SWINBURNE UNIVERSITY OF TECHNOLOGY

TNE20003 – Internet and Cybersecurity for Engineering Applications

Portfolio Task – Lab 8 Credit Task

Aims:

- To develop a Python implementation of a simple protocol using TCP. Students will rewrite their UDP solution to TCP. As TCP is a stream-based protocol, students will be required to modify the program to allow both client and server to locate the beginning and end of each message in the Protocol

Preparation:

- View "Internet Enabled Programming" & "Network Protocol Implementation"

Due Date:

- Your Task will be assessed via an online quiz. You must score the required minimum to pass the test. You will be allowed a number of attempts to pass the test at the grade level you attempt. You are encouraged to complete the test at the end of the lab but if you do not, you must complete it before your next lab class.

Protocol Definition

The Protocol is a very simple protocol. All messages are constructed using ASCII text and each message is self-contained within a single TCP stream.

The Client has only one message type it can send to the server, the message is an ASCII string formatted as:

TNE20003:<message>

In this message:

TNE20003: forms the protocol header and must match this value exactly

<message> is an ASCII string containing the data to send to the server. This string is **NOT** terminated in any way and JUST contains the actual message. ***<message>*** must be at least one character long.

The Server is responsible for responding to the TCP stream received on the open port. The Server **MUST** check the payload of the TCP packet to ensure it matches the valid Client Message as described above.

If the received packet is properly formed (contains a properly formed header and a valid ***<message>***) then the server must respond with a formatted message containing

TNE20003:A:<message>

In this response:

TNE20003: forms the protocol header and must match this value exactly

A: designates the message type as being an Acknowledgement of receipt of the ***<message>***

<message> is an ASCII string and must match exactly the message sent by the client. You will be required to extract ***<message>*** from the received packet to place into the reply

If the receive packet is **NOT** properly formed, then the server must respond with the message:

TNE20003:E:<error_message>

In this response:

TNE20003: forms the protocol header and must match this value exactly

E: designates the message type as being an Error message

<error_message> is an ASCII string. You can choose the exact string but this is an error to report back to the user at the client

Methodology:

You will be developing a new set of programs implementing the relevant requirements of the protocol to function with TCP. It would be strongly advised to use the threaded echo server code from the tutorial as a base for building your TCP server.

The current implementation of the Protocol does not require delineation of the start and end of messages. In the Pass task of this lab the Protocol is UDP based, and all messages are delivered as distinct chunks. When you a message is sent, the remote end (client or server) is guaranteed to receive the entire message in a single read. They are also guaranteed that two messages will not be merged in a single read. This means that the program is simple in that you know you are parsing an entire message for validity and to construct a response.

TCP is more complex as it is a Stream-based Protocol. This means that all messages run into each other and a read from a remote system can contain multiple messages, part of a message, or the end part of one message and the start of a second message. Your task will require modifying the protocol to allow the receiving program to find the start and end of each message from the TCP stream. This means maintaining some persistence of the message between reads. There are multiple valid mechanisms for modifying the protocol, you are free to choose any mechanism you like to do so.

Task:

You will need to develop two Python programs to implement the TCP version of the protocol. The functionality otherwise needs to be exactly the same in terms of parsing and responding to packet contents

Assessment:

As a Credit task, not completing this task will result in the maximum achievable base grade for your Portfolio being restricted.

To pass this task, you must demonstrate the functioning program to your Lab Supervisor. If you do not get time to demonstrate you must upload your program/code to Canvas. This task will be marked as completed upon successful demonstration or a working upload, and the attainment of the score required to pass the online quiz. You will be allowed a number of attempts to pass the test at the grade level you attempt. You are encouraged to complete the test at the end of the lab but if you do not, you must complete it before your next lab class.