The Research and Implementation of Embedded TCP/IP Protocol Stack

Jian Xu
School of Information and Engineering
Hubei University for Nationalities
Enshi, China
hwscu@hotmail.com

***Abstract*—With μC/OS-II as embedded real-time operating system and LWIP as protocol stack in embedded system, μC/OS-II and LWIP are transplanted to constitute the software platform. The hardware platform is constituted by the LPC2210 as MCU, RTL8019AS as the Ethernet controller chip. The thesis mainly elaborates on the principle and communication process of the LWIP protocol stack layers. Test results show that: when transplanted to LPC2210, μC/OS-II kernel can achieve multi-task scheduling, the key agreements of LWIP protocol stack layers can achieve normal communication with the Internet connected at the embedded terminal.**

Keywords-EmbeddedSystem;Ethernet;LPC2210;μC/OS-II;LWIP

I. INTRODUCTION

To realize the internet network of embedded system and support the terminal access to internet, TCP/IP protocol stack must be implemented in embedded system. So far a number of TCP/IP stacks can be used on the embedded device, but the application of open source for low-end embedded network platform is still very rare. LWIP is one of the implementations of TCP/IP protocol stack, which mainly reduces the amount of system memory utilization and code complexity, making it more suitable for embedded systems with limited resources.

II. EMBEDDED TCP/IP PROTOCOL STACK

A. LWIP Process Model

TCP/IP protocol stack process model refers to the approach which divides the system into different processes. The most common kind of process model is every agreement among the TCP/IP protocol exists as a separate process. This model uses the strict protocol layering and the communication nodes between agreements must be strictly defined. In addition, a more common process model seals the communication protocol in the kernel of operating system, and the application process is realized through the system call and communication of protocol stack so that each layer of agreement need not be strictly distinguished, instead it can use the cross-layering agreement technology.

LWIP process has its own unique model, which has encapsulated the whole protocol stack in a process so as to separate protocol stack from operating system kernel. The application layer process can either be a separate process, or reside in LWIP process. The main advantage of LWIP protocol stack as a process is to facilitate the migration in different operating systems [1].

B. Buffer and Memory Management

In terms of transmitting data, LWIP protocol packs data in a certain storage area. Nothing but data pointers are transmitted in protocol stack, a true copy of data can be realized only when the data is eventually sent by driver or removed the application program. And, in order to avoid the extra copy, LWIP stack design of the API allows application program to operate directly on the storage blocks [1].

Running TCP/IP protocol stack for embedded systems, the entire storage area of system can be divided into the protocol stack management buffer and the application management memory. As the memory region to which LWIP kernel can have direct access, LWIP stack management buffer is mainly used for loading the network data packets to be received and transmitted. The application management memory is the area for application management and operation, usually from the region for the application to send data and distribute cache.

C. Operating System Emulation Layer

In order to make LWIP easy to transplant, the function calls and data structure about operating system is not used directly in code, but use the operating system emulation layer instead of the use of these functions. The operating system emulation layer provides a series of unified interfaces to system services such as timers, process synchronization, and message transmission mechanism. Operating system emulation layer also provides a timer function used by the TCP. This timer is a mono-pulse timer with time interval of at least 200ms [1]. Process synchronization mechanism provides only a semaphore. E-mail messaging can be achieved by post and fetching. Postal operations will not block the process. On the contrary, the message is delivered to the mailbox by the operating system emulation layer in the list until other processes remove them.

III. LWIP TRANSPLANTING IN MC/OS-II

In the process of LWIP stack designing, the transplanting-relevant problems such as compiling constants set, defining data type, modifying the operating system-related functions, and initializing and driving the network card, have been taken into consideration. All the parts relevant to hardware, operating system, and compiler are separated by LWIP protocol stack code and placed in

Published by Atlantis Press, Paris, France.

/src/arch directory. So transplanting LWIP in μC/OS-II is realized by modifying the files in this directory [2].

A. The INCLUDE File Associated with the CPU or the Compiler

The definition (such as data length) associated with the CPU or the compiler is placed in /src/arch/include/arch directory, which mainly lies in the three documents cc.h, cpu.h and perf.h. This should be with μC/OS-II definition are the same.

```
#define BYTE ORDER   LITTLE ENDIAN
#define PACK_STRUCT_FIELD (x)  __packed  x
#define PACK_STRUCT_STRUCT
#define PACK_STRUCT_BEGIN   __packed
#define PACK_STRUCT_END
typedef  unsigned char     u8_t;
typedef  signed char       s8_t;
typedef  unsigned short    u16_t;
typedef  signed short      s16_t;
typedef  unsigned int      u32_t;
typedef  signed int        s32_t;
```

B. The Modification of Function Associated with the Operating System

In order to adapt to different operating systems, no system calls and data structures related to a particular operating system are used in the code of LWIP. *Instead,* an encapsulation layer between LWIP and operating system is defined in file sys_arch.c . The encapsulation layer provides a unified interface for the operating system service (such as timing, process synchronization, message passing).

C. The Implementation of Library Functions in lib_arch File

LWIP protocol stack used 8 external functions, which are usually related to the user's system or compiler. Therefore, they are to be realized by the user.

```
u16_t htons(u16_t n);
u16_t ntohs(u16_t n);
u32_t htonl(u32_t n);
u32_t ntohl(u32_t n);
int strlen(const char *str);
int strncmp(const char *strl, const char *str2, int len);
void bcopy (const void *src, void *dest, int len);
void bzero (void *data, int n);
```

IV. EMBEDDED TCP/IP PROTOCOL STACK

Based on real-time kernel μC/OS-II and 32-bit microprocessor LPC2210 hardware and software platform, embedded TCP/IP protocol stack is optimized to achieve the main general agreements such as ARP/RARP, IP, ICMP, TCP, UDP and so on.

A. Protocol Implementation of Network Access Layer

ARP (Address Resolution Protocol) protocol plays a role in the conversion of IP address and Ethernet address by resolving the network layer address to the MAC address of the network access layer so as to provide dynamic mapping between IP address and the corresponding hardware addresses.

Network access layer is at the bottom of TCP/IP protocol stack, which is responsible for receiving and sending IP packets by the selected network, or by the physical frame received from the network to extract the IP data grams to the IP layer. ARP function of LWIP mainly lies in the two documents etharp.c and etharp.h, which complete three main functions:(1)When sending a packet, it resolves IP address to the physical address;(2)when sending ARP request, it responds to ARP requests send by other hosts;(3)when receiving IP datagram, it updates ARP cache[3].

B. Network Layer protocol

Network layer is responsible for packaging the data according to the network standards, mainly including IP protocol, ICMP protocols and IGMP protocols. IP protocol is the basis of TCP/IP protocol, providing connectionless service for operating sequence of the different networks hosts sending packets. Adding packet-head of IP protocol before the packet enables each packet to address. ICMP protocol is mainly used to transfer error-free messages and other information worthy of attention. ICMP protocols specify a variety of protocol types and codes. If fully implemented, it would waste a lot of system resources, for the common embedded internet applications, it is only necessary to test whether the network is connected or not. Therefore, it needs to accomplish ping response protocol of the ICMP, in which the type number is 0, the code is 0. IGMP allows all systems in a physical network to know which multicast group the hosts are in currently, multicast routers need such information to know which interfaces the multicast packets should be transmitted. Embedded Internet communicates rarely by the means of using multicast, so it needn't be achieved in generally embedded Internet.

LWIP IP layer only implements the basic functions, such as the classification of IP packet received, information packaging and sending IP packet and so on, but can not receive or send IP fragments, nor can it deal with packet with IP parameters options. IP functions of LWIP are mainly achieved in two files ip.h and ip.c. ICMP protocol processing is mainly concerned with handling ping, including response to the received ping and sending ping request. LWIP mainly completes the three functions of ICMP protocol: ICMP echo reply, ICMP unreachable response and ICMP time-out response. The ICMP functions of LWIP are achieved in two files icmp.h and icmp.c.

C. Transport Layer Protocol

Devices running on the Internet are usually designated as the hosts, so the TCP/IP transport layer is sometimes called the host-host layer, because the layer involves the data transferring from a host to another .The basic functions of transport layer protocol include the reliable data transmission from the sender to the receiver. Before transmission it is necessary to divide the datagram into fragments, and re-assemble the packet before they are transmitted from the

transport layer to application layer for further processing. Transport layer protocols are mainly TCP and UDP.

TCP protocol provides application layer with a connection-oriented, reliable stream of binary data service. Connection-oriented means that a TCP connection must be created before two applications using TCP to exchange data with each other. The main functions are achieved in three files tcp.c, tcp_in.c and tcp_out.c. UDP is a simple protocol used in the decomposition of the information packet between different processes. Owing to the unreliability of IP protocol resulting from its definition without taking measures to ensure accurate communications, there may be some anomalies, such as packet loss. UDP is unsafe, nor is it connected, so main functions are achieved in the two documents udp.c and udp.h. Each UDP session state is stored in a PCB structure, which in turn is saved in a list retrieved to match when a UDP datagram arrives [4].

V. LWIP STACK TEST AND IMPLEMENTATION

After completing a series of analysis, migration of protocol stack and the programming of the Ethernet driver, etc., the initialization of LWIP can be conducted in the μC/OS-II in, so does the function testing of network. It is noteworthy that LWIP can be run only after μC/OS-II is fully started because the initialization involves semaphores, mailboxes and other RTOS related operations. Network function testing is done in the local area network with hardware development platform as a network communications server, the host computer PC as a network communication client. The two RJ45 network interfaces are connected via twisted pair, while the JTAG debug interface hardware development platform is connected with the PC's parallel port [5].

ARP, ICMP agreement experiment: run the command window at the PC client terminal and use the ping command: ping 192.168.0.110 (the server's IP address) to send the ARP query packet of the IP address to test connection of the Ethernet between the networks; run ARP query command arp-a to check ARP cache entries of hardware development platform MAC address[6].

As can be seen in Figure 1, the hardware development platform can receive the ping command issued by the PC and respond to ping requests. When the PC executes arp command, it can obtain the hardware development platform MAC address consistent with the set value. This shows that ARP and ICMP protocol in LWIP protocol stack can work correctly, and LWIP protocol stack works properly.

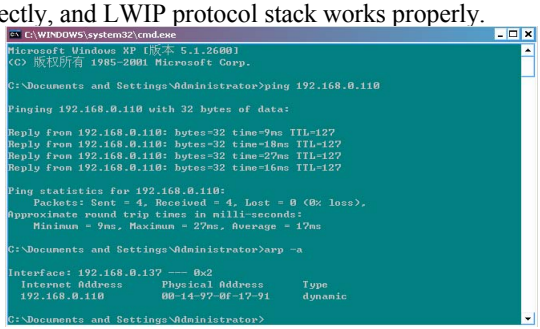

Figure 1. The result of ping and arp test

TCP agreement experiment: the hardware development platform as a TCP server, the local port number set to 1600, and the binding remote port number set to1500, the TCP test software on the PC can send both user datagram information and the user data stream. The received report on TCP server is put into the buffer allocation and then returned to the PC through the cable. Here Data report is tested as a case study. Test results are shown in Figure 2 in which the returned data from the server is the same as the sent data report from PC-side, thus proving the TCP protocol in LWIP stack can work properly.

Figure 2. The result of TCP test

Figure 3. The result of TCP test

UDP protocol test: the hardware development platform as a UDP server, the local port number is set to 1111 and the binding remote port number to 1222. The PC software can send both User Datagram information and the user data stream. The received UDP report is put into the buffer allocation and then returned the PC through the cable. If the user sends a user datagram the returned data can be seen from the PC software interface. If the user sends the file in the form of data streams (such as text format. txt files), the returned text file can be seen in the PC software. As can be shown in Figure 3, the PC (192.168.0.137:1222) sends UDP packets "THIS IS A UDP TEST! HELLO WORLD!" to the hardware development platform (192.168.0.110:1111), and receives the original packet returned by UDP server. This shows the UDP protocol in LWIP stack can run properly.

VI. CONCLUSION

Attaching importance to the study of LWIP stack, the paper makes a detailed analysis of the basic components of the LWIP principle, carries out in-depth study of the LWIP in μC/OS-II in the transplant process and gives specific transplant methods and the realization of key agreement.

Finally, on the basis of preparing the hardware development platform for Ethernet driver, it explores LWIP stack test on the hardware platforms and confirms the successful realization of TCP/IP protocol stack in embedded systems.

REFERENCES

[1] Adam Dunkels.Design and Implementation of the LWIP TCP/IP Stack[R],2001,3:PP3-7.

[2] Long Hainan,Liang Zhaobo.Adaptor between Ethernet and serial ports based on uC/OS-II and LWIP[J].Electric Power Automation Equipment,2007,(7): PP113-114.

[3] Ri Kunliao,Yue Fengji,Hui Li.Optimized Design and Implementation of TCP/IP Software Architecture Based on Embedded System[C].Machine Learning and Cybernetics,2006 International Conference on Aug,2006: PP590-594.

[4] Hague Tariqut,Urbaniak Michael.Step by Step Implementation of Ethernet and TCP/IP Protocols on an embedded System[J].Computers in Education Journa1.2004,14(4): PP70-87.

[5] Tae RimPark, Jae HyunPark, Wook HyunKwon. Reducing OS overhead for real-time industrial controllers with adjustable timer resolution[C].Industrial Electronics,2001. Proceedings.ISIE 2001.IEEE International Symposium on Volume 1,12-16 June 2001: 369 -374.

[6] Oh Soo-Cheol,Janq Hankook,Chung Sang-Hwa.Analysis of TCP/IP Protocol Stack for a Hybrid[C].TCP/IP 5th International Conference,PDCAT 2004.Singapore,2004: PP406-409.

Published by Atlantis Press, Paris, France.