Realization of Transmission Control Protocol Based on μC/OS-II

Qianyuan Wang(✉) and Yujun Gao

Tongji University, No. 4800 Cao'an Highway, Shanghai 201804, China
wangqianyuanl23@qq.com

Abstract. TCP/IP protocol suite plays an important role in computer network, in which TCP is a crucial protocol. To realize a general TCP on μC/OS, this paper adopts the embedded TCP/IP protocol stack called μC/IP as a basic version. The original TCP code in μC/IP will be modified to realize all functions defined in standard TCP. The modified TCP is integrated with standard IP, network interface protocol and tested on STM32F407 demoboard. The demoboard can communicate with computer, which proves the correctness of the modified TCP.

Keywords: TCP · μC/OS-II · Transmission reliability

1 Introduction

Operating system is important in embedded devices. μC/OS-II [1] is an operating system with open source code, compact structure and deprived real-time kernel. Therefore, most projects prefer to choose μC/OS-II as embedded operating system.

Embedded TCP/IP protocol stack is extensively adopted in embedded devices for communication. LwIP [2] and μC/IP [3] are two widely used embedded TCP/IP protocol stacks. It is difficult to separate TCP from LwIP. Whereas, it is very easy to separate TCP code from μC/IP which is designed for μC/OS-II. To realize a general separated TCP function, this paper selects μC/IP as embedded TCP/IP protocol stack. The TCP code in μC/IP does not contain all functions defined in the standard TCP. Therefore, this paper modifies the TCP code in μC/IP to realize a general TCP containing all functions.

The main work in this paper contains three parts. The first part is transplanting LwIP and μC/OS-II in the demoboard. The second part is modifying the existing TCP code in μC/IP and replacing the TCP in LwIP with the modified TCP. The final part is building a TCP client and a server task to test the modified TCP.

The rest of the paper will be organized as follows: In Sect. 2, μC/OS-II is introduced. In Sect. 3, the detailed implementation of TCP is explained. In Sect. 4, the testing results are given. In Sect. 5, we will conclude the paper.

X. B. Zhai et al. (Eds.): MLICOM 2019, LNICST 294, pp. 573–579, 2019.
https://doi.org/10.1007/978-3-030-32388-2_49

2 Brief Review of µC/OS-II

µC/OS-II is a real-time operating system. Because it takes up a small amount of space, it is very suitable for transplanting to embedded devices.

µC/OS-II provides many mechanisms to protect shared data and provide intertask communication [1]. Semaphore is an important mechanism. µC/OS-II's semaphores consists of two elements: a 16-bit unsigned integer used to hold the semaphore count, and a list of tasks waiting for the semaphore count to be greater than 0. Semaphore is used to protect shared data. If semaphore is not available, a task will need to be put to sleep until another task signals the semaphore. Therefore, shared data will be used by only one task.

3 The Implementation of TCP

TCP is a connection-oriented and reliable transport layer protocol [4, 5]. TCP operations include connection establishment, data transfer, and connection termination. A TCP connection is point-to-point and it consists of a server and a client. The process that initiates the connection establishment is called the client process. The process that passively waits for the connection to be established is called the server process.

The flowchart of the client and server is shown in Fig. 1. The client and the server bind the source port number and source IP address. Then, the client will connect with the server. After the connection is established, the client and the server can transmit and receive data. When the client and the server don't need to transmit or receive data, they will close the connection.

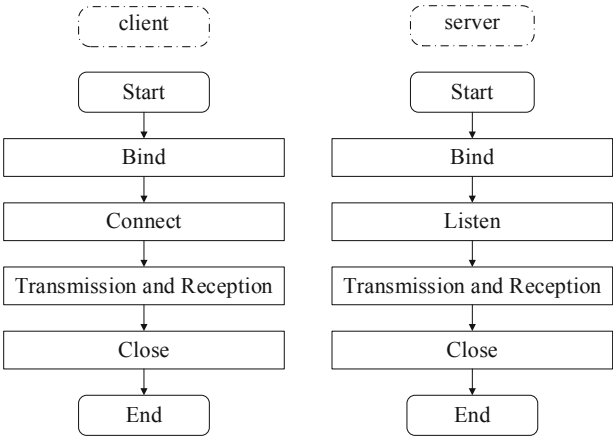

Fig. 1. The flowchart of client and server.

3.1 Connection Establishment

Connection establishment is called three-way handshake. The server and the client bind their own source port number and source IP address. Then the client sends a SYN segment to the server. After receiving the SYN segment sent by the client, the server will also send a SYNACK message to the client. Finally, the client will send an ACK segment to the server. The connection between the client and the server is established. Connection establishment sets initial parameters such as the sending sequence number and the receiving sequence number.

3.2 Data Transfer

A TCP connection provides a full-duplex service. Therefore, the server and the client need to establish their own transmitting and receiving buffers. A TCP connection provides a reliable data transfer service which ensures that data stream is in sequence and not duplicate. As illustrated in Fig. 2, there are four functions to transmit and receive data, namely tcpInput, tcpOutput, tcpRead and tcpWrite. The function called tcpRead is used to receive data from transport layer to application layer. The function called tcpWrite is used to send data from application layer to transport layer. The function called tcpInput is used to receive data from network layer to transport layer. The function called tcpOutput is used to send data from transport layer to network layer.

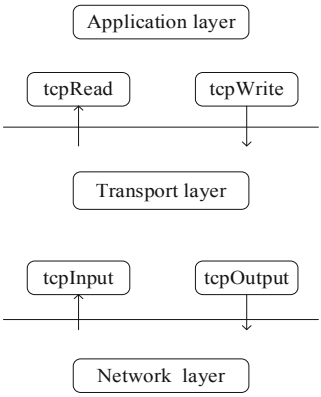

Fig. 2. Functions of data transmission.

As shown in Fig. 3, tcpInput receives and processes segments from network layer. Firstly, the received segment needs to be checked. If checked unsuccessfully, the segment will be discarded. If the segment is checked successfully, the keepalive timer will be reset. Then, data in the received segment will be extracted and saved in receiving buffers. When a new segment is received, out-of-order segments will be handled.

There are only two TCP congestion-control algorithms (slow start and congestion avoidance) in the TCP of μC/IP. By modifying the TCP code, another two TCP congestion-control algorithms (fast retransmit and fast recovery) are added.

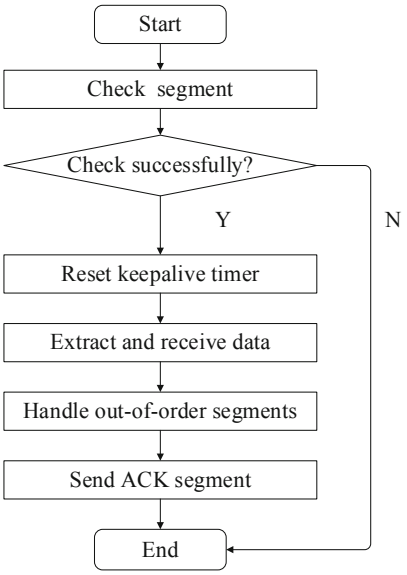

Fig. 3. The flowchart of tcpInput

As shown in Fig. 4, tcpOutput sends segments from transport layer to network layer. If the sending window is 0, the sender will open persistence timer to update the sending window. When the sending window is greater than 0, data will be extracted from the sending buffer to the segment. Then the TCP header of the segment will be filled. When the segment is sent, the retransmission timer will open.

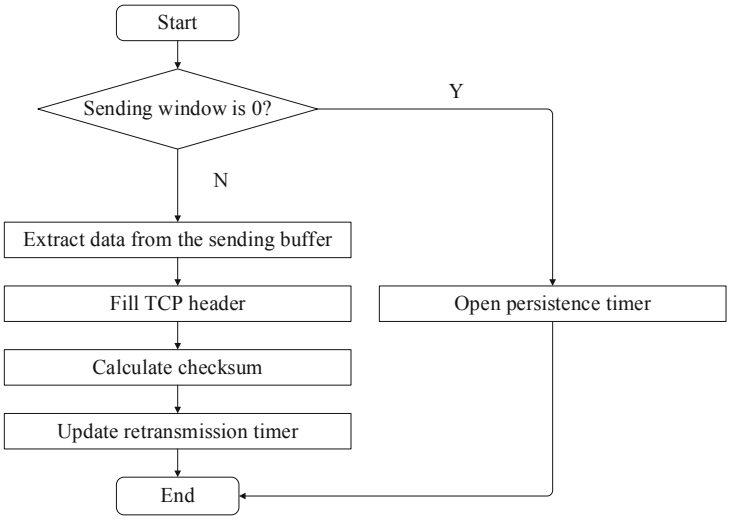

Fig. 4. The flowchart of tcpOutput

Data will be extracted from the receiving buffer to application layer by calling the function called tcpRead. The application process can choose the number of data that extracted from the receiving buffer.

Data will be put into sending buffer by calling the function called tcpWrite. The data will be sent by calling the function called tcpOutput.

3.3 Timer Management

There are mainly three timers (retransmission timer, persistence timer and keepalive timer) in a TCP connection.

To prevent the segments from being unacknowledged or lost, retransmission timer is established. If the segment that has been sent is not acknowledged within a certain period of time, the sender will resend the segment to the receiver. Therefore, the data stream will be in sequence and the transmission will be reliable.

When the sender's sending window is 0, the sender will open persistence timer to check whether the receiving window of the receiver is greater than 0. When the receiver' receiving window is greater than 0, the persistence timer will be closed.

The keepalive timer is used in the server. The keepalive timer is mainly to check whether the client is alive. When the client and the server don't transmit or receive data for a long time, the server will send a segment without data to check whether the client is closed.

The keepalive timer and the persistence timer are combined in the TCP of μC/IP. By modifying the TCP code of μC/IP, the keepalive timer and the persistence timer will work individually.

A timer task is established to process the timers in TCP. Firstly, an empty timer linked list is created. Then different timers are inserted into the timer linked list. The timer task will check whether the latest timer is timeout. When the timer is timeout, the related function will work and the timer will be deleted from the timer linked list.

3.4 Connection Termination

The client or the server can decide to close. Suppose that the client decides to close the connection. Firstly, the client sends a FIN segment to the server. After receiving the FIN segment, the server will send an ACK segment to the client. When the server also decides to close, it will send a FIN segment to the client. The client will send an ACK segment. Eventually, the connection between the client and the server is closed.

4 Testing Results

The hardware is STM32F407 demoboard. μC/OS-II is transplanted in demoboard. Standard IP and network interface protocol is integrated by LwIP. Finally, the modified TCP code is downloaded in demboard. The demoboard and the computer are connected via a network cable.

As shown in Fig. 5, TCP client is established in embedded device and TCP server is established in the computer. The computer's source IP address is 192.168.1.108 and

the computer's port number is 8087. The embedded device's source IP address is 192.168.1.101 and the embedded device's source port number is 49154. The computer receives 750 bytes from the embedded device. It proves that the modified TCP can work with standard TCP.

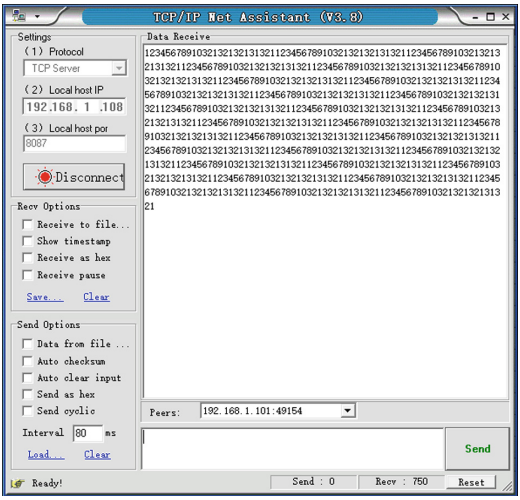

Fig. 5. TCP client is the embedded device. TCP server is the computer.

As shown in Fig. 6, TCP client is established in the computer and TCP server is established in the embedded device. The computer's source IP address is

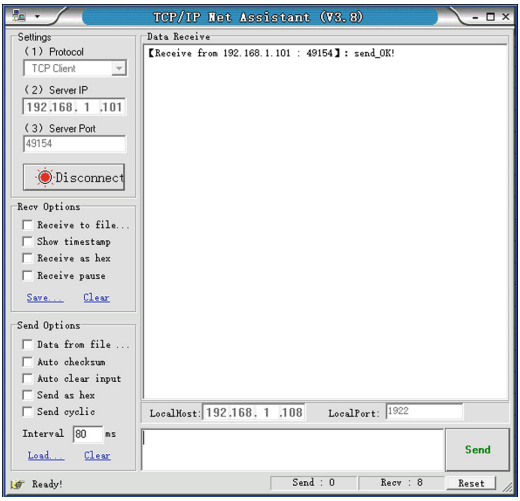

Fig. 6. TCP client is the computer. TCP server is the embedded device.

192.168.1.108 and the computer's source port number is 1922. The embedded device's source IP address is 192.168.1.101 and the embedded device's source port number is 49154. The computer receives 8 bytes from the embedded device.

5 Conclusions

This paper completes the transplantation of TCP in embedded devices. Tested by net assistant, the modified TCP communicates successfully with standard TCP. Many parameters can be reconfigured according to users' requirements in the modified TCP code. Therefore, the modified TCP is a general TCP and easy to change and use for users.

Acknowledgment. This work is supported by the National Natural Science Foundation of China under Grant No. U1733114, the Fundamental Research Funds for the Central Universities, and Shanghai Rising-star Program under Grant No. 19QA1409100.

References

1. Labrosse, J.: MicroC/OS-II: The Real-Time Kernel, 2nd ed, pp. 153–178. CMP Books (2002)
2. Dunkels, A.: Design and implementation of the lwIP TCP/IP stack, pp. 1–3. Swedish Institute of Computer Science (2001)
3. http://ucip.sourceforge.net
4. Kurose, J.F., Ross, K.W.: Computer Networking: A Top-Down Approach, 6th edn, pp. 230–285. Addison-Wesley, Boston (2001)
5. Stevens, W.R.: TCP/IP Illustrated (Vol. 1): The Protocols, 2nd edn, pp. 575–803. Addison-Wesley, Boston (2012)