Parallel and Distributed Computing: Homework 2

Objectives : Use an isolated computer network to expose the layered OSI model of communications. The student is challenged to connect two virtual computers in a single machine, to follow the packets of two communicating processes and identify the OSI layers used in the communications mechanism. Using the C programming language, and a network 'sniffer," two processes will be defined and communicate in a "client-server" configuration to identify the layers packets traverse during the communications. Bit patterns will be identified to answer questions related to the exercise.

"A protocol stack (such as OSI), today is usually provided by the operating system (rather than as a separate library, for instance), it is a set of programs that allow processes to communicate over a network using the protocols that the stack implements.

The application programming interface (API) that programs use to communicate with the protocol stack, using network sockets, is called a socket API. Development of application programs that utilize this API is called socket programming or network programming."

Assignment

Install Virtualbox in your computer if it is not already installed. Setup two new Virtual Machines (VMs): SenderVM and ReceiverVM; with some lightweight Linux distribution such as Xubuntu in each of them. Little RAM and one core per VM will be enough. Configure their shared virtual network so that both VMs share the same network segment. Here there is a good tutorial on YouTube.

Install Wireshark in ReceiverVM. Please note that the use of Wireshark is forbiden in all UCI networks. **Do not** use Wireshark or any other sniffing tool in any UCI network.

Create a C program `receiver.c` in ReceiverVM. Create a C program `sender.c` in SenderVM. The first part of your assignment is, using **sockets**, make SenderVM send a simple (perhaps even hard-coded) message such as "Hello Sockets" to ReceiverVM.

Capturing packets: On ReceiverVM, run the `receiver` program. Then, start capturing packages with Wireshark. In SenderVM, run the `sender` program to send the message. There are plenty of guides and tutorials on the web on how to accomplish this in Linux.

When you start capturing packets, make sure of selecting the appropiate interface. Services, browser, and the OS constantly send packets to the Internet and the local network, therefore, in order to capture only the packet of interest, you must apply a **capture filter** (not a display filter). Something like this:

"`ip src X and ip dst Y`"

Where `X` and `Y` are the IP addresses of SenderVM and ReceiverVM.

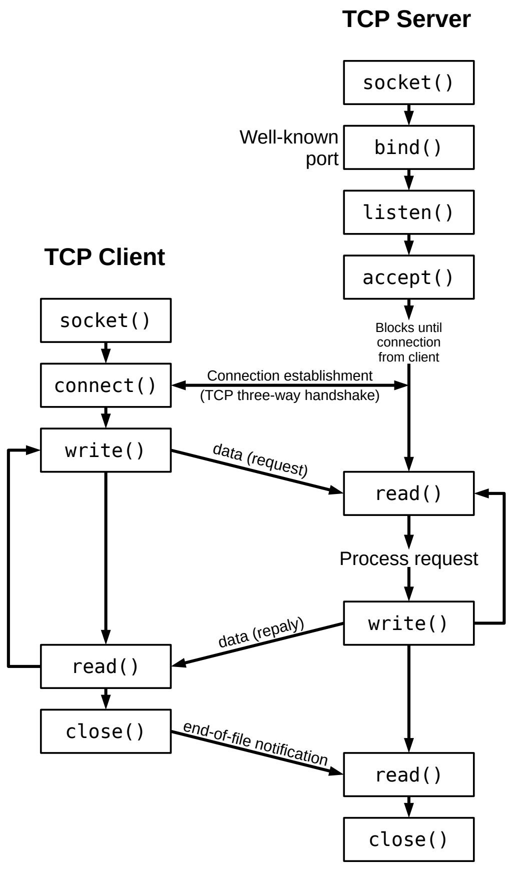
Report

Please write a two-page report.

First page: identify the single packet that contains the message sent by `SenderVM`. Based on this packet, **concisely** answer the following questions. Note that "A-B" denotes the inclusive range from A to B:

- What is encoded, and what is the purpose of bytes 0-5 and 6-11?

- What is encoded, and what is the relationship between, byte 14 and the two bytes 16,17?
- What is encoded, and what is the purpose of bytes 18-19?
- What is encoded, and what is the purpose of bytes 20-21?
- What is encoded, and what is the purpose of byte 23?
- What is encoded, and what is the purpose of bytes 26-29 and 30-33?
- What is encoded, and what is the purpose of bytes 34-35 and 36-37?
- What is encoded after byte 65?



After "Unix Network Programming"
by W. Richard Stevens

Please note that we are asking you to identify **what is** and to briefly explain **the PURPOSE** of each group of bytes.

Second page: Obtain a clear screenshot of the raw block of bits of the previously mentioned single packet that contains the message sent by `SenderVM` (raw blocks are located at the bottom part of Wireshark's window).

Split that block of bits, include it on the report, and associate each partition of bits to the layers 2 (Link), 3 (Network), 4 (Transport), and 7 (Application); by indicating the first and last byte of each layer as an inclusive range "A-B".

Submission

Submit **one** zip file named `hw2.zip`, containing **exactly 5 files**. Make sure the zip file does **not** include sub-directories (as Mac's default compression tool does) or extra files:

1. `sender.c`: The commented C source code of the sender program.
2. `receiver.c`: The commented C source code of the receiver program.
3. `packets.pcapng`: The captured packets in `pcapng` format. The file **must only** contain packets transmitted between the `SenderVM` and `ReceiverVM` VMs. This file cannot weight more than a few kilobytes. If that is the case, then you did not filter the capture and will obtain 0 points in this section.
4. `team.txt`: Text file where each line contains, separated by commas, the UCINetID, first name, and last name of each student, for example:
   `ptanteater, Peter, Anteater`