Design and Implementation of embedded network based on LWIP

Xiao-qian Zheng,Yue Yuan

(School of Information Science and Engineering, Shandong University, Jinan 250100, China)

Abstract: LWIP is a light-weight TCP/IP protocol stack. Lwipis source－opened and can be transplanted very easily. According to the features of LWIP, this paper presents the application of LWIP in embedded functions by transplanting LWIP into μC/OS-II operating system.
Keywords：**LWIP, TCP/IP, μC/OS-II**

Manuscript Number: 1674-8042(2010)supp.-0030-04

dio: 10.3969/j.issn1674-8042.2010.supp..08

1 Introduction

LWIP is a set of source codes opening TCP/IP protocol stack used in the embedded system, and developed by Adam Dunkels in Swedish Institute of Computer Science. LWIP is the Light IP protocol, it has the remarkable merit, can be moved whether there is a operating system.

The main body of LWIP protocol stack is the realization of major function for TCP/IP protocol. As shown in Figure 1, the LWIP protocol manage each major function modules with a duty supervisor[1].

2 Analyst of LWIP

Through the analysis to the sound codes of LWIP protocol stack, LWIP entire data transferring process as shown in Figure 2.

When the network card starts receiving the data, the application layer transfers the tcp_ write() function firstly, following the tcp_enqueue() function would be used to deal with the data which was transmited by upper function. The top_enqueue() function completes the following work: If the data length is excessively long, the tcp_enqueue() function will divide them, Meanwhile,put the divided TCP sectors into waiting list. After the completion of above work, the tcp_output() function is transferred to judge whether the receive buffer to be big enough,if the receive buffer is big enough, ip_route() and the ip_output_if() then transmit data. In the data transfer process, the data packet transmited to ip_input() function by the network interface level,the ip_input() function transmitted the TCP section to the tcp_input() function.the tcp_input() function will carry on the initial inspection to the these data,and judge which connection will the TCP section belongs to. The tcp_process function mainly completes the TCP essential phase transition. If the TCP section is being at the condition that waiting to receive network data, the tcp_receive() function will be transferred, then complete receiving the network data. If the TCP section is at the connection condition that waiting acknowledge confirm data, the tcp_output() function will be transferred[2].

3 Transplant LWIP into UC/OS-II

3.1 Brief Introduction of LWIP

Same to many other TCP/IP,LWIP is similarly designs TCP/IP with taking the lamination agreement as the reference. Each protocol was realized as a module, sometimes several functions also been provided as the protocol's entry point. Although some protocols are realized independently, but others are actually not like this, the goal to do like this is to promote the performance in the processing speed and memory occupancy rate. For instance, when confirms the TCP section's verification which has arrived, and decompose this TCP section, the TCP module must know origin and the goal IP address of this TCP section. If the TCP module knows the IP head's structure, then itself may withdraw this information and thus replace the way transmit the IP address message through the function. LWIP will possess the TCP/IP agreement to the same advancement, then the TCP/IP protocol stack is separated with the operating system essence. But the application layer procedure may either be the independent advancement or resident in the TCP/IP advancement. If the application procedure is the independent advancement, may communicating through communication mechanisms as news formation and operating system's mailbox and TCP/IP advancement. If the application layer procedure presence in the TCP/IP procedure, then application layer carry on communication with RAWAPI and the TCP/IP protocol stack..

3.2 Transplanting Process

Firstly, gaining the source code from the LWIP official website. In the LWIP source code, the operating system simulation level is some spatial

Received: 2010-5-25
Corresponding author: Xiaoqian Zheng（samzhengf@gmail.com）

connection function, needs to use the system call service which the operating system provides to realize. The author of LWIP has provided a more detailed explanation for the operating system simulation level, document named sys_arch.txt,is put in LWIP doc folder. For the operating system simulation level's transplant work is to enrich the operating system simulation level's connection function with some connection functions that provided by μC/OS-II, enables it to display the proper function. In addition, all the parts related with the compiler, the operating system, the hardware also needs to be revised, causing the transplanted LWIP can move smoothly. LWIP is the independent TCP/IP protocol stack, in the code, the function and the construction of data related with the operating system and the hardware has not been used, and when we need such function, it would be used through the operating system simulation level. The operating system simulation level provides set of unified connections for operating system service such as timer, the processing synchronization, the message passing mechanism and so on. In principle, when transplants LWIP to other operating system, only needs to realize operating system simulation level that suits this operating system, it including the following these functions:

sys_init()

//Initialization connection function

sys_arch_timeouts() //Timer connection function

sys_sem_new() //Foundtion signal quality

sys_sem_signal() //Sending signal quality

sys_arch_sem_wait() //Waiting signal quality

sys_sem_free() //Releasing signal quality

sys_mbox_t sys_mbox_new() //Creating new mailbox

sys_mbox_post() //Sending news connection function

sys_arch_mbox_fetch()

//Receiving news connection function

sys_mbox_free() //Releasing new mailbox

sys_thread_new()

//Creating thread connection function

Above function's realization, are basically according to μC/OS-II operating system's corresponding data structure, redefines data construction in these functions like sys_sem_t, sys_mbox_t and so on[3], then finally completed through sealing μC/OS-II operating system's corresponding system call function. Take connection function sys_sem_new() as the example, the realization as follows:

```
sys_sem_t sys_sem_new(u8_t count)
{
    sys_sem_t pSem；
    pSem = OSSemCreate((u16_t)count )；
    return pSem；
}
```

Creating function with this signal quantity in LWIP, is completed by using the foundation of function OSSemCreate() with sealing μC/OS-II operating system's signal quantity, and the used construction of data sys_sem_t is redefined as follows: typedef OS_EVENT* sys_sem_t; And construction of data OS_EVENT similarly owned by μC/OS-II operating system, the realization of other function is similar to this.

4 Design of Interface For LWIP Protocol

After completing the transplant of the operating system simulation level, the LWIP protocol stack's transplanting work had already completed half. The left is how to design the connect the LWIP protocol stack with the first floor network actuation. Using API which LWIP provides to carry on the network programming, firstly, LWIP protocol stack should be initialized. This paper completes the first floor network actuation with a network interface's initialization function the initialization, including increases and disposes the network interface of the first floor, builds the receive and transmission advancement for the first floor, creates TCP/IP advancement and so on. The LWIP first floor connection's initialization mainly completes functions for the data link layer and the physical level, shown like the dashed line frame's part in Figure 3.

5 The Realization of Driver

The major function of this network card actuation is initialize the various internal module register of the network card, the basic byte read-write function, data acceptance and transmits. Its working flow shown in Figure 4: Network card's driver has shielded the first floor hardware's processing details, has provided a connection which has nothing to do with the hardware for the upper formation software.DM9000 is going to reads the data packet that is waiting to transmit in the

input output buffer of the chip according to the form which has been assigned and starts the routing directive, by now DM9000 transformed the data packet to the physical frame form and being transmited on the physical channel automatically, otherwise physical signal DM9000 received will return it to data and input/output buffer according to the form which has been assigned for the application procedure uses. Its driver mainly includes initialization of the network card[4], the data transmission and the receive function, the above corresponding function would be completed by the following function:

void Init_Nie(int num)；

int Rec_packet(void)；

void Send_packet(struct_packet *Txd_data)；

6 Conclusion

Analyzing the basic constitution of LWIP from its internal structure's angle, and discussing the key technologies issue of the transplant from LWIP to μC/OS-II as well as the network actuation part, which causing that the transplant and applications related to LWIP becoming simpler and more convenient to complete.

7 Acknowledgment

This work was supported by the Graduates Innovation Foundation of Shandong University of Science and Technology (No. YCB100150).

References

[1] Leming Sun, The exploration and research of internal structure for embedded TCP/IP protocol stack LWIP,3rd ed,vol.10.Beijing:Technical Front,2008,pp.79-82.

[2] Haibo Jiao,Dukels.Design an implementation of the LwIP TCP/IP stack.

[3] Guofeng Zhao, Wenwu Ma,2008. Design and implementation of system for embedded network based on LWIP. The embedded networking applies,64(8-2):59-61.

[4] Hongjun Luo,Yuqiang Xie,Hui Shu,Youwei Zhang. Implementation of LWIP in μC/OS-II[J].Micro computer information,2005,7-2:46-47.

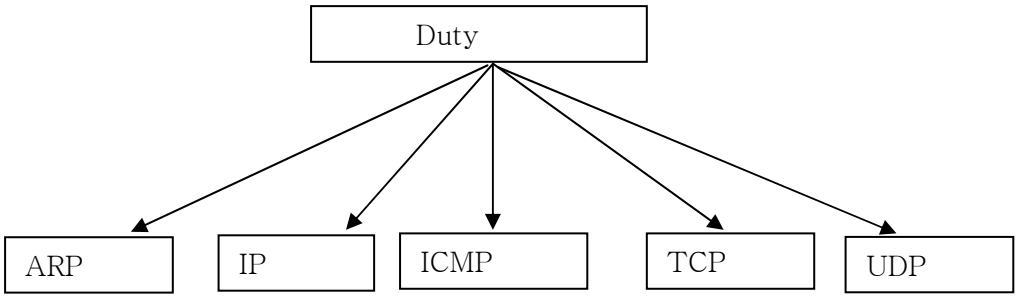

Fig 1.LWIP stucture diagram

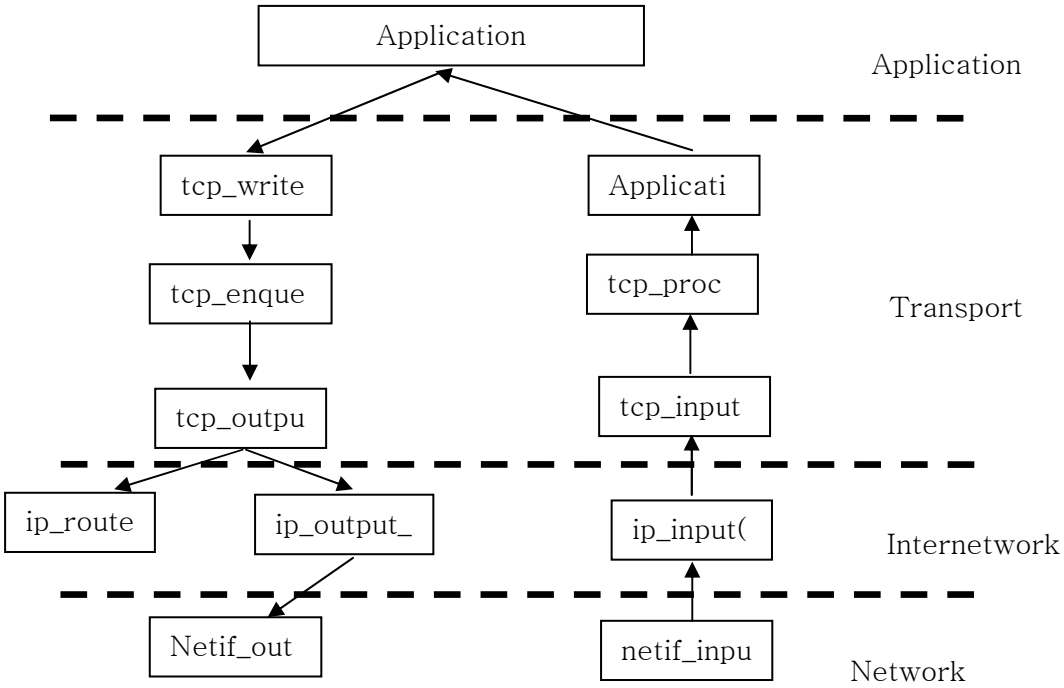

Fig 2. LWIP data process diagram

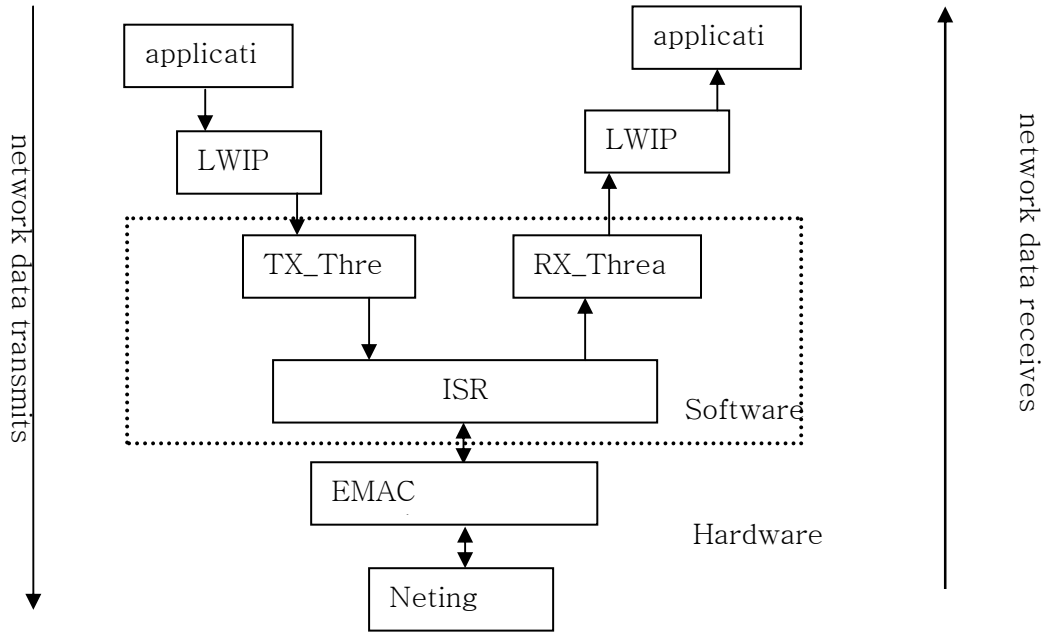

Fig 3. LWIP interface diagram

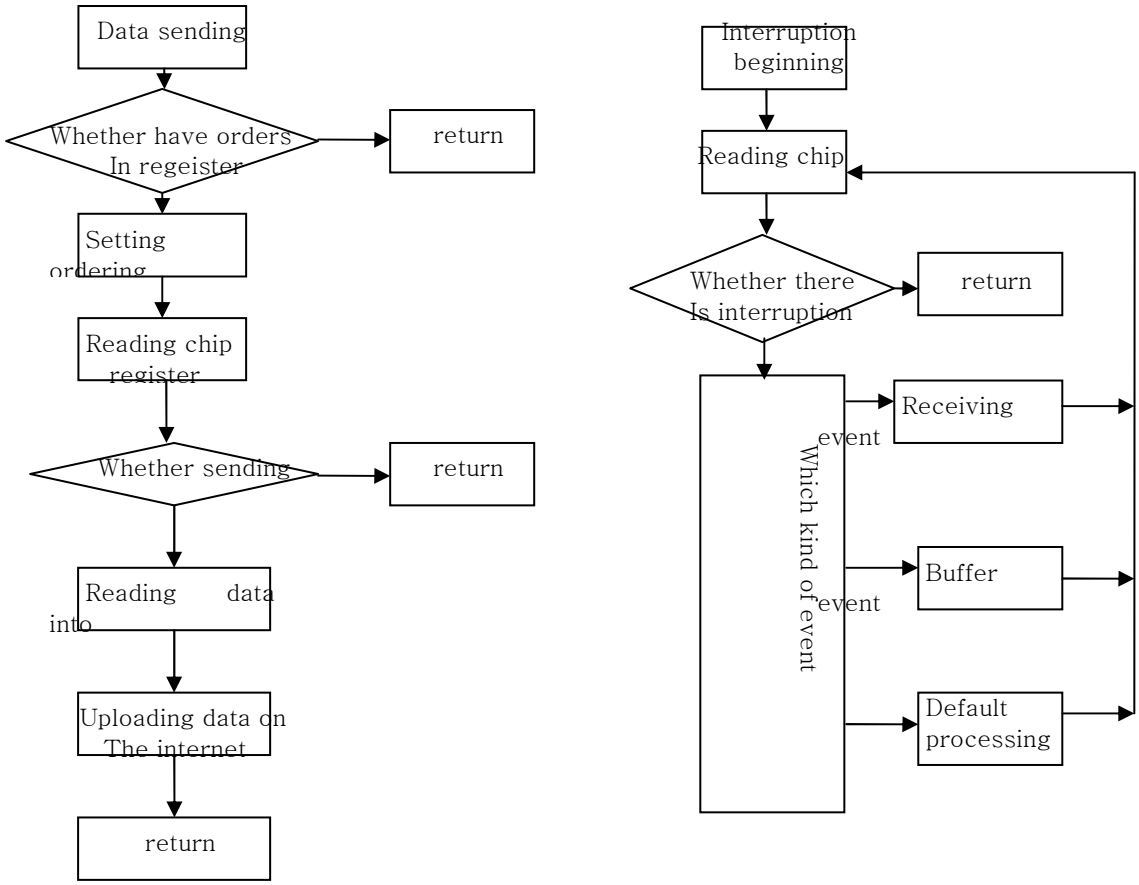

Fig 4. DM9000AE operation principle