TNE20003 – Internet and Cybersecurity for Engineering Applications

Portfolio Task – Lab 8 Pass Task

Aims:

- To develop a Python implementation of a simple protocol using the transport layer protocol UDP. Students will build a simple UDP server and UDP client to implement the specified 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.

Important Notes:

In this task, we will be using UDP as the packetised nature of UDP means that you do not have to worry about finding the start and end of messages in the protocol, the entire message will be encoded in a single UDP packet.

Your UDP server will run a permanent loop accepting packets on an open UDP port. Each packet should contain a message in the allowed format according to the protocol specified below. You will be required to:

- Parse the message to determine if it meets the protocol definition
- If the message does not match the protocol definition, reply with the error message as per the protocol
- If the message does meet the protocol definition, reply with a properly formatted acknowledgment response as per the protocol definition

Your UDP client will be required to connect to the UDP server and send at least two messages using the protocol definition as described below. Each message must be sent in a unique datagram. You will be required to:

- Parse replies from the server
- Print server replies to the screen and extract packet contents

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 UDP datagram

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 UDP datagrams received on the open port. The Server **MUST** check the payload of the UDP 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:

Examine the Python UDP socket commands to get an understanding of how to receive and send messages

You may wish to examine the following options for Python programming to make your life easier

The struct library to pack and unpack messages. This can allow you to combine multiple variables into a single structure (to create a formatted message) and to unpack a message into multiple variables

Look at how you can segment python strings and/or byte objects using indexing (eg. var[:10], var[10:] and var[10:20])

Task:

You will need to develop two Python programs. The server program will need to:

- Open a UDP port
- Listen for all incoming UDP packets
- Parse the packet as per the protocol in this document
- Construct an appropriate response and send it back to the remote system

The client program will need to:

- Create a UDP socket
- Send at least two properly formatted messages to the server
- Listen for responses
- Parse the reply, extracting protocol parameters and display to the screen

Assessment:

Not completing this task will result in you failing the Unit

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.