Slovak Technical University
Faculty of informatics and information Technology
Ilkovičova 2, 842 16 Bratislava 4

Computer and Communication Networks

Communication using UDP datagram

Sultan Numyalai

exerciser: Ing. Rastislav Bencel
Study Field: INFO3
Year: 2. Bc
Academic Year: 2017/2018

adfa, p. 1, 2011.

Contents

To be continued…

POČÍTAČOVÉ A KOMUNIKAČNÉ SIETE

Komunikácia s využitím UDP protokolu

1 Zadanie úlohy

Nad protokolom UDP (User Datagram Protocol) transportnej vrstvy sieťového modelu TCP/IP navrhnite a implementujte program, ktorý umožní komunikáciu dvoch účastníkov v sieti Ethernet, teda prenos správ ľubovoľnej dĺžky medzi počítačm (uzlami). Program bude pozostávať z dvoch častí – vysielacej a prijímacej. Vysielací uzol pošle správu inému uzlu v sieti. Predpokladá sa, že v sieti dochádza k stratám dát.

Vysielajúca strana rozloží správu na menšie časti - fragmenty, ktoré samostatne pošle.Správa sa fragmentuje iba v prípade, ak je dlhšia ako max. veľkosť fragmentu. Veľkosť fragmentu musí mať používateľ možnosť nastaviť menšiu ako je max. prípustná pre linkovú vrstvu.

Po prijatí správy na cieľovom uzle tento správu zobrazí. Ak je správa poslaná ako postupnosť fragmentov, najprv tieto fragmenty spojí a zobrazí pôvodnú správu. Komunikátor musí vedieť usporiadať správy do správneho poradia, musí obsahovať kontrolu proti chybám pri komunikácii a znovuvyžiadanie rámca, vrátane pozitívneho/negatívneho potvrdenia. Pri nečinnosti komunikátor automaticky odošle paket pre udržanie spojenia. Odporúčame riešiť cez vlastne definované signalizačné správy.

2 **Analyza**

UDP stands for user datagram protocol, the client sends a datagram to the server, which then processes the information and returns a response, and UDP is a simple Transport-Layer protocol, the application writes a message to a UDP socket which is then encapsulated in a UDP datagram.

2.1 Encapsulation and headers

In networking Encapsulation is the process of taking data from one protocol and translating it into another protocol, so that the data can continue across the network.

When the data passes through TCP/IP protocol stack, the protocol at each layer adds or removes data to the basic layer, when a protocol adds data to the packet header, the process is called data encapsulation. When the data arrives at the transport layer, the protocol starts the process of data encapsulation. The end results depend on whether the data has been handled by TCP or UDP.

2.2 Error Control and re-requesting the packet

Under UDP datagram, there is no guarantee of receiving, ordering or duplication protection . We Use Cyclic redundancy Check (CRC) for fragment controlling.

CRC : is an error detecting node among the best check sums available to detect or correct errors in communication transmission. CRC is commonly used in digital networks to detect accidental changes to the raw data.

3 Specifications of the requirements

An UDP header consists of Source Port, Destination Port, UDP Length and UDP checksum.

For each message the user has to give as standard input the source address and source port. for each node we can show message and file address. Once the date is sent, it is divided into fragments according to the given size of fragment, fragment size will be controlled so that we can't give negative, 0 or an unreal size.

The connection of the sending and receiving message is maintained by the keepalive which shows out whether the connection between the devices is still operating or is broken.

4 Solution Design (Navrh Riesenia)

The program will be written in c/c++ programming language in Visual Studio Development environment.

4.1 **Design of the Protocol**

At first we have to give Destination IP and port through standard input, it converts the data to bit array, and divides the data into fragments according to the given fragment length. Each fragment will be then given a number an a checkcum, Cyclic redundancy check will check the fragment according to the checksum so that all the fragments are without any error.

The client sends a datagram to the server using the sendto function and returns the address of the destination as a parameter the same way the server does not establish a connection with the client instead it call the recvfrom function and waits until the data arrives from the client. Recieve returns the IP address of the client with the datagram so that the server can answer to the client

The receive function will receive the data fragmentedly, and the received fragment is controlled whether it has received correct fragment number and whether the CRC check is correct . then it sends the fragmented data to an array and at the end writes the message that the data is received.

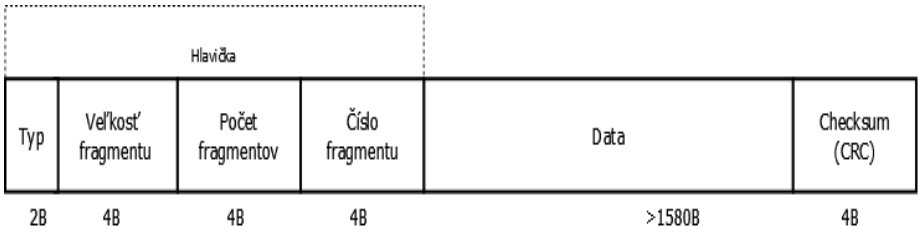
- Create a socket using the `socket()` function;
- Send and receive data from the `recieve()` and `send()` functions.
- Make a socket with the Socket function bind
- Bind the data
- Send and receive data

4.2

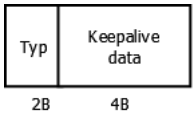
The KeepAlive packet checks whether there is a connection between the devices or is broken, in case the connection is broken the program writes error…

4.3 **Packet**

The program Is able to receive and send two different types of packets, one for the message and the second (KeepAlive) packet. First 2 bytes of each of the packets determine the type of the packet.



Header of the packet has 14B size, maximum data size is 1600B and the last 4B determines the check sum of the packet

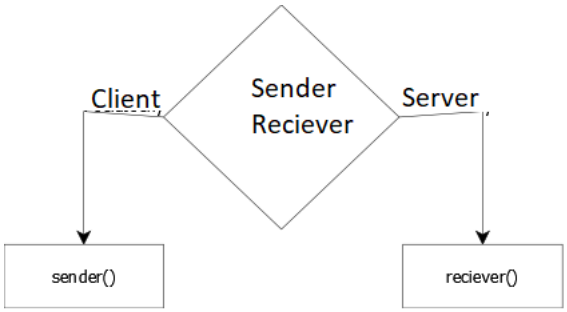


Keep alive packet is 6B big, the first 2 bytes are for the type of the packet and the rest 4B shows the data to be controlled.

4.4 User Interface

The program at the start show all the information about the functions data input the way the function works and at the end of each command writes which command was pressed and the program operated.

4.5 Program Diagram



Main function at start sets wether the program has to work as a server or client.

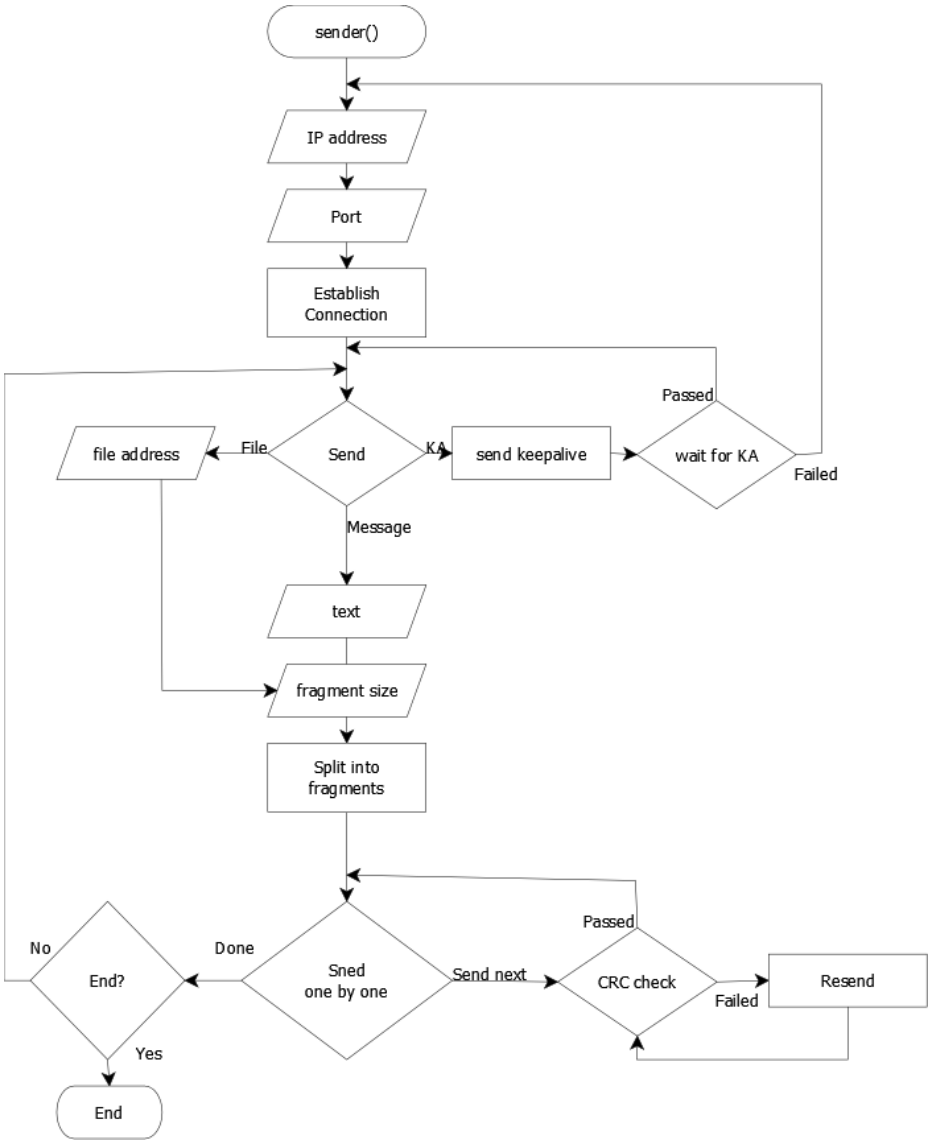


Image 4. Sender

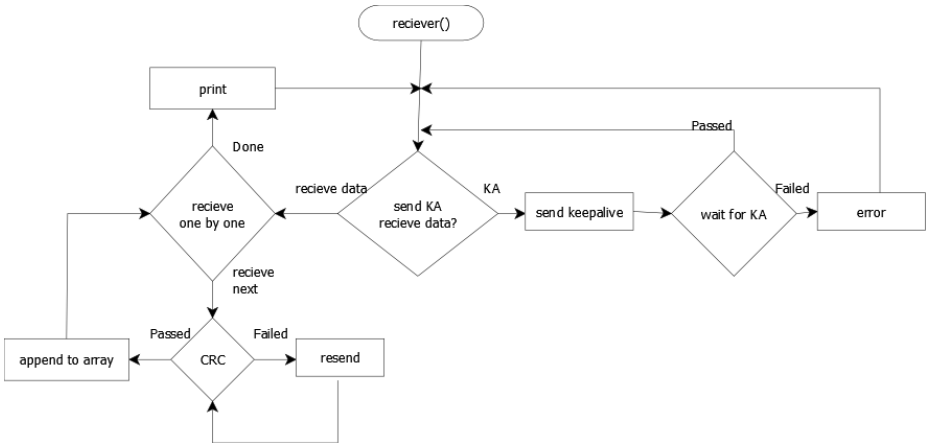


image 5. Receive()

//Source Wikipedia, Datmout University