Assignment #2

Part 1: Short Answer Questions

1) A TCP entity opens a connection and uses slow start. Approximately how many round-trip times are required before TCP can send N segments?

TCP initializes the congestion window to 1, sends an initial segment, and waits. When the ACK arrives, it increases the congestion window to 2, sends 2 segments, and waits. When the 2 ACKs arrives, they each increase the congestion window by one, so that it can send 4 segments. In general, it takes $\log_2 N$ round trips before TCP can send N segments.

2) Consider the effect of using slow start on a line with 10 msec RTT and no congestion. The receive window is 24 KB and maximum segment size is 2 KB. How long does it take before the first full window can be sent?

The first bursts contain 2K, 4K, 8K, and 16K bytes, respectively. The next one is 24 KB and occurs after 40 msec.

3) Although slow start with congestion avoidance is an effective technique for coping with congestion, it can result in long recovery times in high-speed networks, as this problem demonstrates.

   a. Assume a round-trip time of 60ms (about what might occur across a continent) and a link with an available bandwidth of 1Gbps and a segment size of 576 octets. Determine the window size needed to keep the pipe full and the time it will take to reach that window size after a timeout using Jacobson's algorithm.

$W = (10^9 \times 0.06) / (576 \times 8) \approx 13{,}000$ segments
If the window size grows linearly from 1, it will take about 13,000 round trips, or about 13 minutes to get the correct window size.

   b. Repeat (a) for a segment size of 16Kbytes.

$W = (10^9 \times 0.06) / (16{,}000 \times 8) \approx 460$ segments
In this case, it takes about 460 round trips, which is less than 30 seconds.

4) In a leaky bucket scheme, what should be the capacity of the bucket if the output rate is 5 gal/min, and there is an input burst of 80 gal/min for 12 sec and there is no input for 45 sec?

Input: $(80/60) \times 12 + 0 \times 45 =$ **16** gallons
Output: **5** gallons
Left in the bucket: $16 - 5 =$ **11**

5) In a leaky bucket scheme, let the data rate on the network is 2Mbytes/sec, and the data rate on the link from host to the bucket is 2.5Mbytes/s. Assuming that a host has 250 Mbytes to send in a burst, what should be the minimum capacity of the bucket in order that there is no loss of data?

Time for host to transmit data = total bits / max trans rate = 250 Mbytes / 2.5 Mbyte = 100 sec

Actual data sent on network in 100 s = network rate * 100s = 2 Mbyte /s * 100s = 200 Mbyte

Minimum bucket size = 250 Mbyte -200 Mbyte = 50 MByte

6) Consider the case where packets arrive as a Poisson process to a router's queue at the rate of 300 packets/sec. The time to service a packet is exponentially distributed. Suppose that the mean packet length is 500 bytes and the link capacity is 1.5 Mbps.

a. Find the router utilization and the mean number of packets in the queue.

Packet length is 500 bytes = 4000 bits, so the service rate is 1,500,000 bits/sec = 375 packets/sec. Therefore, the utilization is 300/375 = 0.8 The mean number of packets in the system is 0.8 / (1 – 0.8) = 0.8 / 0.2 = 4. Of these, we expect three to be in the queue, and one to be in service.

b. What is the probability that the link's queue has one, two, and ten packets, respectively?

When the link queue has one packet because one packet is being served. So, we need to consider state probability for state = 2 (i.e. Pr_2) in this case.

Prob (queue has one packet) = $Pr_2 = (1- \rho) \rho^2 = 0.128$
Prob (queue has two packets) = $Pr_3 = (1- \rho) \rho^3 = 0.1$
Prob (queue has ten packets) = $Pr_{11} = (1- \rho) \rho^{11} = 0.0067$

Part 2: Client/server Programming

This exercise requires you to write a client/server program using Java/Python/C/C++ socket package. The client prompts the user for an operation character and, depending on it, some input data may be needed as well. The client sends a command string to the server, and then displays the server's response. The server receives a command string from the client, executes the command, and sends back the response to the client.

The format of the command string sent to the server is as follows:

<commandType> <studentNumber> [parameters]

where
• <commandType> is the string **typeI** with **I** being the command character as specified in the table below.
• <studentNumber> is **your** numerical uOttawa id.
• [parameters] is an optional string argument which may be needed depending on the command type.

The following table describes the command types and the action the server is expected to perform.

Command Character	Meaning	Parameters	Server Action
	Connection Made	--------	When the connection is set up with the server the server should reply with the message "GREETINGS"
A	Knowing my IP address	--------	The server replies with the IP address of the client
B	Receiving back the sentence capitalized	A lowercase sentence made of any characters (although it doesn't matter if you include number or special characters)	The server replies with the sentence capitalized and it keeps the log of this command but discard the message content.
C	Send Message	A string of characters	Receives the message and stores it in memory (a new message is appended to the existing one). Then it sends "ACKN" to the

			client.
D	Breaking Connection	--------	Reply "Exit"

Requirements:
1. Java/Python/C/C++ socket must be used.
2. Name the client and server programs myClient and myServer, respectively.
3. The server ignores all operation commands not matching the above ones.
4. The client program exits/disconnects only after requesting the breaking connection operation and receiving the reply from the server.
5. The server must keep running after handling a client, i.e. the server must not exit after performing the 'exit' operation. Instead the server waits for another client.
6. Do **NOT** hard code the server address and port number in your source code. You should set them as command line arguments. (This enables your program to run on other machines and ports)
7. The server should **allow** a connection request from "localhost". This means that the client and the server can run on the same machines.
8. Building a nice GUI for the client is not necessary. Using a command line for the user input is good enough.
9. Your server program should only perform the actions mentioned above when the uOttawa id coming from client is exactly your uOttawa id. When the client sends a id different to your uOttawa id, either response a error message or simply not response anything. You may lose marks if your server fails to identify your id.

An example of user interaction is provided below. The ">" symbol identifies the command prompt.

C:\>java myClient 127.0.0.1 9999
Session has been established.
Server: GREETINGS
> typeA 5432190
Server: 10.1.20.19
> typeB 7345071 abcd
Server: ABCD (The server should save "typeB 7345071" in memory or print it to console)
> typeC 1234567 JUNIPER
Server: ACKN (The server should also save "JUNIPER" in memory or print it to console)
> typeD 1234567
Server: EXIT
Session is terminated.

What to submit:
1. The source files (Java/Python/C/C++) for the server and client programs.
2. A copy of the "Instructions for Users" **clearly** describing how to compile (on which platform i.e. Windows/MacOS/Linux) and run your programs. If your program will not handle certain situations, or should not be used in a certain way, also mention it in this file.