The Design and Implementation of a General Reduced TCP/IP Protocol Stack for Embedded Web Server

Chong FU[1], Zhi-liang ZHU[2], Xiao-xing GAO[2] and Pei-rong WANG[1]

[1] School of Information Science and Engineering, Northeastern University,
Shenyang 110004, China
email: fu_chong@sohu.com
[2] School of Software, Northeastern University, Shenyang 110004, China, email: zzl@mail.neu.edu.cn

Abstract -- The embedded web server technology is the combination of embedded device and Internet technology, which provides a flexible remote device monitoring and management function based on Internet browser and it has become an advanced development trend of embedded technology. In this paper, a design and implementation scheme of a general reduced web server protocol stack which aims at the limited resources characteristic that the embedded system has was proposed based on AT90S8515 single chip that uses RISC technology and RTL-8019 network interface controller hardware platform. The overall design issue was thoroughly discussed and the implementation methods of core protocols in protocol stack such as 802.3, ARP, IP, TCP and HTTP were analyzed in detail.

I. INTRODUCTION

Embedded system is a special computer system that focuses on application, bases on computer technology, software and hardware customizable, suitable for the strict requirement of application system on function, reliability, cost, volume and power. It is widely used in military, civil electronics, household appliances and consumption electronic products in recent years. But most of the embedded systems are used independently at current stage, the communication protocols are relatively less and their coverage is limited, which cause it very difficult to perform flexible remote access and management. The Internet has become one of the most important basic information facilities in the world, the WWW service it offers has become one of the fastest growing and widest applied service, which have a great deal of advantages such as visualization, easy remote accessing, multi data format supporting, platform independent and thin client, etc. Connecting the embedded device to the Internet, implementing perfect Web service on it, and thus realizing a flexible remote monitoring and management through Internet browser has already become an inevitable development trend of embedded technology[1-3].

Although some embedded system such as Embedded Linux, VxWorks and WinCE had already provided TCP/IP protocol stack support, they are not suitable for applying in most of the low cost area due to their relative high hardware and software costs. So how to utilize the limited resource of the embedded system to design a reduced and application oriented protocol stack is the key technology of implementing an embedded web server. In this paper, the design and implementation process of a general reduced TCP/IP protocol stack was analyzed based on AT90S8515 single chip that uses RISC technology and RTL-8019 network interface controller hardware platform.

II. PROTOCOL STACK ARCHITECTURE DESIGN

It has significant differences of developing web server on embedded system from common machines. The embedded system is resource restricted, the single chip used in this paper has only 8kb programming space and 512 bytes data storage space, while any of the TCP, IP or HTTP protocol is too large to be implemented. The embedded system does not have enough storage resource to realize a completed protocol stack. So we must evaluate the protocols in standard TCP/IP protocol stack carefully to decide which parts are necessary in the stack and which parts can be saved, using different implementation method for different protocols. The following principles must be fully considered in protocol stack design: (1) follows the protocol hierarchy architecture, constructing clear interfaces between adjacent layers; (2) the content of the protocol must be reduced; (3) guarantee the reliability and security of the system.

If an embedded system can provide WWW service and uses Ethernet technology as its lower layer standards, then the protocol stack should at least contain: HTTP protocol in application layer responses for Web page request and response; TCP protocol in transport layer responses for reliable end to end message transmission; IP protocol in Internet layer responses for IP packet transmission; ARP protocol in Internet layer responses for the address resolution from IP to MAC and the NIC driver in the data link layer responses for controlling the operation of network adapter. Ping is the most commonly used network diagnosis tool, so the relational protocol ICMP should also be provided. From above discussion, a general reduced protocol stack for embedded web server can be designed as that shown in Fig.1.

The embedded web server is constructed by AT90S8515 single chip that uses RISC technology, RTL-8019 NIC and reduced protocol stack that composed of 802.3, ARP, IP, ICMP, TCP, and HTTP protocol. RTL-8019 is the most widely used Ethernet control chip, which provides the completed function of IEEE802.3 MAC sublayer for the reduced TCP/IP protocol stack. AT90S8515 single chip is the control heart of embedded server, the driver of RTL-8019 NIC and above protocols and all implemented in 8kb programming space.

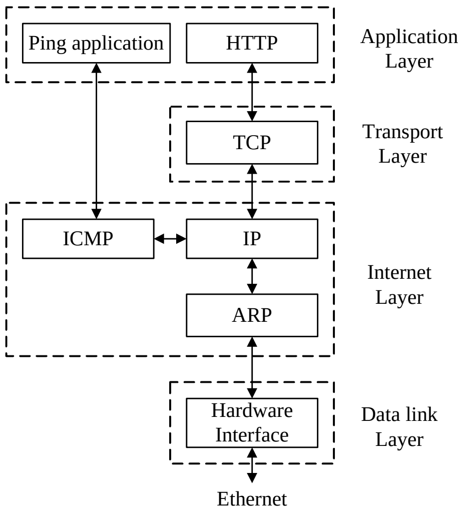


Fig.1 A general reduced TCP/IP protocol stack for embedded web server

III. The Implementation of Core Protocols in Protocol Stack

A. The Lower Layer IEEE 802.3 Protocol Support Based on RTL-8019 NIC

TCP/IP protocol stack does not define or even mention the specification of data link layer and physical layer. It only provides a Host-to-Network layer that allows the developer to select the lower layers standard follows the interface specification at their will. The Ethernet (IEEE 802.3) protocol is the most mature and widely used protocol in lower layer constructions. The inner double DMA channels and FIFO perform the frame sending and receiving function efficiently at 100Mbps. NIC encapsulates the data into frames according to RFC 849 standard[4] as shown in Fig.2.

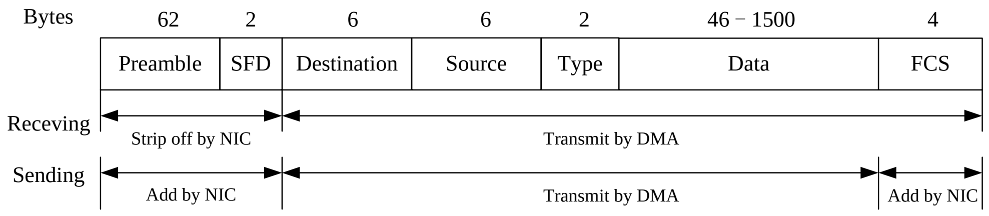


Fig.2 Ethernet frame structure defined by RFC 894

RTL-8019 NIC implements frame sending and receiving operation through 16 bits local and remote DMA channels. The frame sending and receiving function can be done flexibly by programming the inner registers of DMA controller. The data transmission principle of local and remote DMA channels is shown in Fig. 3. When the embedded web server have data to be send, the data in the main memory first be encapsulated into Ethernet frames and the remote DMA controller fetches them into the sending buffer ring inside NIC. The local DMA controller brings the frames in the buffer into FIFO where the frame is injected into cables. When NIC decides to accept a frame, the local DMA controller accepts it and buffers it into receiving buffer ring inside NIC, and then the remote DMA controller transmits it to system memory where it is further processed and passed to upper layer under the control of program, thus completing the frame receiving process.

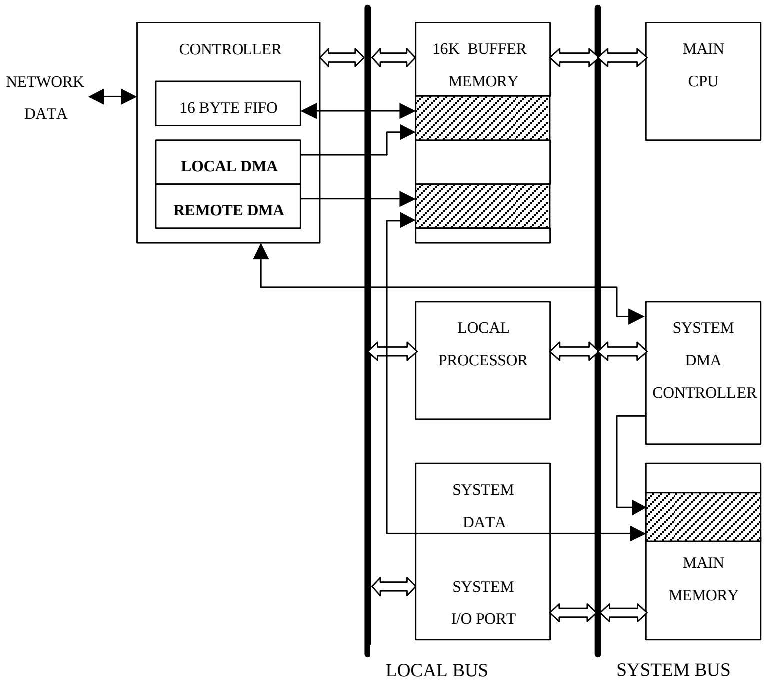


Fig.3 Data transmission principle of local and remote DMA channels

B. The Implementation of Reduced ARP protocol

ARP protocol provides a dynamic address mapping function from 32 bits Internet layer IP address to 48 bits data link layer MAC address. The request and response packet format of ARP used in Ethernet is shown in Fig. 4. ARP request is send as broadcast and all the machines in the subnet can receive it. When NIC receives a broadcast frame and confirms that it is an ARP request, it would further check whether the destination IP address is identical with itself, if it is, an ARP response will be send back, otherwise, the frame is just discarded.

Embedded web server usually works passively. It always receives the service request from the clients and sends back response but seldom send data frames to certain IP address forwardly. The Embedded web server needs not send ARP request to any other machines and what needs to do is only processing received ARP request and send back response, so only the response part of ARP need to be implemented in reduced protocol stack. If the storage space is sufficient, the ARP buffer can be constructed. Once the buffer is constructed, it would first seek in the buffer whether there is already an IP to MAC mapping when sending a frame, if it exists, the request-response process is just skipped.

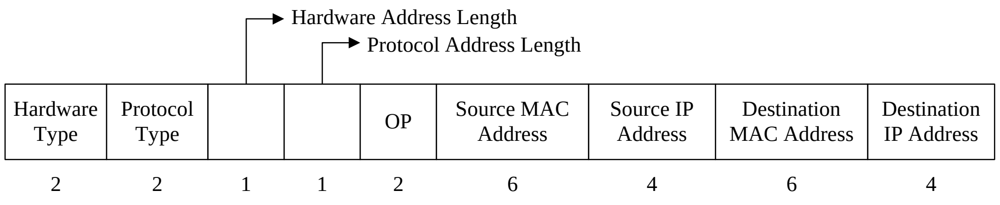


Fig.4 ARP request/response packet format used in Ethernet

C. The Implementation of Reduced IP protocol

IP is one of the core protocols of TCP/IP protocol stack, which provides unreliable connectionless datagram transmission service. All of the TCP, UDP, ICMP and IGMP messages are encapsulated in IP packet for transmission. Implementation a completed IP protocol is relatively complex, but for the special requirements of implementing an embedded web server, only the following two issues needs to be considered: (1) processing the received packets, delivering them to upper layers; (2) encapsulating TCP and ICMP messages into IP packets, delivering them to data link layer for further process. The above two functions are basic functions of IP protocol and the necessary functions the embedded web server must have.

When NIC receives a frame, it first checks the data type field in the frame header, the value 0x0800 indicates that the data field contains IP packet, then the data field is passed to Internet layer for further processing. IP protocol first checks if the destination IP address in the packet header is identical with local settings, if not, the packet just discarded, otherwise it examines the checksum field in packet header, after confirming that the packet was not damaged in transmission, stripping off IP header, delivering the data field in IP packet to upper layers for further processing. It is delivered to TCP or ICMP is based on the protocol field in IP packet. Besides, the IP protocol needs to listen whether the upper layer have data to be sent, if it have, it should encapsulate the data into IP packets and deliver then to data link layer.

D. The Implementation of Reduced TCP protocol

TCP is a transport layer service, which implements a connection oriented and reliable end-to-end byte stream transmission. It is a full-duplex service and each TCP connection supports a pair of byte stream, each stream a direction. It contains a flow control mechanism for each stream that allows the receiver to restrict the amount of data that the sender sends in a certain time interval and allows multiple applications on any machines exchanging with its own peer entities simultaneously. When establishing a TCP connection, the client executes an open command to the server initiatively, after a connection established successfully, the two sides begin to transmit messages. Similarly, when one side finishes data transmission, it will close the connection on its side.

Generally, TCP determines its connection state based on state transition map[5]. This means we must maintain an independent state machine for each TCP connection. But for embedded system, the RAM space is insufficient to do so. To solve this problem, notice that embedded web server always responses the client's request, it never connects other machines forwardly. So the TCP protocol can be largely reduced with the precondition that the embedded web server just needs response to TCP request correctly. The TCP state is "LISTEN" when there is no request. When receiving a TCP connection request, the web server executes open operation passively, sending "SYN/ACK", and then changing state to "SYN_RCVD". When receiving correct "ACK", the establishing connection process is finished, the states of two sides all changes to "ESTABLISHED". From above analysis we can see, only the "LISTEN" and "SYN_RCVD" states need to be provided for the web server during the connection establishing period. Although the "CLOSED" and "SYN_SENT" states are absence, the "LISTEN" and "SYN_RCVD" states are enough to describe the state when the web server establishes connection. The "FIN" state is set when the web server sends the last packet to release the current connection. The active or simultaneous close may happen, so only the states related to active and simultaneous close need to be provided for the web server.

TCP state map maintenance is very import for implementation of complex TCP protocol. According to the characteristics of embedded web server, discarding the state that would not happen and implementing any possible state changing are important rules of maintaining TCP state. The maintenance of two states used by the embedded web server

TCP state machine when establishing connection is shown in Fig.5, other state maintenances are similar.

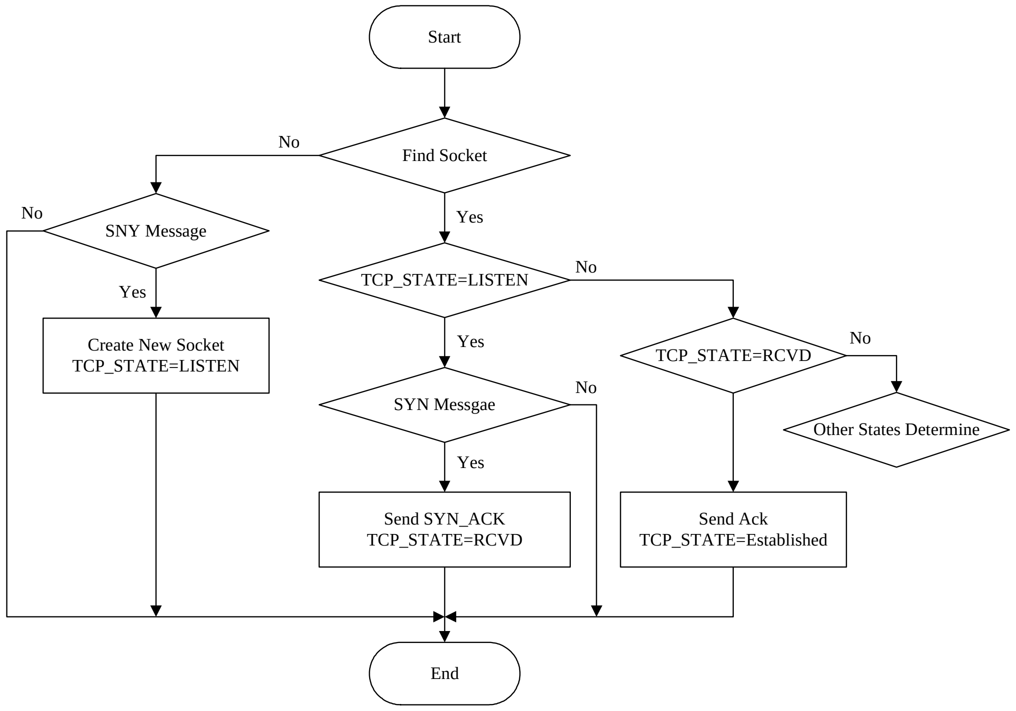


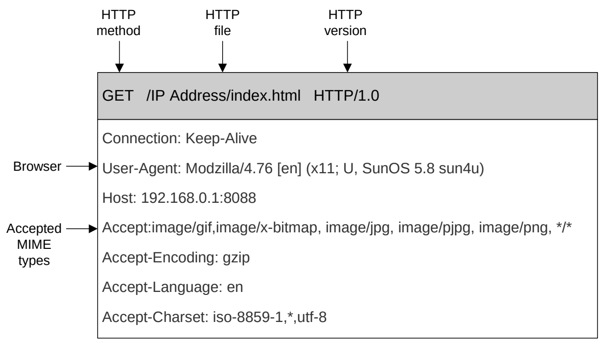
Fig.5 The state transition diagram of connection establishing process

E. The Implementation of Reduced HTTP protocol

HTTP protocol is the foundation of WWW technology. The web client (browser) uses one or more TCP connection communicates with web server through port 80. The client process establishes TCP connection with web server and then sends requests and reads the response from the server. The connection close operation of web server indicates the end of response. There are two types of HTTP1.0 message: request and response, as shown in Fig.6 (a) and (b).

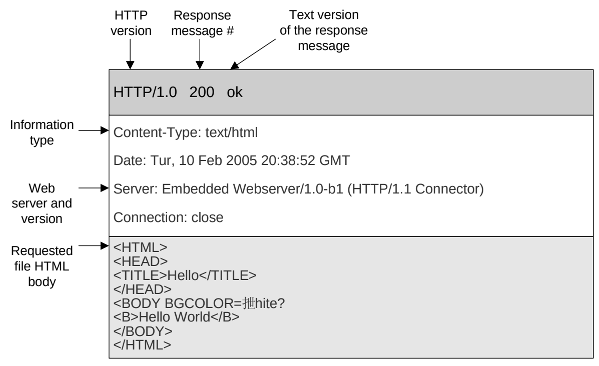
HTTP uses request-response mechanism to fetch web pages on web server. HTTP1.0 supports the following three kinds of request: (1) GET request; (2) HEAD request; (3) POST request. Browser sends HTTP request to web server using multiple line string format. The first line is command line with parameters follows it. In order to reduce the storage space usage to minimum, the web server does not care the browser type information and even the HTTP version is not necessary because the limited resource could not offer a read-write log. So the portion after file name can be discarded while the main function of the web server would not be affected.

The HTTP response message is constructed as Fig.6 (b). The first line is status line that starts with HTTP version, 3 numbers follows it representing response message code, and the last is the readable response phrase. The next part is the information type and web server information and the last part is the HTML body of the requested web page. When the web server receives a HTTP request, it first analyzes the request information and then passes the required file location to TCP protocol for processing. The request web page files are stored in the outer storage space, the data that the HTTP protocol passes to TCP protocol is not HTTP message (web page) but the location where the web page file at. Neither TCP nor IP reads HTTP messages to data storage space until NIC transmits the frames, then the web page data are read into NIC buffer according to file pointer.



```
GET   /IP Address/index.html   HTTP/1.0
Connection: Keep-Alive
User-Agent: Modzilla/4.76 [en] (x11; U, SunOS 5.8 sun4u)
Host: 192.168.0.1:8088
Accept:image/gif,image/x-bitmap, image/jpg, image/pjpg, image/png, */*
Accept-Encoding: gzip
Accept-Language: en
Accept-Charset: iso-8859-1,*,utf-8
```

(a)



```
HTTP/1.0   200   ok
Content-Type: text/html
Date: Tur, 10 Feb 2005 20:38:52 GMT
Server: Embedded Webserver/1.0-b1 (HTTP/1.1 Connector)
Connection: close
<HTML>
<HEAD>
<TITLE>Hello</TITLE>
</HEAD>
<BODY BGCOLOR=拙hite?
<B>Hello World</B>
</BODY>
</HTML>
```

(b)

Fig. 6 (a) HTTP request message format, (b) HTTP response message format

IV. CONCLUSIONS

The advantages of shifting traditional scheme to “using embedded web application as center” are the low cost of thin client, visualization, platform independent, flexible deployment, excellent remote accessing and troubleshooting ability, etc. The equipments can be configured and controlled flexibly in web pages through CGI interface on embedded web server. In industry control field, the using of embedded web server on intelligence device, instrument and sensor to realize flexible remote control has very high theoretical and application value.

V. REFERENCES

[1] Hong J W, Kong J Y, Yun T H, *et al*. “Web-based intranet services and network management,” *IEEE Communications Magazine*, vol.35, no.10, Oct, 1997, pp. 100–110.

[2] Ju H T, Choi M J, Hong J W. “Ews-based management application interface and int–egration mechanisms for web-based element management,” *Journal of Network and Systems Management*, vol.9, no.1, Jan, 2001, pp. 31–50.

[3] Ju H T, Choi M J, Hong J W. “An efficient and lightweight embedded web server for web-based network element management,” *International Journal of Network Management*, vol.10, no.5, May, 2000, pp. 261–275.

[4] Douglas E. Comer, *Computer Networks and Internets*, New York: 1998, pp. 76–78.

[5] Andrew S. Tanenbaum, *Computer Networks*, New York: 1999, pp. 532–535.