Server Program Analysis Based on HTTP Protocol

Min CHE[1] Ming Fu TUO[1]
[1]*College of Science, Air Force Engineering University 710051 Xi'an, China*

Abstract—The overwhelming majority of Web developments are built based on Web application of HTTP protocol. To analyze the server program based on the HTTP protocol and the process of interaction between the browser and server, a simple HTTP-based server program is taken as an example to describe how to use the theories of multi-threading and asynchronous operation，and the core section of HTTP-based server program is gradually resolved with Winsock programming. This method can achieve the purpose to avoid blocking the calling thread，and improve the responsiveness of a server program.

Keywords- HTTP; HTTP protocol; Server; WinSock

1 INTRODUCTION

Hypertext Transfer Protocol (HTTP) is the protocol of web's application layer. It is the web's core issue. HTTP can be realized in the web client's program or server's program. The programs running on different end systems of client and server can exchange with each other through exchanging HTTP message. Web Page is also called document composed of various objects. Object is a document which is only single URL addressable. It might be an HTML file, or JPG image, or GIF image, or JAVA applets, or voice clip, etc. Usually, most web pages consist of one basic HIML file and a number of the referenced objects. HTTP protocol defines how to request a web page from the client (web browser) and how to transfer the web pages to client from web server.

2 HTTP PROTOCOL

HTTP is a protocol of application layer basing on stateless, request and response model. Usually it is based on TCP connections. The vast majority of web developers are the web applications on the basis of HTTP protocol. A lightweight web server is accomplished via the program in this study [1].

2.1. HTTP request consists of three parts namely consumer request, line interest header, and request body.

Request line begins with a method symbol separated by a space, then followed by the Request-URI and the version of HTTP. The form is defined as follows: Method Request-URI HTTP-Version CRLF. Method represents request method; Request-URI is a uniform resource identifier; HTTP-Version indicates the request version of HTTP protocol; CRLF represents carriage return and line feed. Except the CRLF at the end, CRLF cannot be separated into CR or LF characters in other part [2].

2.2. HTTP response consists of three components namely the status line,message header,and response body .

The format of status line is: HTTP-Version Status-Code Reason-Phrase CRLF. HTTP-Version indicates the version of HTTP protocol used in web server; Status-Code indicates the status code of a response sent back from web server; Reason-Phrase indicates the text description of status code. The status code is made up of three digits. The first digit defines the categories and there are five possible values:

1xx: Indication Information It indicates that the request has been received and processing was continued;

2xx: Success It indicates that the request has been successfully received, understood, and accepted;

3xx: Redirection It indicates that further operating is needed to complete the request;

4xx: Client Error It indicates that the request has a syntax error or cannot be achieved;

5xx: Server Error It indicates that the server failed to achieve a legitimate request

Common status code, status description, explanation:[3]

200 OK //successful client requests

400 Bad Request//Syntax error in the client request. It cannot be understood by the server

HTTP message is composed of requests from the client to the server and responses from server to client. Request and response messages consist of start line (For request message, the start line is the request line; For response message, the start line is the status line), message header (optional), blank lines (only CRLF line), and message body (optional).

3 HTTPSVR PROGRAM ANALYSIS

HTTPSVR is a simple web server with a graphical interface, you can specify the port and the main directory of web page.You can generate web accessing log. It is constituted by an endless loop: Receive the request from client, parse and process the request according to the HTTP protocol, then sent the response back to the client.

3.1. HTTPSVR main work flow chart

HTTPSVR is a powerful software with rich graphical interface. The main workflow is shown in Figure 1.

Firstly ClistenSocket: OnAccept (int nError-Code) function generates CRequestSocket class. When monitoring the connection requests, Accept function creates a new socket pRequest and returns to the handle of sentence. AsyncSelect function monitors two events of FD_READ and FD_ CLOSE in port 80. When an incoming event is FD_READ, the receiver is prepared and OnReceive ( ) function is triggered. When an incoming event is FD_WRIT, OnSend( )function will be triggered and the data is sent.

Void Crequest Socket::On Receive (int nError Code) function of ReqSock.cpp will be executed when data arrive. The code int nBytes = Receive (m_buf, GetData (), m_buf, GetSize()) store the data packets uploaded from tcp layer in the m_buf buffers of request and response.

Next, use the switch statement to process three kinds of request status (REQ_REQUEST, REQ_HEADER, and REQ_BODY). When requested state of m_reqStatus == REQ_DONE, call Star-tResponse ( ) to began constructing the response message.

When the response message constructing with Start Response ( ) is finished, Async Select (FD_WRITE | FD_CLOSE) is used. The void Crequest Socket :: OnSend (int nError Code) function is called to send the data in buffer m_buf [4].

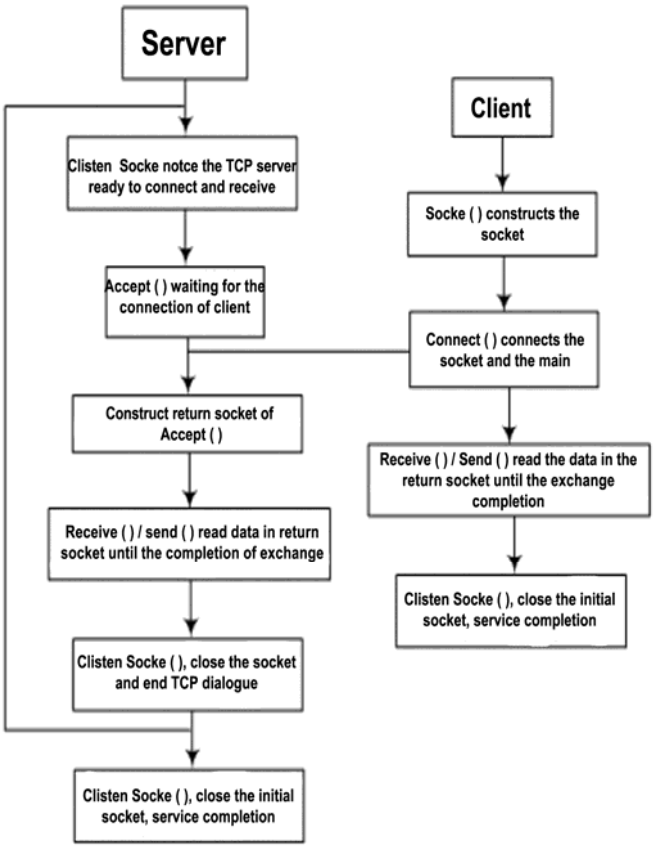

Figure 1. The main workflow of HTTPSVR

3.2. Asynchronous non-blocking CAsyncSocke

Create () function of CasyncSocket In addition to creating a Socket, WSAAsyncSelect ( ) is used to associate this socket with the window object so that the object can handle the events (message) from the socket. However, after CSocketWnd receives the event of Socket, it just calls back the virtual functions such as Casync Socket:: On Receive () Casync Socket :: On Send (), CasyncSocket :: OnAccept (), Casync Socket :: On Connect (), etc. So CAsyncSocket derived class only needs to add code or send code in these virtual functions. When using CasyncSocket, all networks IO are asynchronous operation if you use Create to create socket by default. The network data transfer needs the following functions: OnAccept, OnClose, OnConnect, OnOut-OfBandData, OnReceive, OnSen [5].

3.3. HTTPSVR KEY CATEGORY ANALYSIS

Void CListenSocket:: OnAccept (int nErrorCode) Create a new CRequestSocket and monitor port 80. Return to a socket when requests arrive. Through AsyncSelect (FD_READ | FD_CLOSE), the void CRequestSocket :: OnReceive (int nErrorCode) function is called to receive data.

Void CrequestSocket :: OnReceive (int nError-Code) All major features of httpsvr are achieved including accepting request from web browser, analyzing the request packet according to http protocol, constructing the response message, and sending the processed data packets back to the browser. Receive (m_buf, GetData ( ), m_buf, GetSize ( )) function receives the request packet from browser. Switch statement processes it depending on the state. When the browse's request status m_reqStatus is REQ_REQUEST, process with a while loop. Process Line ( ) function parses the each line of received packets and then initialize m_pRequest class. After the swich statement is finished, StartResponse ( ) is called according to the parsing results of m_pRequest to construct the response packet. Finally, the function AsyncSelect (FD_WRITE | FD_CLOSE) recalls the function OnSend to sent.

void CRequestSocket :: OnSend (int nErrorCode) call int nBytes = Send (m_buf. GetData ( ), m_cbOut) function to send the data in the cache and process the possible errors by closing or re-transmitting.

void CRequestSocket :: ProcessLine (void) conducts processing according to different states of m_reqStatus to complete the initialization of CRequest * m_pRequest.

BOOL CRequestSocket :: StartResponse (void) [6] This function is used to construct the response message. It saves the constructed response packet in the cach of m_buf and sends m_buf using void CRequestSocket :: OnSend (int nErrorCode). It can construct different response packet according to different Method (GET, HEAD, POST). For example, when the Method is GET, it indicates that the browser needs to browse the web page. The FindTarget (strFile) is started to search in the default directory. If it exists, the packet is open and the http protocol requiring head is constructed using StuffHeading ( ). Then using StartTargetStuff, the contents of opened web page is put into m_buf and waiting to be sent. After the construction is completed, return to BOOL CrequestSocket::StartRes ponse (void) and then call AsyncSelect (FD_WRITE | FD_CLOSE) function. This function can call function of void CrequestSocket :: OnSend (int nErrorCode) and send the data packs in m_buf catch.

BOOL CRequestSocket :: FindTarget (CString & strFile) Because the requests sent from browser are usually the network address format, so the function changes the network address format files into windows path format system. For example, GET/default.html is converted c:\ WebPages \ default. Html (httpsvr default root directory c\WebPages) after processed by this function.

void CRequestSocket :: StartTargetStuff (void). This function reads the content of a web page required by client and transfer in into m_buf for further sending.

UINT CGIThread (LPVOID pvParam) [7] The function can run in the thread model, call the corresponding GCI process, and output the results to a temporary file. After the thread model process is completed, the temporary files will be sent to the browser by BOOL CRequestSocket :: StartResponse (void).

4 HTTPSRV PROGRAM EEXCUTION PROCESS

4.1. Run Function

Program is started to be executed by the _tWinMain function of APPMODULE.CPP file in MFC. The function of return AfxWinMain is executed.

4.2. Link Server

A worker thread pThread is created by the int AFXAPI AfxWinMain function in WINMAIN.CPP file and initialize htt psvr. The BOOL CHttpSvrApp :: InitInstance method in HttpSvr.cpp file is called to initialize the HTTP server and then run the main function of thread. Finally, int CWinThread :: Run function is run in THRD-CORE.CPP file. The loop of server begins to run.

Within loops, the void CLis-tenSocket OnAccept method in Listen.cpp is called to generate CRequestSocket, which is set for monitoring port 8080. The port is set as 8080, the address of web service file is directed to the address of root. Now, the result is displayed as shown in Figure 2.

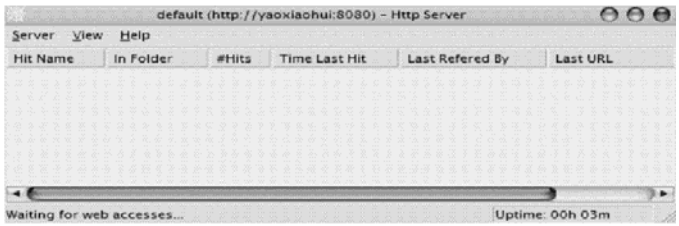

Figure 2. Link the server.

4.3. Requesting Client

In ReqSock.cpp file, different response-processing is conducted for the received data packets according to the difference of response status m_reqStatused. When the browser sends the first data packet, the response status m_reqStatus is set to the REQ_REQUEST. Then, each row of REQ_QUEST packet is processed. The m_pRequest is initialized using ProcessLine method according to http protocol. After the above operation is completed, the StartResponse method is called to construct the response message. The function of AsyncSelect is run. Thereafter, void CRequestSocket :: OnSend method is called to send the response message in m_buf cache to the client.

When you click submit, the return page is shown.

5 CONCLUSION & DISCUSSION

The primary function of a web server is to store, process and deliver web pages to clients. The communication between client and server takes place using HTTP. Pages delivered are most frequently HTML documents, which may include images, style sheets and scripts in addition to text content.

A user agent, commonly a web browser or web crawler, initiates communication by making a request for a specific resource using HTTP and the server responds with the content of that resource or an error message if unable to do so. The resource is typically a real file on the server's secondary storage, but this is not necessarily the case and depends on how the web server is implemented.

While the primary function is to serve content, a full implementation of HTTP also includes ways of receiving content from clients. This feature is used for submitting web forms, including uploading of files.

Many generic web servers also support server-side scripting using Active Server Pages (ASP), PHP, or other scripting languages. This means that the behavior of the web server can be scripted in separate files, while the actual server software remains unchanged. Usually, this function is used to create HTML documents dynamically ("on-the-fly") as opposed to returning static documents. The former is primarily used for retrieving and/or modifying information from databases. The latter is typically much faster and more easily cached but cannot deliver dynamic content.

Web servers are not always used for serving the World Wide Web. They can also be found embedded in devices such as printers, routers, webcams and serving only a local network. The web server may then be used as a part of a system for monitoring and/or administering the device in question. This usually means that no additional software has to be installed on the client computer, since only a web browser is required (which now is included with most operating systems).

In 1989 Tim Berners-Lee proposed a new project to his employer CERN, with the goal of easing the exchange of information between scientists by using a hypertext system. The project resulted in Berners-Lee writing two programs in 1990:

- A browser called World Wide Web.
- The world's first web server, later known as CERN httpd, which ran on NeXTSTEP

Between 1991 and 1994, the simplicity and effectiveness of early technologies used to surf and exchange data through the World Wide Web helped to port them to many different operating systems and spread their use among scientific organizations and universities, and then to industry.

In 1994 Tim Berners-Lee decided to constitute the World Wide Web Consortium (W3C) to regulate the further development of the many technologies involved (HTTP, HTML, etc.) through a standardization process.

In current study, by analyzing the HTTP server program, the authors will be familiar with HTTP service workflow and further understand the Socket, and then can create asynchronous Socket methods. In this article, source code of HTTP server with graphical interface is analyzed and debugged. The main functions of program, process running, the various kinds of function are introduced. Of course, this program is just a lightweight server whose function is relatively simple. However, after familiar with the basic principles of server creation basing on HTTP protocol, the authors will construct server with rich functions basing on this.

The HTTP is an application protocol for distributed, collaborative, hypermedia information systems [8]. HTTP is the foundation of data communication for the World Wide Web.

HTTP functions as a request-response protocol in the client-server computing model. A web browser, for example, may be the client and an application running on a computer hosting a web site may be the server. The client submits an HTTP request message to the server. The server, which provides resources such as HTML files and other content,. The response contains completion status information about the request and may also contain requested content in its message body.

A web browser is an example of a user agent (UA). Other types of user agent include the indexing software used by search providers (web crawlers), voice browsers, mobile apps, and other software that accesses, consumes, or displays web content.

HTTP is designed to permit intermediate network elements to improve or enable communications between clients and servers. High-traffic websites often benefit from web cache servers that deliver content on behalf of upstream servers to improve response time. Web browsers cache previously accessed web resources and reuses them when possible to reduce network traffic. HTTP proxy servers at private network boundaries can facilitate communication for clients without a globally routable address, by relaying messages with external servers.

HTTP uses Internet Media Types (formerly referred to as MIME Content-Types) to provide open and extensible data typing and type negotiation. For mail applications, where there is no type negotiation between sender and receiver, it's reasonable to put strict limits on the set of allowed media types. With HTTP, where the sender and recipient can communicate directly, applications are allowed more freedom in the use of non-registered types.

When the client sends a transaction to the server, headers are attached that conform to standard Internet e-mail specications (RFC 822). Most client requests expect an answer either in plain text or HTML. When the HTTP Server transmits information back to the client, it includes a MIME-like (Multipart Internet Mail Extension) header to inform the client what kind of data follows the header. Translation then depends on the client possessing the appropriate utility (image viewer, movie player, etc.) corresponding to that data type.

HTTP is an application layer protocol designed within the framework of the Internet Protocol Suite. Its definition presumes an underlying and reliable transport layer protocol,

and Transmission Control Protocol (TCP) is commonly used. However HTTP can use unreliable protocols such as the User Datagram Protocol (UDP), for example in Simple Service Discovery Protocol (SSDP).

HTTP resources are identified and located on the network by Uniform Resource Identifiers (URIs)—or, more specifically, Uniform Resource Locators (URLs)—using the http or https URI schemes. URIs and hyperlinks in Hypertext Markup Language (HTML) documents form webs of inter-linked hypertext documents.

HTTP/1.1 is a revision of the original HTTP (HTTP/1.0). In HTTP/1.0 a separate connection to the same server is made for every resource request. HTTP/1.1 can reuse a connection multiple times to download images, scripts, style sheets, etc after the page has been delivered. HTTP/1.1 communications therefore experience less latency as the establishment of TCP connections presents considerable overhead.

The term Hyper Text was coined by Ted Nelson in 1965 in the Xanadu Project, which was in turn inspired by Vannevar Bush's vision (1930's) of the microfilm-based information retrieval and management "memex" system described in his essay As We May Think (1945). Tim Berners-Lee and his team are credited with inventing the original HTTP along with HTML and the associated technology for a web server and a text-based web browser. Berners-Lee first proposed the "World Wide Web" project in 1989 — now known as the World Wide Web. The first version of the protocol had only one method, namely GET, which would request a page from a server. The response from the server was always an HTML page [9].

The first documented version of HTTP was HTTP V0.9 (1991). Dave Raggett led the HTTP Working Group (HTTP WG) in 1995 and wanted to expand the protocol with extended operations, extended negotiation, richer meta-information, tied with a security protocol which became more efficient by adding additional methods and header fields. RFC 1945 officially introduced and recognized HTTP V1.0 in 1996 [10].

The HTTP WG planned to publish new standards in December 1995 and the support for pre-standard HTTP/1.1 based on the then developing RFC 2068 (called HTTP-NG) was rapidly adopted by the major browser developers in early 1996. By March 1996, pre-standard HTTP/1.1 was supported in Arena, Netscape 2.0, Netscape Navigator Gold 2.01, Mosaic 2.7, and in Internet Explorer 2.0. End-user adoption of the new browsers was rapid. In March 1996, one web hosting company reported that over 40% of browsers in use on the Internet were HTTP 1.1 compliant. That same web hosting company reported that by June 1996, 65% of all browsers accessing their servers were HTTP/1.1 compliant [14]. The HTTP/1.1 standard as defined in RFC 2068 was officially released in January 1997. Improvements and updates to the HTTP/1.1 standard were released under RFC 2616 in June 1999.

In 2007, the HTTPbis Working Group was formed, in part, to revise and clarify the HTTP/1.1 spec. In June 2014, the WG released an updated six-part specification obsolescing RFC 2616.

An HTTP session is a sequence of network request-response transactions. An HTTP client initiates a request by establishing a Transmission Control Protocol (TCP) connection to a particular port on a server (typically port 80). An HTTP server listening on that port waits for a client's request message. Upon receiving the request, the server sends back a status line, such as "HTTP/1.1 200 OK", and a message of its own. The body of this message is typically the requested resource, although an error message or other information may also be returned .

HTTP defines methods to indicate the desired action to be performed on the identified resource. What this resource represents, whether pre-existing data or data that is generated dynamically, depends on the implementation of the server. Often, the resource corresponds to a file or the output of an executable residing on the server. The HTTP/1.0 specification defined the GET, POST and HEADS methods and the HTTP/1.1 specification and added 5 new methods: OPTIONS, PUT, DELETE, TRACE and CONNECT. By being specified in these documents their semantics are well known and can be depended upon. Any client can use any method and the server can be configured to support any combination of methods. If a method is unknown to an intermediate it will be treated as an unsafe and non-idempotent method. There is no limit to the number of methods that can be defined and this allows for future methods to be specified without breaking existing infrastructure. For example, WebDAV defined 7 new methods and RFC 5789 specified the PATCH method .

In conclusion, this study reviews the concepts, history, development, and future direction of web server and HTTP. The server program based on HTTP protocol and the process of interaction between the browser and server were analyzed to show how to use the theories of multi-threading and asynchronous operation. The core section of HTTP-based server program is gradually resolved with Winsock programming. This method can achieve the purpose to avoid blocking the calling thread and improve the responsiveness of a server program.

References

1. Ye Q. “Hypertext Transfer Protocol HTTP1.0”. Information Development & Economy,2004; (8):66-68
2. Xiao G. “HTTP protocol”. Technical Analysis Jiangxi Communication Technology, 2001; (1):39-41.
3. Xu J., Wang T. “Analysis of HTTP protocol 1.1”. Journal of Southwestern Normal University (Science edition), 2004; (2):47-48.
4. Tanenbaum AS. “Internet”. Tsinghua University Press, Beijing, 2004, pp. 150–153.
5. Li L, Li C. “The computer netwotk technique”. National Defense Industrial Press, Beijing, 2004, pp. 250–266.

6. Wang Y, Zhang Y. “Windows network and communication program design”. People’s Posts and Telecommunications Press. Beijing, 2006, pp. 340–345.
7. Zhang H, Liu Y. “Research on HTTP based on socket”. Journal of Qiqihar University, 2004; (2):70-72.
8. Roy T, James G, Jeffrey CM, Henrik FN, Larry M, Paul JL, Lee B. Hypertext Transfer Protocol -- HTTP/1.1. IETF. RFC 2616, 1999 Jun.
9. Berners-Lee T. “The Original HTTP as defined in 1991”. World Wide Web Consortium. 2010 July.
10. Raggett D, Berners-Lee T. “Hypertext Transfer Protocol Working Group”. World Wide Web Consortium. 2010 September.
11. Simon S. "Progress on HTTP-NG". World Wide Web Consortium. 2010 June.