ATLANTIS PRESS
Advances in Intelligent Systems Research (AISR), volume 150
2017 International Seminar on Artificial Intelligence, Networking and Information Technology (ANIT 2017)

Real-time Optimization of TCP/IP Protocol Stack in Local Area Network Environment

Chengcheng Yang[1, a] , Yongchao Tao[1, b], Wencheng Xiang[1, c] , Xianghu Wu[1, d]

[1] Shenzhen Academy of Aerospace Technology, Shenzhen, China

[a]sjjg_hit@163.com, [b]taoyongchao652@163.com, [c]HPEC_SAAT@163.com [d]wxh_hit@126.com

Keywords: TCP/IP, real-time communication, LwIP protocol.

Abstract. As the basic protocol for Internet communication at present, TCP/IP, which offers a unified way to communicate through different nodes, is widely applied in various communication environments and has become the basis of Internet. But the original TCP/IP protocol is designed and realized for the transmission environment of WAN Internet, it is unable to optimal fit some communication environments which may have higher requirements of real-time. This paper focuses on local area communication environment, and makes an effort to research and realize the extended TCP/IP protocol.

1 Introduction

Today, the computer model is diverse, the application of the operating system is not the same, but they can communicate thanks to TCP/IP protocol. TCP/IP protocol is the Internet's most basic protocol, by the transport layer of TCP transmission control protocol and the network layer of IP network interconnection protocol. TCP/IP defines the standard way of connecting electronic devices to the Internet and the standard way that data is transferred between network nodes[1].

The TCP/IP protocol is a transport layer and a network layer protocol developed for wide area Internet. Among them, TCP protocol is an end-to-end reliable transmission protocol, mainly to solve the problem of Internet data transmission delay, high transmission error and the order can't guarantee[2]. The standard TCP protocol uses unformatted character stream model, can't achieve the priority transmission of high priority packets, do not meet the needs of real-time communication. In addition, the standard TCP / IP protocol algorithm is designed to take into account the Internet data transmission delay, transmission error rate is high and so on. It uses a more complex overtime retransmission, error detection, sliding window and other flow control and error detection methods, resulting in real-time and efficiency are affected.

In the local area network environment, the applications of real-time generally have a higher demand[3]. This article is mainly to optimize the TCP / IP protocol stack, for the local strong real-time application environment. This article will optimize the design of TCP module, UDP module, and IP module of LwIP[4].

LwIP, full name Light Weight IP, it is a lightweight communication protocol, the kernel does not have operating system targeted, and it can achieve a perfect communication function both in the Operating System environment and in the environment without Operating System[5]. LwIP is universal to the operating system, which abstracts the code associated with the platform, and the user must encapsulate the code in order to port it to the personal system, which provides more API interfaces for later network applications.

2 Real-time Improvement of Internal Algorithm of TCP Module

We need to optimize the TCP protocol module design, from Send Start Management, Congestion Control, Timeout Retransmission, Packet Acknowledgment and other aspects.

1) Send Start Management

For real-time data transmission, TCP slow start will cause a large number of packets at the beginning of the delay sent. Real-time TCP uses a "fast start" approach to avoid the beginning of

the sliding window due to the small data caused by delayed transmission. The specific approach is to use a large sliding window initial value, ranging from the front of the packet is ACK, and to send to the new queue to send the packet.

Send management in the real-time protocol stack in the specific optimization changes reflected in the following two aspects. On the one hand, in the tcp_connect () function of the tcp.c file,It initializes the newly established TCP connection corresponding TCP control block tcp_pcb the value of each field, in fact, includes the sliding window to control the sliding window size field "cwnd".The traditional slow start process this value is set to 1, we use the fast start mode, the use of large sliding window initial value TCP_WND / TCP_FAST_SETUP_PARA. On the other hand, in the tcp_process () and tcp_recieve () functions of the tcp_in.c file, for the fast start algorithm, we will expand the window to twice the current speed, the acceleration window in the initial expansion of the speed. Only need Log2 (TCP_FAST_SETUP_PARA) ACK packets can be extended to send the maximum window, enter the congestion control phase.

2) Congestion Control

Congestion control in the real-time protocol stack in the specific optimization changes reflected in the TCP connection to determine the network congestion that is sent when the packet loss situation. In order to achieve the sliding window hold technology, we add a sliding record field slow_keep_count in each TCP connection corresponding to the control block tcp_pcb, record the number of consecutive occurrences of time. This field is initialized to 0 when the pcb is created, and the field is incremented when the TCP connection corresponding to the pcb control block times out. This field is cleared when a packet is received normally again. The Congestion Control Sliding Window Technology The main control flow is shown in Figure 1.

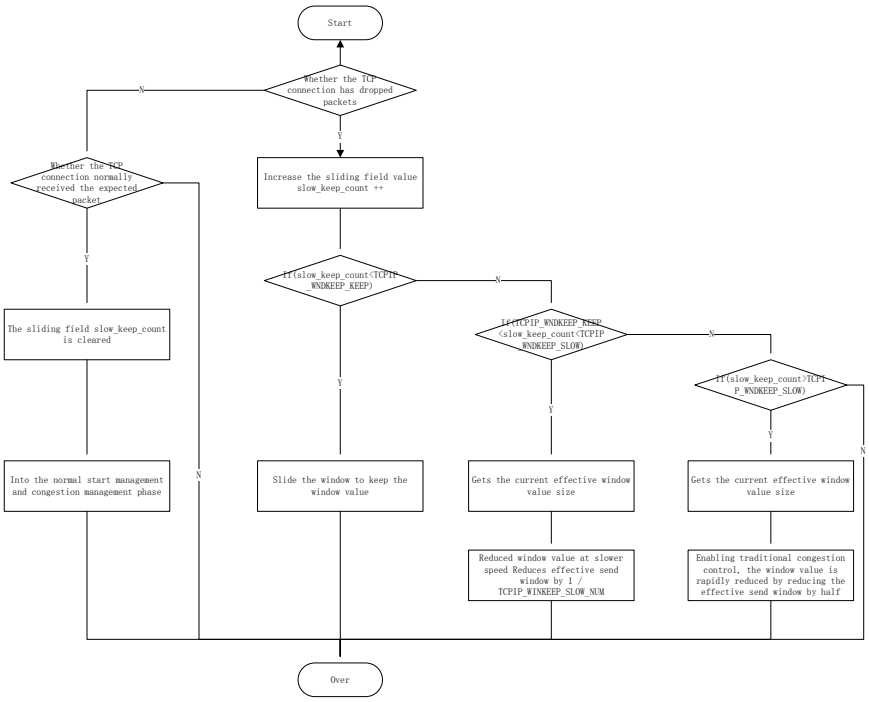

Fig.1. the main control flow of Congestion Control sliding window technology

3) Timeout Retransmission

Timeout retransmission timer used by traditional TCP is set to a minimum of several hundred milliseconds (Linux RTO minimum value of 200 milliseconds). Obviously it is not suitable for real-time communication requirements. To this end, real-time TCP uses a highly sensitive timeout retransmission timer, the timer timeout value is set in a few milliseconds range. Its specific optimization changes are reflected in the sys_arch.c fileSys_init_timeing () function and sys_get_ms_longlong () function.

4) Packet Acknowledgment

Traditional TCP protocol packet authentication management is mainly reflected in the delay

acknowledgment (DACK) technology, TCP delay confirmation is likely to cause data delay transmission. Real-time TCP uses fast packet acknowledgment technology. That is, an ACK operation immediately occurs after receiving the packet, to let the data sender immediately known whether the data has arrived.

Packet Acknowledgment management in the real-time protocol stack in the specific optimization changes reflected in the tcp_out.c file tcp_output (). This article requires fast packet acknowledgment, then call the tcp_nagle_disable () function to cancel the Nagle algorithm of the TCP control block cp_pcb, cancel the DACK, and directly acknowledge each received TCP packet for fast packet acknowledgment.

3 Real-time Improvement of Internal Algorithm of UDP Module

We need to optimize the UDP protocol module design, from Packet Acknowledgment, Select Retransmission Management and other aspects.

1) Packet Acknowledgment

Real-time UDP maintains a number for each packet that is sent out. Unlike TCP serial number, real-time UDP does not use the number of bytes transmitted as the serial number, but the number of the entire packet, so that maintenance and management are very efficient.

2) Select Retransmission Management

For packets that are not acknowledged by the receiver in time, the real-time UDP will be retransmitted immediately. If the packet sent after the first received confirmation, the previously sent out of the data packets are considered lost, selective retransmission.

Packet validation management and selection of retransmission management changes are in order to add real-time protocol stack real-time data packets on the real-time support for real-time flags with the packet, to ensure that its relatively ordinary data packets have a higher Priority for transmission. Especially in the network load is relatively high, to ensure that real-time data packets sent in a timely manner, and require packet validation management and selective retransmission, to ensure that real-time packet information has been real and successful transmission. The Packet Acknowledgment and Selective Retransmission Mechanism control flow is shown in Figure 2.

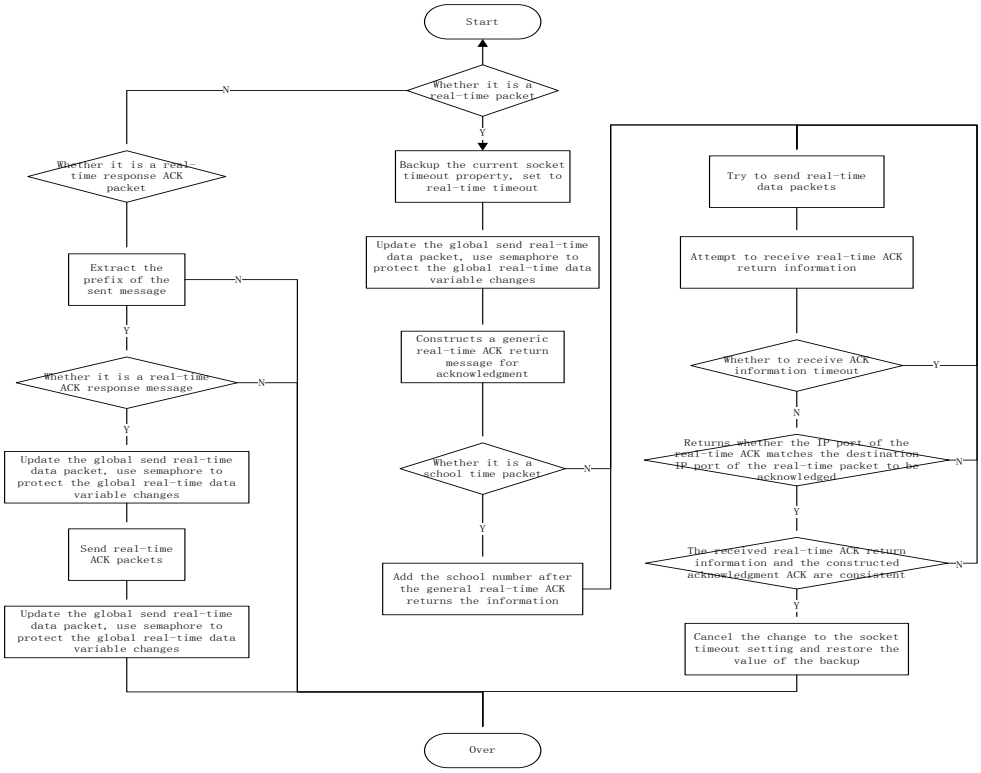

Fig.2. Packet acknowledgment and selective retransmission mechanism control flow

Choose to restore the management module to achieve and confirm the management module together to achieve the same belong to Lwip_sendto () function in the sockets.c: file. When sending real-time data packets, backup the existing socket to receive the timeout attribute information, set the timeout time of the socket to be real-time UDP to confirm the timeout time. If you wait for the confirmation message, if the timeout time is exceeded, retransmit the packet. Received the confirmation information number, the algorithm determines whether the received information is a legitimate confirmation information, whether it is from the correct IP address and port and whether it is the correct serial number, if they are all correct, the algorithm confirms success, and restores the raw timeout information of the socket. Otherwise it is judged as a confirmation failure, retransmit the real-time data packet, re-enter the confirmation management module.

4 Real-time Improvement of Internal Algorithm of IP Module

We need to optimize the IP protocol module design, from Priority Management, Segmentation and Reorganization Management, Error Control and other aspects.

1) Priority Management

Real-time IP distinguishes real-time data streams and non-real-time data streams. DSCP protocol uses real-time packets for special priority marking to ensure that their transmission clock takes precedence over non-real-time packets. Another problem with priority management is how to avoid high-priority time synchronization UDP packets that can't be received due to full buffer. This problem can be solved in the IP layer of the buffer for the time synchronization UDP packet to retain resources. The priority part of the real-time packet management The key part of the control flow is as follows:

```
if(flags != RTTCPIP_REALTIME_SEND_FLAG)     {
      // A regular packet is waiting for up to    WAIT_COUNT times
   for(tempwaitingcount=0; tempwaitingcount< WAIT_COUNT; tempwaitingcount ++)
    {
            // determine whether there is a real-time data packet being sent
            // Obtains the information of the global sending real-time data packet, and judges whether or not to suspend for the priority transmission
            tempusingrealtime = using_realtime_udp;
            // If the global real-time packet information is less than the threshold to send the general packet, it does not need to wait, jump out of the loop
            if(tempusingrealtime < REALTIME_WAIT_CONTROL)    break;              }
            Else {
            // Ordinary UDP packets are sent for a time to wait for REALTIME_WAIT_TIME, and the packets are sent in priority
                  Sleep(REALTIME_WAIT_TIME);
            }
      }
 }
```

2) Segmentation and Reorganization Management

From the characteristics of high real-time requirements, the algorithm eliminates the overhead of IP data segmentation and reorganization, and real-time IP simplifies the segmentation of datagrams. For standard Ethernet, the MTU size is 1500 bytes, minus the IP header 20 this section, the remaining 1480 bytes as the upper layer TCP protocol to provide a valid data load, which for the general control data transmission has been enough.

3) Error Control

As the Gigabit LAN network environment is stable and reliable, the bit error rate is very low, too complicated and cumbersome reliability verification takes a long time and lack of necessary. Real-time IP selects a simple validation algorithm to eliminate unnecessary time overhead.

5 Communication and response speed test for Real - time protocol stack

We use windows-windows, RTX-RTX and the algorithm of this paper to do communication and response speed comparative test. The test results are shown in Table 1 below:

Table.1. Communication delay comparison test results

Test items	Min Com-delay	Max Com-delay	Average Com-delay
windows-windows	125us	483us	263us
RTX-RTX	205us	270us	217us
algorithm of this paper	146us	238us	183us

RTX protocol stack and real-time protocol stack response speed of the test results shown in Table 2 below:

Table.2. Response time comparison test results

Test items	100KB/s	1MB/s	3MB/s	7MB/s	10MB/s
Real-time data packets	16us	16us	18us	21us	24us
Ordinary data packets	18us	17us	21us	25us	32us
RTX packets	16us	19us	24us	31us	44us

The transmission response time of Real-Time protocol stack real-time data packet does not change much with the increase in network load. It can be seen that real-time data packets have high priority transmission control, and real-time transmission of real-time data packets can be ensured when the network load is large.

6 Conclusion

This paper analyzes the traditional algorithm of TCP / IP protocol stack, modifies the complex algorithm and mechanism for real-time communication in the local area network environment, and optimizes the communication performance and real-time performance of the protocol stack from the internal algorithm level of the protocol stack. And this article adds real-time data packet transmission support, to ensure that real-time packet information has been real and successful priority transmission.

Acknowledgement

In this paper, the research was sponsored by the future industry special funds of Shenzhen (Project No. CXZZ20140718154349249).

References

[1] Jaehyun Park, Youngchan Yoon，An extended TCP-IP protocol for real-time local area networks，Control Engineering Practice 6 (1998) 111-118

[2] Aaron Gember，Real-Time TCP for Embedded Devices，Marquette University, Dept. of Mathematics, Statistics, and Computer Science

[3] C. Zhang and V. Tsaoussidis，TCP-Real Improving Real-time Capabilities of TCP over Heterogeneous Networks，NOSSDAV'01, June 25-26, 2001, Port Jefferson, New York, USA.

[4] Hans Weibel etc., IEEE 1588 Implementation and Performance of Time Stamping Techniques, 2004 Conference on IEEE 1588.

[5] APOSTOLOS MELIONES.Engineering of Embedded Linux ATM for MPC8260 and Derivatives. Proceedings of the 9th WSEAS International Conference Communications . 2005