Exploring the Transport Control Protocol

In this exercise, you will explore the behavior of TCP to gain an understanding of the concept of a TCP connection and message buffering as it applies to TCP data streams

Exercise 1: Understand Connections

1. Run 2 copies of the Networking workbench on your computer:
   i. On 1 copy, set the program to client mode by selecting TCP→ TCP Active Open from the menu
   ii. On the other copy, set the program to server mode by selecting TCP→ TCP Passive Open from the menu
2. Create sockets at both the client (active) and server (passive) ends.
3. At the Active end, click on Connect.

   What happens and why?

   An error message appears stating the socket cannot be connected because it is non-blocking.

4. Click on bind at the Passive end.
5. At the Active end, try to Connect again

   What happens this time and why?

   An error message appears stating the socket cannot be connected because it is non-blocking.

6. Click on Listen then Accept at the Passive end.
7. At the Active end, try to Connect again

   What happens and why?

   The connection to the socket of the Passive end succeeds.

8. Close connections and sockets at both ends.
9. Repeat steps 2 and 4, and then, listen at the passive end.
10. At the Active end, try to Connect

    What happens and why?

    The connection to the socket of the Passive end succeeds.

11. Accept at the Passive end.

    What happens and why?

    The Connect socket is set to Non Blocking

    Given the results of the previous steps, what are the roles of the Listen and Accept primitives of TCP sockets?

    The roles of the Listen and Accept primitives of TCP sockets is to establish a connection between to sockets and ensure that both sockets are connected.

    Why do you think might it be important to separate these 2 steps of connection establishment?

The importance of separating these 2 steps of connection establishment is to ensure a reliable connection between the two sockets.

Exercise 2: Understand Stream Behavior and Communication

1. Set the message to be sent from the Active end as “Hello” then send a packet to the Passive End
2. On Passive end, click on enable receiver.

   In this step, was the message delivered exactly as it was sent?

   The message was delivered and the Active end said it sent 5 bytes of “Hello.”

3. Using the Passive end, also set its message to be sent to “Hello”, then send it twice to the Active end.
4. On the Active end, click on Enable Receiver then observe what the message that was received.

   What happens (what exactly is delivered to the application)?

   HelloHello

   Recall your experiments on UDP communications before. How does it differ when calling the receive function after multiple packets have been sent?

   How it differs when calling the receive function after multiple packets have been sent is that once the connection has been established, it automatically receives the data.

   What does this tell you about the nature of TCP as stream-oriented communication protocol (2pts)?

   The nature of TCP as a stream-oriented communication protocol is that data is sent as a continuous streams of bytes, rather than as discrete messages.

5. Close the connections and sockets on both copies of the workbench

   Based on the tests done in the previous steps, fill in the in the table below with the correct TCP primitive sequence leading to successful receipt of data on both client and server. Add more rows to the table as necessary:

Step #	Client	Server
1	Bind	
2		Bind
3		Listen
4		Accept
5	Connect	
6	Send	
7		Recv
8		Send

9	Recv	
10	Close	Close