**HTTP服务器实验**

1. 实验内容

使用C语言实现最简单的HTTP服务器，满足以下要求：

1.分别使用两个线程监听，同时支持HTTP（80端口）和HTTPS（443端口）；

2.实验GET方法，解析请求报文，返回相应应答及内容，需要支持的状态码包含200 OK、301 Moved Permanently、206 Partial Content、404 Not Found。

1. 实验基础
2. HTTP协议

超文本传输协议（英语：**H**yper**T**ext **T**ransfer **P**rotocol，缩写：HTTP）是一种用于分布式、协作式和超媒体信息系统的应用层协议。通过HTTP或者HTTPS协议请求的资源由统一资源标识符（Uniform Resource Identifiers，URI）来标识。本次实验选取了HTTP/1.1版本的协议。HTTP/1.1默认采用持续连接（Connection: keep-alive），可以在一次连接中传输多次数据。

通常，由HTTP客户端发起一个请求，创建一个到服务器指定端口（默认80端口）的TCP连接。HTTP服务器则在那个端口监听客户端的请求。一旦收到请求，服务器会向客户端返回一个状态，比如"HTTP/1.1 200 OK"，以及返回的内容，如请求的文件、错误消息或者其它信息。

HTTP协议的请求报文由请求行、请求头部、空行和请求数据4个部分组成。请求行包括请求方法（本实验仅支持GET方法）、URL、协议版本，中间以空格隔开，以回车符和换行符作为结尾。请求头部（1行或多行）由关键字/值对组成，每行一对，关键字和值用英文冒号“:”分隔。请求头部可以用于通知服务器有关于客户端请求的信息，如客户端可接受的文件类型、请求的主机名等。空行用于通知服务器请求头部的结束。请求数据不在GET方法中使用。具体的请求报文格式如图1所示。



图1 请求报文格式

HTTP协议的响应报文由状态行、响应头、空行和响应体4个部分组成。状态行由协议版本号、状态码、状态消息组成，中间以空格隔开。响应头由0或多个字段组成，允许服务器传递超出状态行的附加信息，充当响应修饰符。它们提供有关服务器或有关进一步访问目标资源或相关资源的信息。空行用于通知客户端响应头的结束。响应数据为请求文件的内容。当响应状态(Response Status)为1xx/204/304，响应正文长度为0；如果使用了非"identity"的Transfer-Encoding编码方式,则响应正文长度由"chunked"编码决定,除非该消息以连接关闭为结束；如果存在"Content-Length"相应头,则响应正文长度为该数值。具体响应报文格式如图2所示。



图2 响应报文格式

1. HTTPS协议

安全超文本传输协议(HTTPS)是超文本传输协议(HTTP)的扩展，用于计算机网络上的安全通信。在HTTPS中，通信协议使用传输层安全性（TLS）或安全套接字层（SSL）进行加密。HTTPS 的主要作用是对访问的网站进行身份验证以及在传输过程中保护交换数据的隐私和完整性。

HTTP协议和安全协议同属于应用层（OSI模型的最高层），具体来讲，安全协议工作在HTTP之下，传输层之上：安全协议向运行HTTP的进程提供一个类似于TCP的套接字，供进程向其中注入报文，安全协议将报文加密并注入运输层套接字；或是从运输层获取加密报文，解密后交给对应的进程。严格地讲，HTTPS并不是一个单独的协议，而是对工作在一加密连接（TLS或SSL）上的常规HTTP协议的称呼。服务器必须使用一个受信任机构签发的公钥证书。客户端会验证证书的合法性，如果验证通过才会进行后续通信，否则根据错误情况不同做出提示和操作本实验采用SSL加密。

1. Socket API

Socket（套接字）是用户进程和网络之间的统一接口。“IP+端口+协议”的组合就可以唯一标识网络中一台主机上的一个进程，这也是Socket的主要参数。Socket可以看成在发送端与接受端进行通讯连接中的端点。具体的服务器与客户端之间通过Socket进行连接通信的流程如图3所示。

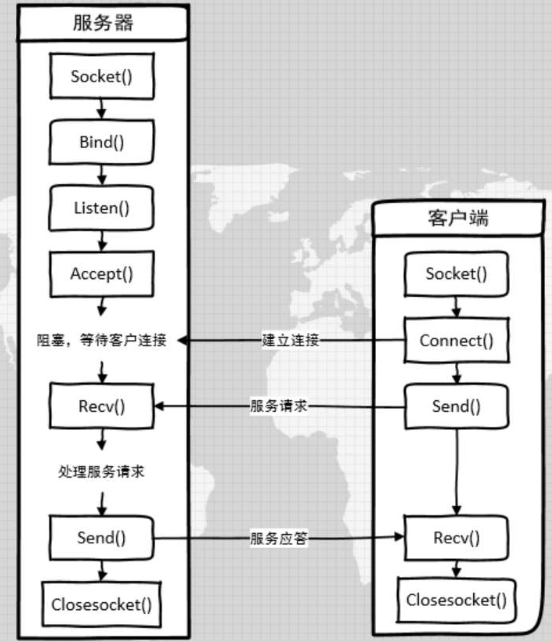


图3 Socket通信流程

1. SSL协议

SSL是一个不依赖于平台和运用程序的协议，位于TCP/IP协议与各种应用层协议之间，为数据通信提高安全支持。SSL的体系结构中包含两个协议子层，其中底层是SSL记录协议层（SSL Record Protocol Layer）；高层是SSL握手协议层（SSL HandShake Protocol Layer）。

SSL握手层协议用于SSL管理信息的交换，允许应用协议传送数据之间相互验证，协商加密算法和生成密钥等。SSL记录协议层的作用是为高层协议提供基本的安全服务，具体实施压缩解压缩、加密解密、计算和校验MAC等与安全有关的操作。

整个SSL通信的流程包括：客户端向服务器请求证书，验证无误后拿到服务器的公钥；双方协商生成一个session key；最后双方采用session key进行加密通信。具体的SSL连接过程如图5所示。

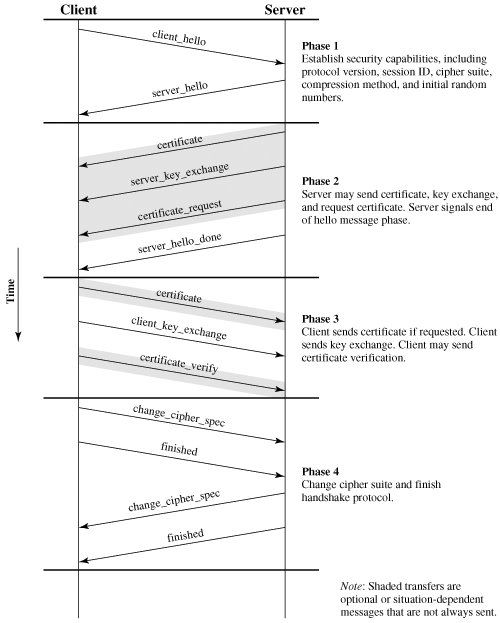


图4 SSL连接过程

1. 设计思路
2. 测试结果
3. 视频传输流程分析

参考资料

RFC 7230, HTTP/1.1: Message Syntax and Routing

RFC 7231, HTTP/1.1: Semantics and Content

RFC 7232, HTTP/1.1: Conditional Requests

RFC 7233, HTTP/1.1: Range Requests

RFC 7234, HTTP/1.1: Caching

RFC 7235, HTTP/1.1: Authentication