CHAPTER THREE

METHODOLOGY

3.1 Protocol Converter:

Protocol Converter is a software application that converts the standard or proprietary protocol of one device to the protocol suitable for another device to achieve interoperability. a common analogy is using a person to translate one language to another language so to people who don't speak the same language can communicate[20].

A segment containing three pc devices .on pc1 a java program will be running which transmits file over Ethernet link to pc2 then pc2 runs a C program that captures the Ethernet packets and converts them into serial format, pc3 is just to check whether software is running correctly or not .PC1 and PC2 are connected over a Local Area Network as shown in figure3.1 [20].

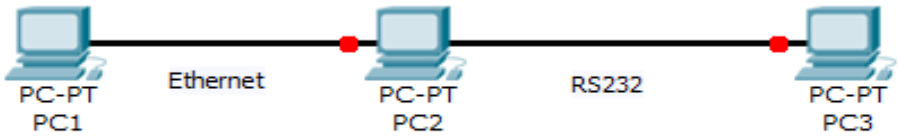

Figure3.1: Setup to test the protocol converter software.

There will be two types of packets: First type is:"File name packet" (F packets) that send the name of the file to be transmitted as shown in figure3.2.

Type of Packet (F)	Number of Packets	Length of File Name (FL)	File Name

Figure3.2: File name Packet for Ethernet.

Second type is:"Data packets"(D packets) that contain the actual contents of the file. The formats of F and D packets are shown respectively as shown in figure 3.3.

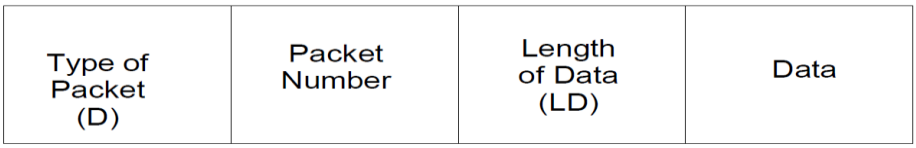

Figure3.3: Data Packet for Ethernet.

In serial communication we need two type of packets .First type:"File name packets"(F packets) as shown in figure3.4.

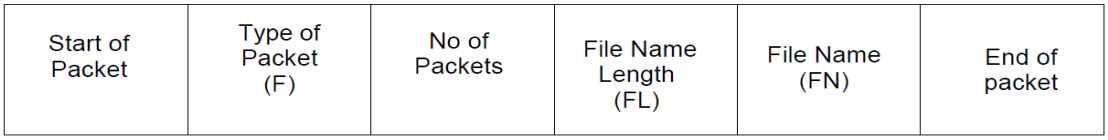

Figure3.4: File name Packet for serial communication.

Second type "Data packets"(D packets) also we need additional fields for "start of packets" and "end of packets" as shown in figure3.5.

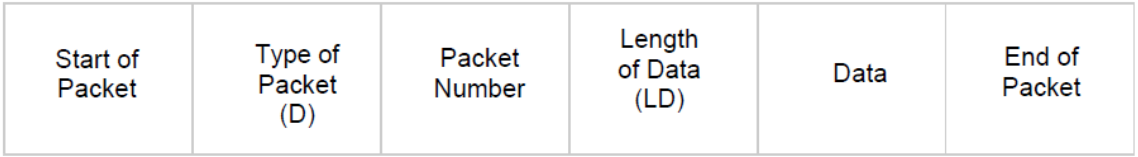

Figure3.5: Data Packet for serial Communication

3.2 Implementation:

The details of the programs that will be running on different machines are as follows:

*UDPFileTx.java will be running on PC1, which is a Windows machine.

*Protocol Converter will be running on PC2 which is a Linux machine.

*SerialRx.c will be running on PC3 which is a Linux machine.

3.3 Flow Chart:

3.3.1: Pseudo code 1

- Take the file name as an argument.
- Generated the file name packet (F) and data packet (D) for the file to be transmitted.
- Transmitted them over Ethernet LAN.
- Note that the destination machine IP address to be given .the address (192.168.2.1) corresponds to the Linux machines IP address.

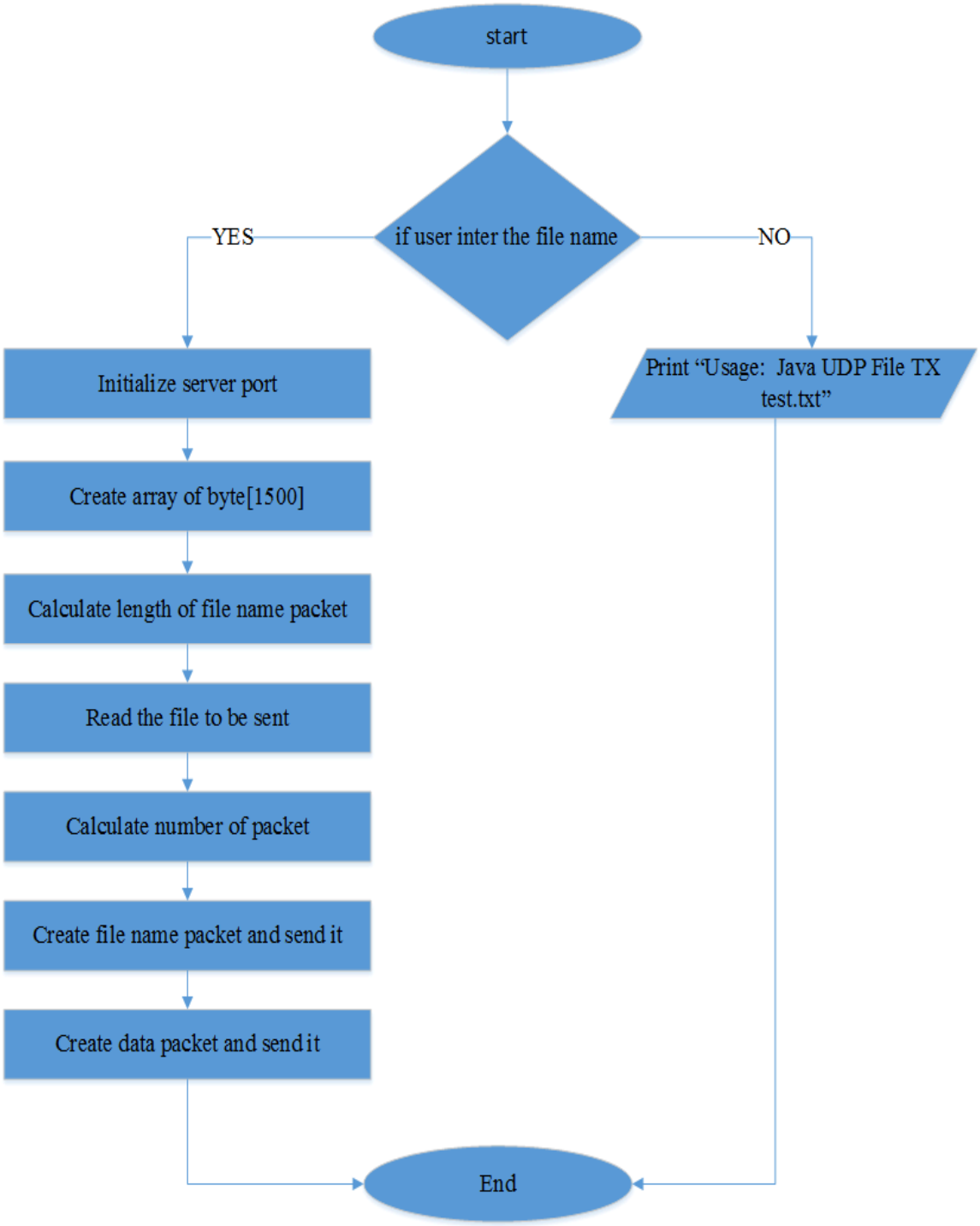

Figure3.6: Flow chart for pc1

3.3.2: Pseudo code 2

- Preprocessor directives and declarations.
- Opened the serial port .here needed to use the appropriate serial port device file based on the entry in directory.
- Configured the serial port parameters.
- Written data to serial port.
- Closed the serial port.
- Read Ethernet data to read the Ethernet frames .note that in Ethernet frame the actual user data started at 42byte. The earlier byte contained Ethernet header. To receive packet from data link layer created socket of type SOCK_PACKET.
- Convert the Ethernet packet into serial packet and to send them to serial port.
- Main function which calls functions to read the Ethernet data , open the serial port , configure the serial port parameters , convert the Ethernet data to serial packets and transmit to the serial port .

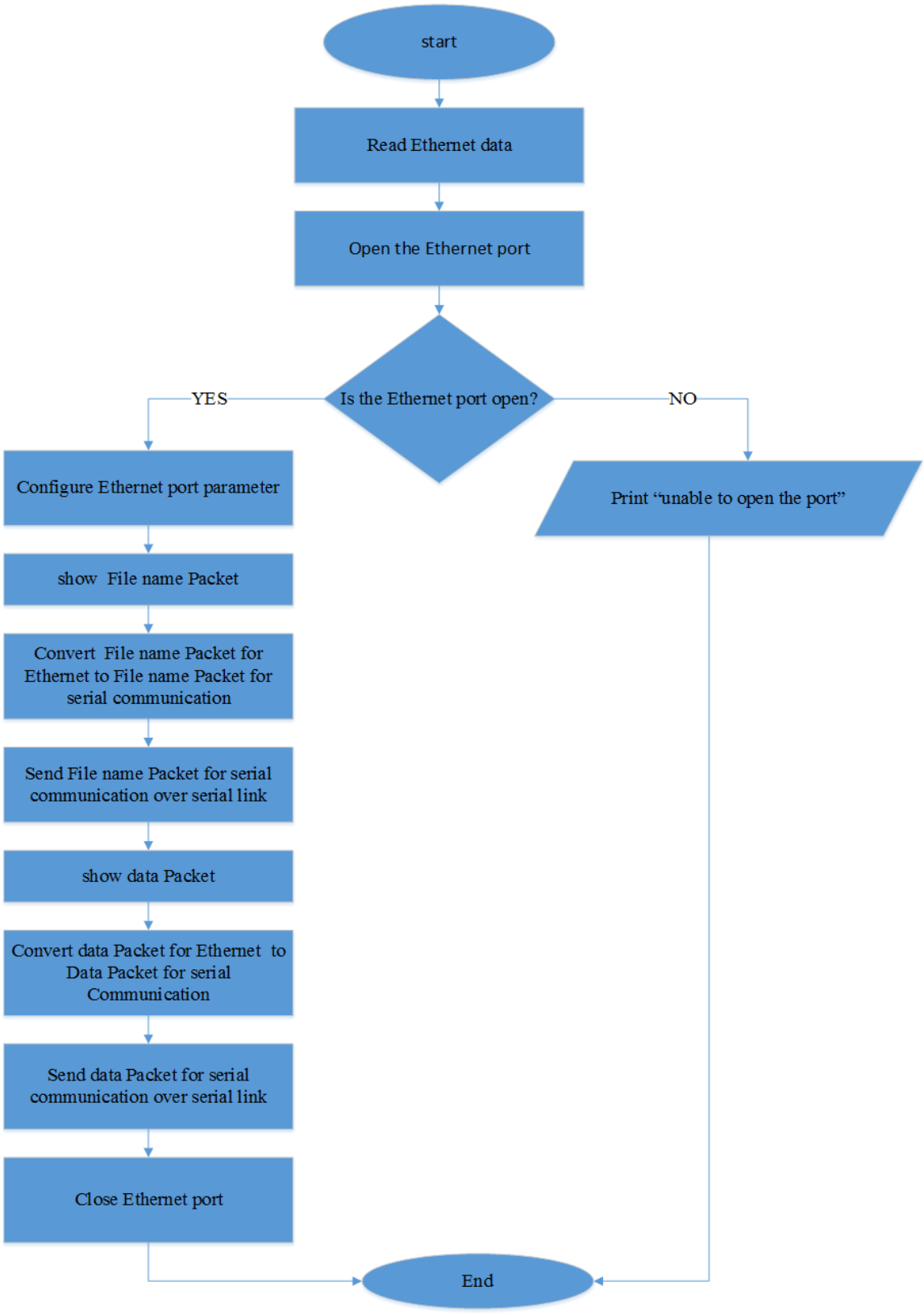

Figure3.7: Flow chart for pc2.

3.3.3: Pseudo code 3

- Preprocessor directives.
- Opened the serial port.
- Configured the port parameters.
- Read the data from serial port and receive the packet from the file.
- Closed the port.
- Main function calls function.

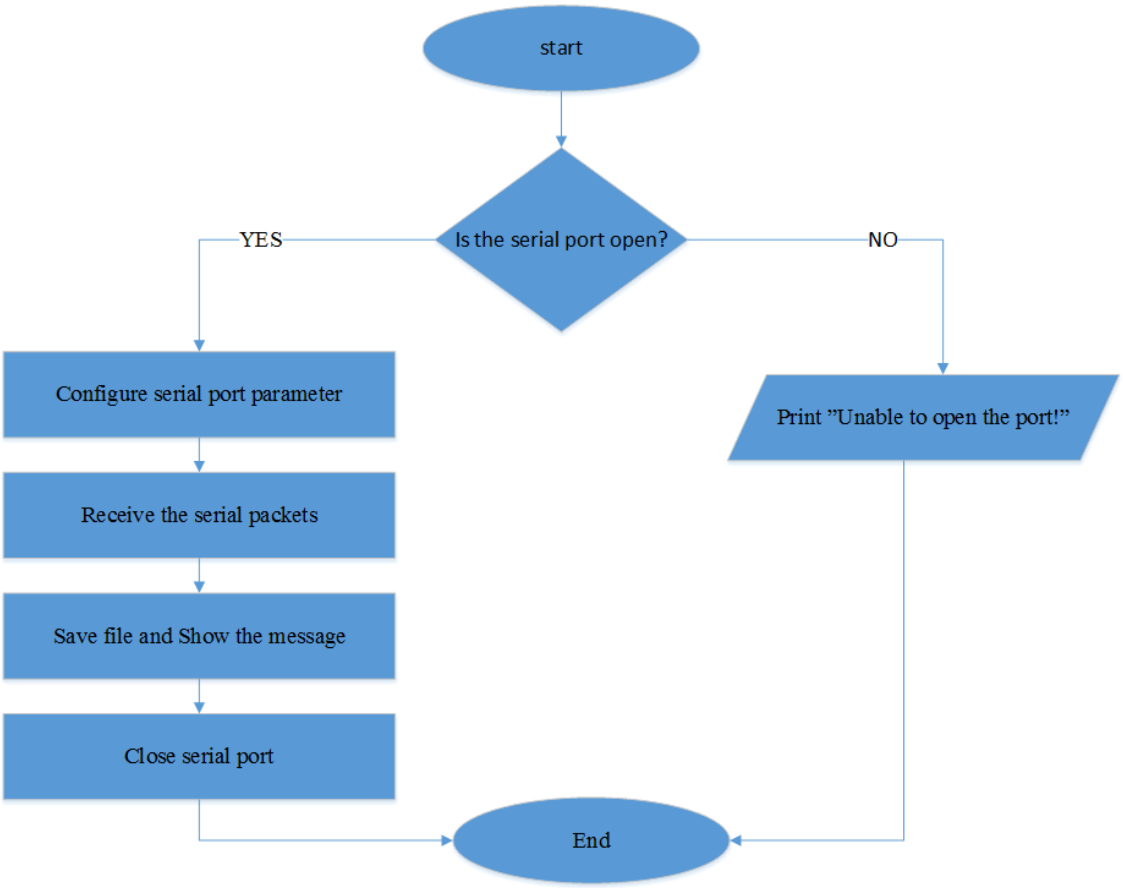

Figure3.8: Flow chart for pc3