A Tiny Web Server on Embedded System

Lain-Jinn Hwang*, Tang-Hsun Tu†, Tzu-Hao Lin‡, I-Ting Kuo§, and Bing-Hong Wang¶
Department of Computer Science and Information Engineering
Tamkang University, Tamshui, 251, Taipei, Taiwan
**E-mail: micro@mail.tku.edu.tw*
†*E-mail: war3_515@hotmail.com*
‡*E-mail: singy000@msn.com*
§*E-mail: yiting2002@hotmail.com*

Abstract

Bhttpd (BBS httpd) is a tiny web server for embedded system, the entire size it consumed is 14KB only. Bhttpd is HTTP/1.1 compliance, GET/POST method, keep-alive connection and flow control support. For computer programmer, Bhttpd is easily for reuse because of its structure is simple. In this paper, we introduce the means of bundling Bhttpd to make a new web page, by just adding the required functions. This makes the static web page development on embedded system not only simple but also efficient than other web server.

Keyword – Bhttpd, embedded system, flow control, web server.

1 Introduction

The first web server was built by the British physicist Tim Berners-Lee at CERN, the European Centre for Nuclear Research at Geneva, Switzerland. Today, several web servers had been developed. Up to now, the common used web server is Apache and it has served nearly 60% of all websites on the Internet [1]. Except for Apache server, the IIS (Internet Information Services) is second popular web server that is developed by Microsoft. Even Apache and IIS are widely used on web server market, with the development of embedded system day by day, the usage of web server will come to the family.

On normal conditions, we may usually select Apache [2] as web server because of it is powerful. Of cause, there are always some exceptions. Because of hardware resource on embedded system is limited, for instance, SBC-2410x [3] has only 64MB RAM and 100MIPS and SiS550 [4] has 128MB RAM and 400MIPS. When we need to build a web server on these enviroment we have to retrench the size and simplify unnecessary functionality [5]. Usually, the web server built on embedded system is for personal usage. For example, we can configure the router or switch network setting by browser because there is web server built on it, and we may not often change the setting. Briefly, what we need to used on embedded system is a light, small and fit web server.

Usually, web server mean two things, one is a computer that responsible for accepting HTTP requests from clients, which are known as web browser such as Microsoft Internet Explorer [6] or Mozilla Firefox [7]; the other one is a computer program that provides function corresponding to all the client requests. The latter is mainly issue we discuss here.

This paper is organized as follows. Section 2 describes the different web server model and its type. Section 3 describes the example of bundling web server required software, and how to make web server more efficient. Section 4 we compare Bhttpd with different tiny web servers for experiment result and section 5 is our conclusion.

2 Web Server Introduce

To a webmaster, there is nothing important than stability and efficiency. When a request is asked from client, web server must reply it with corresponding action, and the time during a server reply to client we call it response time. When there are many people ask for request at the same time, it is obvious to know how that how stable the server is. Especially, response time is affected by how web server

is implemented. We can refer the way a web server is implemented to several models, and we introduce in next section.

2.1 Server Model

Generally, there are four kinds of server model, and each model has their own advantage on speed, file size or efficiency.

1. Finite State Machine Servers

   To maximize the scalability, many small web servers are implemented as a single process and a finite state machine. Every task is split into two or more small steps that are executed as needed as Figure 1. When a request comes, the web server will analyze what client asks for and lets application handle function responds to their request, just like finite state machine. To personal web server, because most of requests are for static web pages, response time can be shortening without redundant context switches. Besides, by keeping the state of each connection and asynchronous I/O, it is possible to implement ultra fast web servers, at least for serving static content such as static web pages and graphic files. In addition, this kind of model holds the minimum memory size rather than other model.

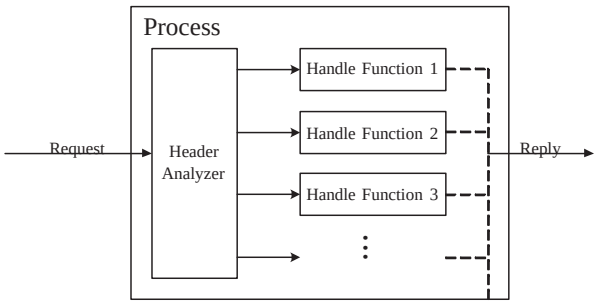

Figure 1: Single Process Implement

2. Thread-based Servers

   Except for the single process, multithread is another common used model. Multithread means that inside each server's process, at least one threads are run, and each one able to execute its own task and the local variable independently from the others. After thread finish its task, the memory it occupied would be free. When a user visits the web site, web server will use a thread to serve the page to that user. If another user visits the site while the previous user is still being served, the web server can serve the next visitor by using a different thread. Thus, the next user does not have to wait for the first visitor to be served. It is very important because not all the user have the same speed Internet connection. The slower should not delay other visitors from downloading a web page. Hence, threads are often used to serve dynamic content for better performance because of the large number of discrete objects. This model is illustrated as Figure 2.

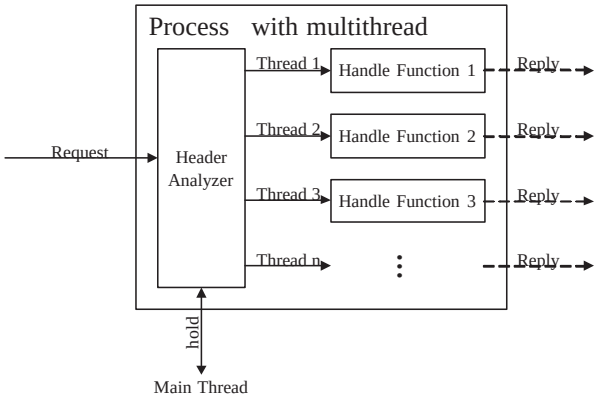

Figure 2: Process with Multithread Implement

3. Process-based Servers

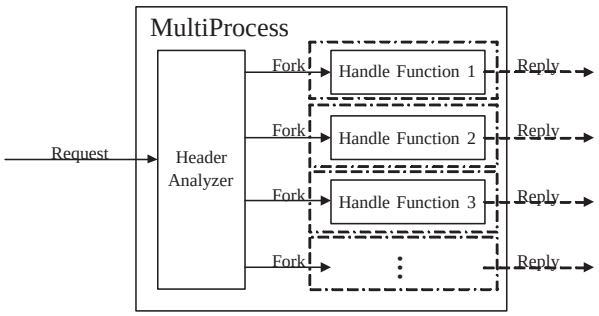

Figure 3: Multiple Process Implement

For reliability and security reasons, some web servers use multiple processes rather than multiple threads within a single process. Usually, a pool of processes is used until a certain threshold of requests has been served by process before replaced by a new one. Because threads share a main process context such as global variable, if a thread crash its own task and the error variable may jam other thread, this may easily crash the whole application, and a buffer overflow can have disastrous consequences. Moreover, memory leak is usually out of the control for third party libraries such as PHP, the application programmer cannot deal with it using threads, but can use a pool of

processes with a limited lifetime because of OS automatically frees all the allocated memory when process dies. Furthermore, global variable addresses in a process also differ from each process, even if one process has crash, the others do not be interfered. Multiprocess illustrated as Figure 3.

4. Mixed Module Servers
   To integrate all advantages of the former model, many web servers implement a mixture of all these programming techniques such that serving client according to the different task. For example, static web page implement with single process and thread, . For serving static or dynamic content, etc.).

2.2 Application

Actually, we can only classify it according to the function at most. Here, we refer the web server to two types, the full-featured and tiny web server. We can clearly know what the full-featured server is by its name, this type web servers support a lot of plugin for third party program and function about Internet optimization setting. Relative to full-featured server, tiny web server is light and simple, and some of them only support the basic HTTP 1.1 even because for the embedded support. The following tiny web servers are often used on embedded system:

1. Thttpd [8]
   According to official website declaration, it is a tiny, turbo and throttling HTTP server. It handles only the minimum necessary to implement HTTP/1.1, and has a very small run-time size, since it does not fork and is very careful about memory allocation. In typical use, it is about as fast as the best full-featured server such as Apache. Under extreme load, it is much faster.
2. Corehttp [9]
   Corehttp is designed to be a minimalist http server focusing on speed and size. Similarly, asynchronous I/O handling, single-process select client concurrency model without forking and thread, it is easy for portability and efficiency. Besides, many configuration options cannot slow the program down.
3. Boa [10]
   Boa is a single-threaded HTTP server. Unlike traditional web servers, it does not fork for each incoming connection, nor does it fork many copies of itself to handle multiple connections, and forks only for CGI programs, so the load on the server never exceeded over 1%. The design goal of Boa is security, portability and speed, and is not intended as a full-feature server.
4. LightTPD [11]
   LightTPD is a security, speed and flexibility web server. As its name, it is designed for high performance environments. Compared to other web servers, there are effective management of the CPU-load and advanced feature such as output-compression, URL-rewriting and fast CGI with a small memory usage.
5. Shttpd [12]
   Shttpd is simple web server. The main design goals are the ease of use and the ability to embed. Ideal for personal use, web-based software demos (like PHP, Perl etc), quick file sharing.

Although most of web servers are used on PC now, tiny web server would become more and more practical and common on embedded system for widely application increasingly. In next paragraph, we give an example that how to implement the web server more efficiently.

3 BBS httpd Web Server (Bhttpd)

We have introduced some tiny web server before, and those web servers are famous as light, simple and tiny for embedded system use. In fact, except for those feature, there is still a key point. Usually, when clients ask for request, web server just takes the content by path. If we can embed the web page data in source code, it would reduce the loading time due to I/O buffer. Maybe it is strange and seldom to do as this way because of it is hardly to modify the most of web server's source code. We design a web server that callled "BBS Httpd (Bhttpd)" is not only easy to be embedded web page in its function, but also can bundle web server with application [13], [14]. Originally, this web server is written for web BBS use, so we call it "Bhttpd".

3.1 Feature

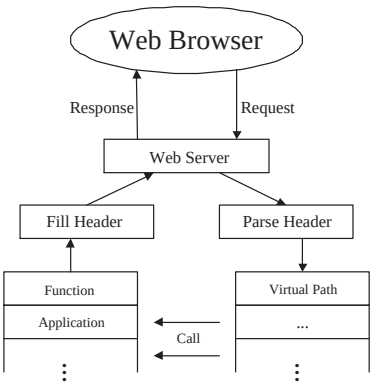

Figure 4: Process of Bhttpd

As another single-task web server, Bhttpd is also a small web server. The entire size it consumed is only 14KB with flow control, keep-alive and HTTP/1.1 supporting. Besides, it is easy and convenient to add a new web page as a function to source code as Figure 4, and it is important because this make web developments on embedded system simpler. However, we still have to know how the function works before using.

3.2 Instance of Bhttpd Server

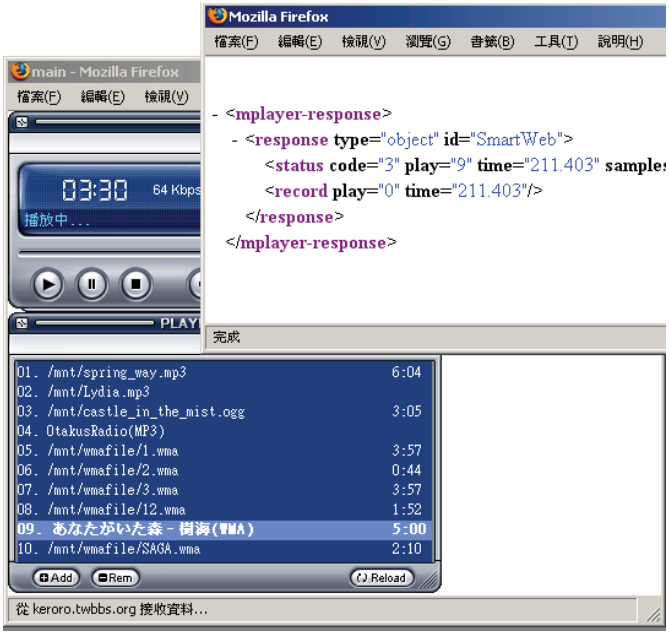

Figure 5: Web-Based Interactive Interface

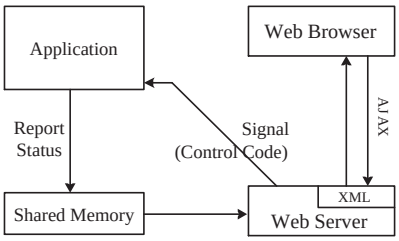

Figure 6: Implementation of Bhttpd

How Bhttpd differs from others is that it is mainly aimed for easy rewrites. This feature is an advantage especially on embedded system. For example, if we want to design an web-based interface for media player such that user can control the play, pause and stop message by browser. Besides, the detail as time, sample rate, and CPU utilization could be refreshed, how can we implement for efficiency and speed ? As Figure 5, we add Bhttpd to Mplayer [15] for the interactive web-based interface. When browser request for the web page by AJAX technique [16], web server read the playing status in shared memory and return that message directly. When web browser tries to connect to server first time, server would send web page data which includes the graphic or playlist URL back. Except for those URL, there is some control code be used to communicate with web server. In fact, the control code just only asks for response repetitively. Every time web server receives a request, it read the playing status in shared memory and has the message return in XML format. This is known as AJAX technique today. The completed process is illustrated as Figure 6.

In general, it is a troublesome method because you have to rewrite part of source code for new function. However, when this method is applied on embedded system, it would be more practical and useful since different embedded system devices have their own demand for web-based interactive, if we can modify the Bhttpd standard source code easily, it could become an advantage instead. Besides, we can embed the function on web pages in Bhttpd web server whether the server is applied on embedded system or not, the speed of data access must faster than I/O buffer reading. Therefore, if there are a large number of client request such as previous instance, the response time would be shorter and required data would be smaller when we combine the method with AJAX technique especially.

4 Experiment Result

Before compare with other web server, we introduce the hardware platform. The enviroment we used is SBC-2410x embedded system as shown in Table 1.

Platform	SBC-2410x
CPU	S3C2410@200MHz 100MIPS
Memory	64MB SDRAM 64MB NAND Flash 1MB NOR Flash

Table 1: Hardware Platform

For objectivity, we compare Thttpd, Boa, Corehttp, LightTPD and Shttpd with Bhttpd because both of their features are similar. We can understand this point by the server model, all the web server is implemented with single-task program for the speed and size.

As Table 2, except for Corehttp, it is obvious that Bhttpd has the smallest memory size at run-time and thttpd is biggest

Software			Feature		Size		
Server	**Version**	**Release**	**Model**	**Flow Control**	**Code (B)**	**Modules (B)**	**Memory (KB)**
Thttpd	2.21b	02/Jun/2006	select	Yes	1,748,716	X	1,856
Core http	0.5.3a	05/Aug/2005	select	No	18,179	X	284
Boa	0.94.13	23/Feb/2005	select	No	57,143	X	428
Lighttpd	1.4.11	09/Mar/2006	select	No	139,760	352,380*	632
Shttpd	1.35	07/Arp/2006	select	No	68,716	X	556
Bhttp	1.3b	01/Aug/2006	select	Yes	14,394†	X	240†

* Runing server needs load modules.
† Smallest size.

Table 2: Feature Comparison

Server	**Concurrency**	**Keep-Alive (KA)**	**RPS** KA / Non-KA	**Mean Time-Per-Request(ms)** KA / Non-KA
Thttpd	1	Y/N	593.14 / 368.98	1.686 / 2.710
	2		754.42 / 440.54	2.651 / 4.540
	3		911.02 / 470.06‡	3.293 / 6.382§
Core httpd	1	N	Non / 309.09	Non / 3.235
	2		Non / 360.70	Non / 5.545
	3		Non / 357.64	Non / 8.388
Boa	1	Y/N	606.54 / 351.92	1.649 / 2.842
	2		727.96 / 384.15	2.747 / 5.206
	3		759.73 / 405.83	3.949 / 7.392
Lighttpd	1	Y/N	505.08 / 352.64	1.980 / 2.836
	2		634.83 / 418.12	3.150 / 4.783
	3		656.01 / 416.73	4.573 / 7.199
Shttpd	1	N	Non / 183.40	Non / 5.452
	2		Non / 198.12	Non / 10.095
	3		Non / 201.64	Non / 14.878
Bhttpd	1	Y/N	654.03 ‡/ 384.57‡	1.529§ / 2.600§
	2		960.31‡ / 443.52‡	2.083§ / 4.509§
	3		981.24‡ / 453.42	3.057§ / 6.616

‡ Largest request per second (RPS) .
§ Least mean time per request.

Table 3: Efficiency Comparison

maybe it is bundled with PHP module. Both of them have the flow control function, but Bhttpd is much smaller than thttpd. Here we do not mention other feature such as authentication because there is no need for personal use. The maximum user number is also unmeaning if we do not act after connecting to server. Therefore, we use Apache HTTP server benchmark tool [2] to measure the efficiency difference between them [17], [18].

As Table 3, we compare the Request-Per-Second (RPS) and the mean Time-Per-Request with keep-alive or not. The more RPS has represents the server could accept more request at the same time. In here, Bhttpd is still the most one. From mean Time-Per-Request (TPR), we can know the average time of server response to clients. As previous, Bhttpd is the smallest one. All the data we test are based on static web pages.

5 Conclusion

According to the comparison, Bhttpd is really a superior tiny web server. Not only the response time and size, as well as the feature for rewriting source code are its advantage. For the use of embedded system, it could be implemented as part of system, and this does make the web server required development easier. Even on general purpose, advantage of its rewrite could make being embedded in application easier. In other words, it is not restricted to any hardware platform. Anyway, Bhttpd is certainly different from other tiny web server.

References

[1] "Netcraft." http://news.netcraft.com.

[2] "The Apache Software Foundation." http://www.apache.org.

[3] "Samsung-electronics." http://www.samsung.com/tw.

[4] "SiS: Silicon Integrated Systems Corp." http://www.sis.com.

[5] M.-J. Choi, H.-T. Ju, H.-J. Cha, S.-H. Kim, and J. W.-K. Hong, "An efficient embedded Web server for Web-based network element management," in *IEEE/IFIP Network Operations and Management Symposium*, pp. 187–200, Apr 2000.

[6] "Microsoft Corporation." http://www.microsoft.com.

[7] "Mozilla Firefox." http://www.mozilla.com/firefox.

[8] "THTTPD - tiny/turbo/throttling HTTP server." http://www.acme.com/software/thttpd.

[9] "CoreHTTP web server." http://corehttp.sourceforge.net.

[10] "Boa Webserver." http://www.boa.org.

[11] "LightTPD: Light HTTPD." http://www.lighttpd.net.

[12] "SHTTPD: Simple HTTPD." http://shttpd.sourceforge.net.

[13] T. Berners-Lee, R. Fielding, and H. Frystyk, "Hypertext Transfer Protocol – HTTP/1.0." http://www.faqs.org/rfcs, May. 1996. RFC 1945.

[14] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, and L. Masinter, "Hypertext Transfer Protocol – HTTP/1.1." http://www.faqs.org/rfcs, Jun. 1999. RFC 2616.

[15] "Mplayer: The Movie Player." http://www.mplayerhq.hu.

[16] B. McLaughlin, *Head Rush Ajax*. O'Reilly and Associates, first ed., Mar 2006.

[17] J. Hu, S. Mungee, and D. Schmidt, "Techniques for developing and measuring high performance Web servers over high speed networks," in *IEEE INFOCOM Computer and Communications Societies*, pp. 1222–1231, Mar 1998.

[18] J. Hu, I. Pyarali, and D. Schmidt, "Measuring the impact of event dispatching and concurrency models on Web server performance over high-speed networks," in *IEEE GLOBECOM Global Telecommunications Conference*, pp. 1924–1931, Nov 1997.