Embedded Linux Multimedia Communication Based on TCP Protocol

Min Shi, Qifeng Zhou *

College of Information Science and Technology, Jinan University, Guangzhou, 510632, China.

*Corresponding author e-mail: 1093372288@qq.com

Abstract

With the advent of the intelligent era, various AI products appear in people's daily life and work, and embedded network communication has provided great technical support for the interconnection of all things. Based on the embedded Linux system, the multimedia network communication application can be realized under the TCP protocol through the socket network interface. The user and the server can communicate in real time, and the user and the user can communicate with each other. This article installs vmware+ubuntu as the upper computer in the PC, in s3c6410 Embedding the Linux system as the target machine, developing the server system on the host computer, developing the client system on the target board, thereby realizing the communication between the client and the server, and the client indirectly communicates with each other through the server, which is different from the previous development of the socket communication. The point is that the use of multi-threading technology avoids the phenomenon that the number of processes is too large and the system is stuck during communication.

Keywords

TCP ;Socket network interface ;Embedded Linux; Multithreading.

1. Introduction

With the popularity of the Internet of Things and AI, more demands are placed on the basic hardware of embedded devices. The connection of objects and objects requires that embedded and network be combined to meet the various needs of consumers. The most popular network protocol today is the TCP protocol, and the use of the socket network interface makes it easier to implement networking operations. Most of the traditional development is based on multi-process, the consumption of resources is relatively large, and the consumption of resources by multi-threading technology is relatively small. Since the thread is a lightweight unit compared to the process, the development of multi-threading technology is more reasonable and efficient. Commonly used in network protocols are TCP and UDP, TCP is the transmission control protocol, connection-oriented protocol, similar to the call to maintain the connection. The advantage is that all error data can be automatically resent, and the security and integrity of the data are ensured. The receiver of the data can notify the data sender to control the data flow in real time. The disadvantage is that the pressure of the server is relatively large, and the resource occupancy rate is relatively high [1]. UDP is the user datagram protocol, non-connection-oriented protocol, similar to texting does not require a full connection, communication needs to be connected in an instant. The advantage is that the pressure of the server is relatively small, and the resource occupancy rate is relatively low. The disadvantage is that the error data is not automatically re-transmitted, and the security and integrity of the data are not guaranteed, and the receiver of the data cannot notify the sender to control the traffic in real time. In

view of the advantages and disadvantages of TCP and UDP network protocols, this paper chooses TCP as the communication type of socket[3].

2. Introduction to network knowledge and important functions

2.1 Network part introduction

IP addresses are often used in network programming. The IP address is mainly divided into two parts, namely: network address and host address. The subnet mask is used to divide the network address and host address in the IP address. The subnet mask is used to determine whether the two IP addresses are in the same LAN. Method of dividing an IP address: IP address & subnet mask. For example, the IP address is 172.28.17.46, the subnet mask is 255.255.255.0, the network address is 172.28.17, and the host address is 45. According to the IP address, you can locate a specific host. According to the port number, you can locate a specific process in the host. The port number is essentially an unsigned short type. The port number ranges from 0 to 65535, but the port number between 0 and 1024 is generally occupied by the system[2]. The usage starts from 1025. The IP address and port number are provided in network programming. We know that the system of byte order is divided into small endian systems and big endian systems. In the little endian system, the low-order data is stored in the low-order memory address, and the big-end system stores the low-order data in the high-order memory address [4]. Generally, the data sent to the network is converted to the network byte order and then sent before being sent. The data received from the network is first converted to the host byte order and then parsed; and the network byte order is essentially big endian. The endianness of the system, the byte order of the host refers to the endianness of the local system.

2.2 Introduction to the function

The essence of Socket is the communication interface or carrier. The socket (int domain, int type, int protocol) function is mainly used to create communication points to realize communication. Successfully returned the socket's descriptor, failing to return -1. The first parameter: protocol family / domain. Mainly decide whether local communication or network communication. AF_UNIX/AF_LOCAL local communication, AF_INET based on ipv4 network communication, AF_INET6 based on ipv6 network communication. The second parameter: the type of communication. SOCK_STREAM is based on connection-oriented, orderly and reliable data stream transmission. The essence is the communication method based on TCP protocol. SOCK_DGRAM is based on non-connection-oriented, unreliable datagram transmission, which is essentially the communication method based on UDP protocol. The third parameter: the specific protocol, the default is 0.

That implements network communication .The structure type struct sockaddr_in{

Sa_family_t sin_family;//protocol family AF_INET.

In_port_t sin_port;//port number

Struct in_addr sin_addr;//IP address

};

The bind(int sockfd, const struct sockaddr *addr, socklen_t addrlen) function is mainly used to bind sockets and communication addresses. The first parameter: the socket descriptor, the return value of the socket function. The second parameter: the first address of the communication address, type conversion. The third parameter: the size of the communication address, calculated using sizeof.

The connect(int sockfd, const struct sockaddr *addr, socklen_t addrlen) function is mainly used to connect sockets and communication addresses. For parameters and return values, please refer to the bind function. The listen(int sockfd, int backlog) function is mainly used to monitor the number of connections on the specified socket. The first parameter: the socket descriptor, the return value of the socket function. The second parameter: the maximum number of connections allowed to be queued for response. A similar accept function is mainly used to accept/respond to a client connection request on a socket, successfully returning a descriptor for communication, and failing to return -1. The first

parameter: the socket descriptor, the return value of the socket function. The second parameter: the first address where the client's communication address is stored. The third parameter: the size of the space where the client's communication address is stored. The pthread_create function is mainly used to create a new thread in the currently calling process, successfully returns 0, and returns an error number. The first parameter: the pointer type, used to store the ID of the new thread. The second parameter: the attribute of the thread, the default is NULL. The third parameter: the function pointer type, which represents the thread's processing function. The fourth parameter: the actual parameter [2] as the third parameter.

The above functions need to pay attention to the fact that the descriptor created by the socket function is mainly used for listening, and the descriptor created by the accept function is mainly used for communication.

3. Design and Implementation of Communication System in Linux

3.1 TCP protocol network communication process

The entire communication system adopts the C/S architecture, and uses the socket principle to implement network communication based on the TCP protocol. The flow chart of the TCP network program is shown in Figure 1 below.

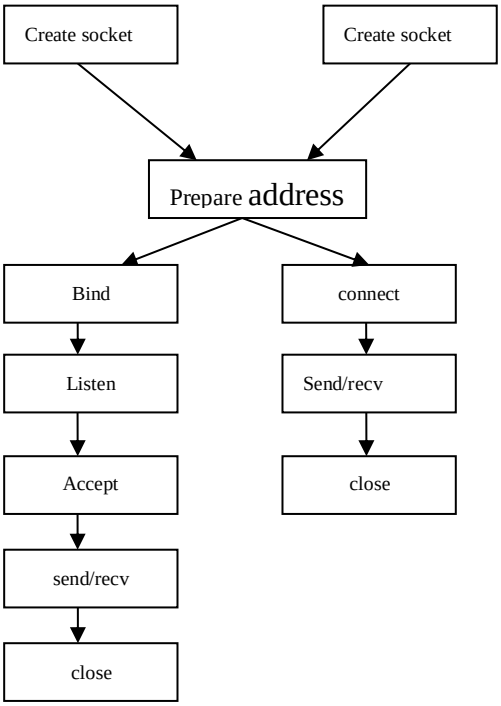

Figure 1 TCP program flow chart

Server network communication based on TCP protocol is roughly divided into seven steps:

(1) Create a socket, use the socket function

(2)Prepare the communication address, using the structure type

(3)Bind socket and communication address, using bind

(4) Listen, use the listen function

(5)Respond to the client's connection request using accept

(6)Communication, use the send function, etc.

(7)Close socket

The client network communication based on the TCP protocol is roughly divided into five steps:

(1) Create a socket, use the socket function

(2) Prepare the communication address, address of the server

(3)Connect socket and communication address, use connect

(4)Communicating, using the recv function, etc.

(5)Close socket

3.2 Socket network communication implementation

This experiment was conducted under the 64-bit UBUNTUN 16.04 version of the Linux operating system. The operating system kernel version is s3c-Linux-2.6.28.4. The entire program flow chart is shown in Figure 2 and Figure 3.

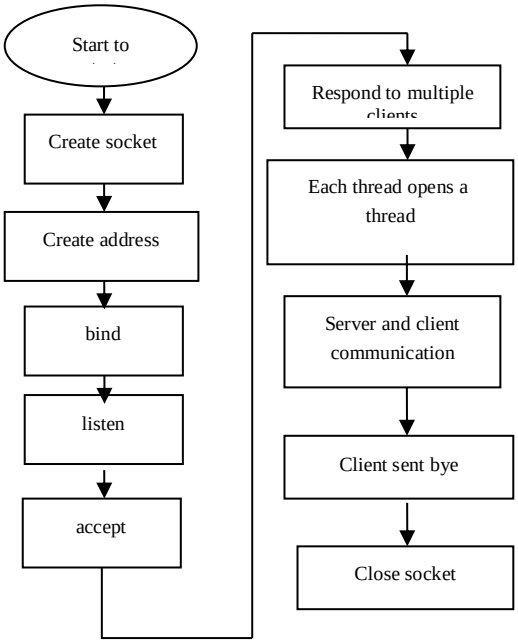

Figure 2 Server flow chart

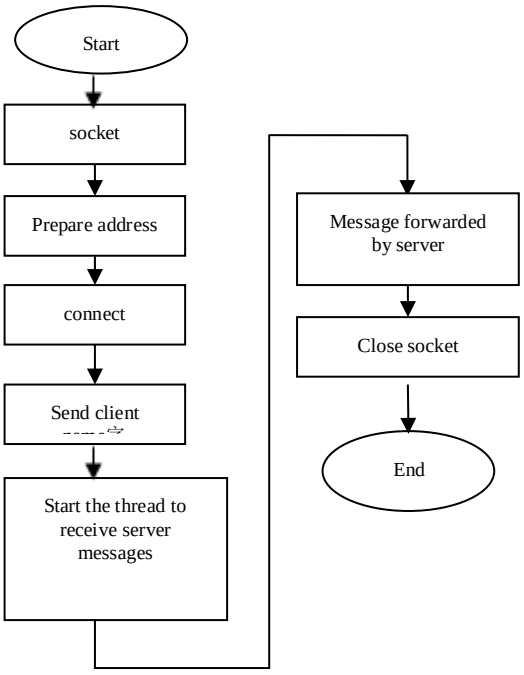

Figure 3 Client flowchart

4. Experimental Results

After writing the server program according to the server's flowchart, do the following:

gcc -pthread server.c -o tcpA

./tcpA

Run the client program on Linux as follows:

gcc -pthread client.c -o client

./client

After the server and client connect successfully, the running result is seen through the terminal as shown in Figure 4 and Figure 5. The experiment shows that the communication is realized, and the result is consistent with the expectation. The biggest advantage of this experiment and the previous embedded application development is the adoption of multi-threading technology. Since the thread relative process is lightweight, a thread can be opened for each networked client device. When the client exits, the thread is closed. This has the advantage of high resource utilization.

```
lsz@ubuntu:~/BM$ gcc tcpA.c -o tcpA
lsz@ubuntu:~/BM$ ./tcpA
Create socket success...
Binding success!
Monitor success...
Turn off the server, press ctrl+c...
Client172.18.57.49 connection success...
The message sent by the client172.18.57.49 is:Hello!
the message size is: 6
```

Figure4 Server operation diagram

```
lsz@ubuntu:~/BM$ ./client
Please enter your name:zhouqifeng
The chat room started to start...
Client startup success!
Warmly welcomezhouqifeng to log in to this chat room..
Please enter TF for the transfer file:
TF
Please enter TF for the transfer file:01.txt
Transfer file success !
```

Figure 5 Client running diagram

5. Conclusion

This article uses multi-threading technology, which can open up a thread for each client to serve it. At the same time, multi-threading can work in parallel, which can avoid the phenomenon that the operating system is stuck in the device with relatively tight resources in the embedded system, which is the software development and networking of the embedded device. The application has certain inspirational value.

Acknowledgements

This work was supported by the Youth Science Foundation project (Fund No. 61603153).

References

[1] Dong W. Kim,Ho-Dong Lee,Clarence W. de Silva,Jong-Wook Park. Service-provider intelligent humanoid robot using TCP/IP and CORBA[J]. International Journal of Control, Automation and Systems,2016,14(2).

[2] Guo Li, Jin Zhang, Ang Peng, Rulong Wang. Web Services System Optimization Based on

Modular Classification and Multi-Threading[P]. Management and Service Science (MASS), 2010 International Conference on,2010.
[3] Lili Fan. A Study on TCP/IP Network-based Embedded Linux Intelligent System[P]. 2016 4th International Conference on Electrical & Electronics Engineering and Computer Science (ICEEECS 2016),2016.
[4] Amiri, Rami,Elkeelany, Omar. Fusing Internet Protocol (IP) receive module at receiving path of open TCP/IP custom Single-Purpose Processor[P]. SOUTHEASTCON 2014, IEEE,2014.