass.edu/cs453/index.html. The Host : field indicates the server's name and /cs453/index.html indicates the file name.

b. The browser is running HTTP version 1.1, as indicated just before the first <cr><lf> pair.

c. The browser is requesting a persistent connection, as indicated by the Connection: keep-alive.

d. This is a trick question. This information is not contained in an HTTP message anywhere. So there is no way to tell this from looking at the exchange of HTTP messages alone. One would need information from the IP datagrams (that carried the TCP segment that carried the HTTP GET request) to answer this question.

e. Mozilla/5.0. The browser type information is needed by the server to send different versions of the same object to different types of browsers.