实验三 网络通信实验

一、实验目的

1、复习计算机网络的基础知识；

2、了解计算机网络在嵌入式 linux 系统中的应用；

3、学习使用嵌入式 linux 系统中的计算机网络编程接口；

4、了解计算机网络应用软件的软件架构与编程技术。

二、实验器材

iTop-4412 实验开发板一套配套串口线一根

配套串口 USB 转接器一个

CAT5E 网线一根

PC 机一台（Windows 操作系统）

U 盘一个。

三、实验原理

3.1 计算机网络的五层架构表 1 TCP/IP 与 ISO 模型对

关系表

下图示意了 TCP/IP 与 ISO OSI 参考模型之间的对应关系。

OSI 体系结构	TCP/IP 协议集	
应用层	应用层	TELNET、FTP、HTTP、SMTP、DNS 等
表示层		
会话层		
传输层	传输层	TCP、UDP
网络层	网络层	IP、ICMP、ARP、RARP
数据链路层	网络接口层	各种物理通信网络接口
物理层		

3.2 传输层简介

3.2.1 TCP[6]

TCP（Transmission Control Protocol 传输控制协议）是一种面向连接的、可靠的、基于字节流的传输层通信协议，由 IETF 的 RFC 793 定义。在简化的计算机网络 OSI 模型中，它完成第四层传输层所指定的功能，用户数据报协议（UDP）是同一层内另一个重要的传输协议。在因特网协议族（Internet protocol suite）中， TCP 层是位于 IP 层之上，应用层之下的中间层。不同主机的应用层之间经常需要可靠的、像管道一样的连接，但是 IP 层不提供这样的流机制，而是提供不可靠的包交换。

应用层向 TCP 层发送用于网间传输的、用 8 位字节表示的数据流，然后 TCP 把数据流分区成适当长度的报文段（通常受该计算机连接的网络的数据链路层的最大传输单元（MTU）的限制）。之后 TCP 把结果包传给 IP 层，由它来通过网络将包传送给接收端实体的 TCP 层。TCP 为了保证不发生丢包，就给每个包一个序号，同时序号也保证了传送到接收端实体的包的按序接收。然后接收端实体对已成功收到的包发回一个相应的确认（ACK）；如果发送端实体在合理的往返时延（RTT）内未收到确认，那么对应的数据包就被假设为已丢失将会被进行重传。TCP 用一个校验和函数来检验数据是否有错误；在发送和接收时都要计算校验和。

图 1 是 TCP 建立连接三次握手示意图，图 3 是 TCP 释放连接的四次握手示意图。

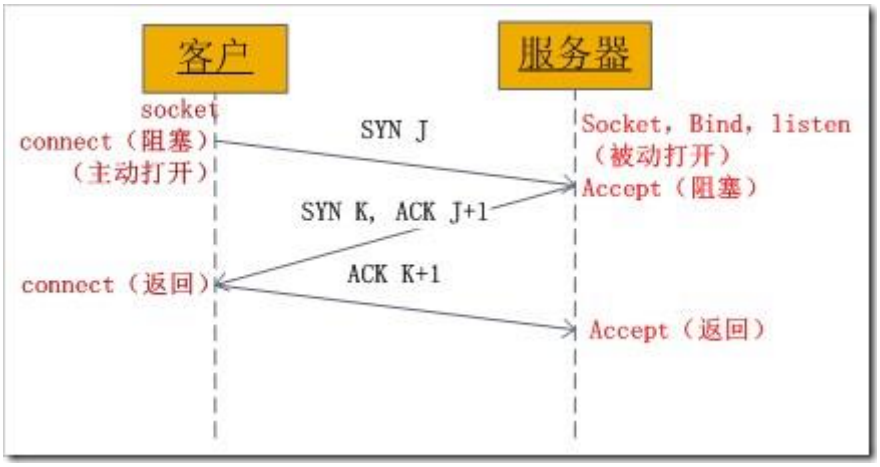

图 1 socket 中 TCP 的三次握手建立连接示意图

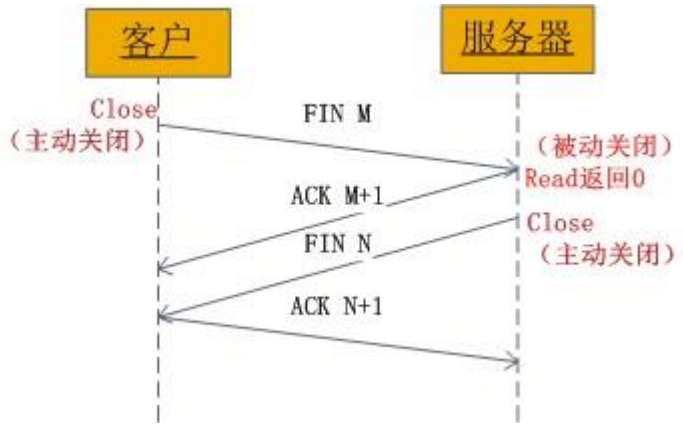

图 2 socket 中 TCP 的四次握手释放连接示意图

3.2.2 UDP[7]

UDP 是 User Datagram Protocol 的简称， 中文名是用户数据报协议，是 OSI（Open System Interconnection，开放式系统互联） 参考模型中一种无连接的传输层协议，提供面向事务的简单不可靠信息传送服务，IETF RFC 768 是 UDP 的正式规范。UDP 在 IP 报文的协议号是 17。

UDP 协议全称是用户数据报协议，在网络中它与 TCP 协议一样用于处理数据包，是一种无连接的协议。在 OSI 模型中，在第四层——传输层，处于 IP 协议的上一层。UDP 有不提供数据包分组、组装和不能对数据包进行排序的缺点，也就是说，当报文发送之后，是无法得知其是否安全完整到达的。UDP 用来支持那些需要在计算机之间传输数据的网络应用。包括网络视频会议系统在内的众多的客户/服务器模式的网络应用都需要使用 UDP 协议。UDP 协议从问世至今已经被使用了很多年，虽然其最初的光彩已经被一些类似协议所掩盖，但是即使是在今天 UDP 仍然不失为一项非常实用和可行的网络传输层协议。

与所熟知的 TCP（传输控制协议）协议一样，UDP 协议直接位于 IP（网际协议）协议的顶层。根据 OSI（开放系统互联）参考模型，UDP 和 TCP 都属于传输层协议。UDP 协议的主要作用是将网络数据流量压缩成数据包的形式。一

个典型的数据包就是一个二进制数据的传输单位。每一个数据包的前 8 个字节用来包含报头信息，剩余字节则用来包含具体的传输数据。

3.2.3 编程接口：套接字 socket

传输层协议 TCP 和 UDP 用主机的 IP 地址加上主机上的端口号作为连接的端点，这种端点就叫做套接字（socket）或插口。

套接字用（IP 地址：端口号）表示。

它是网络通信过程中端点的抽象表示，包含进行网络通信必需的五种信息：连接使用的协议，本地主机的 IP 地址，本地进程的协议端口，远地主机的 IP 地址，远地进程的协议端口。图 3 是 socket 通信模型图。

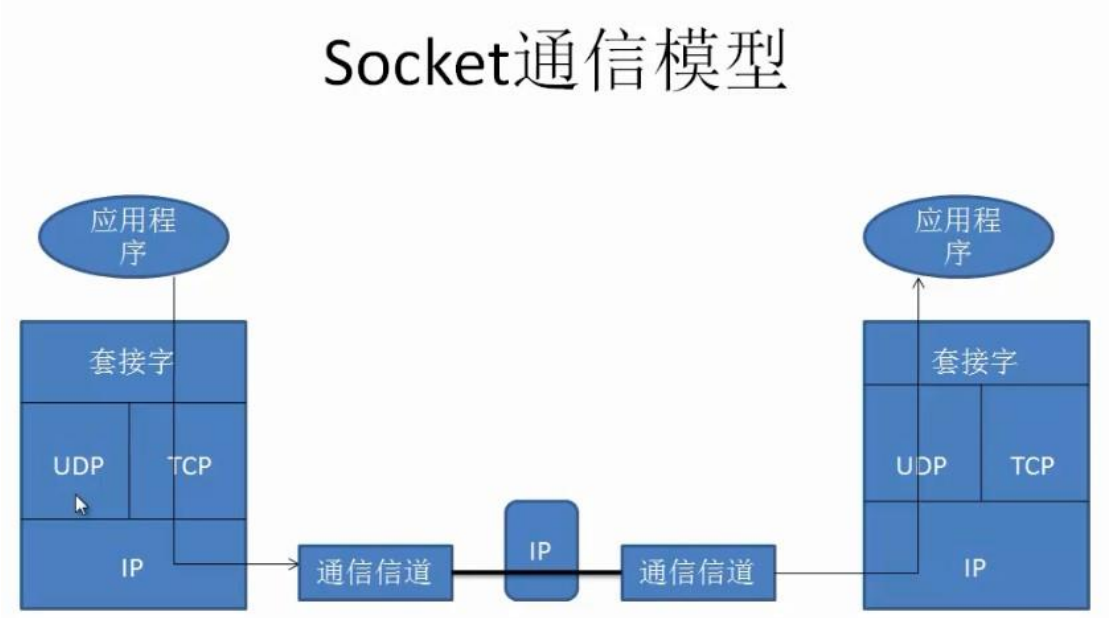

图 3 socket 通信模型图

3.3 网络通信与 IO 模型[1][2]

3.3.1 网络通信与文件读写的比较

文件读写与网络通信的区别主要在两点：一是网络通信读写数据的时间不确定性；二是网络通信连接的数量常常比文件 IO 大的多。因此，在使用文件 IO 读写文件时，使用简单的 IO 模型常常可以完成所需功能，而网络通信则需要针对特定的应用场合，合理选择使用 IO 模型，如大量连接且实时性要求不高的应用场景、少量连接但实时性要求高的应用场景等。在 Linux 中，有五种 IO 模型。

IO 模型是通用的，适用于网络通信、文件读写或其他 IO 操作。有关于 java 和 windows 的 IO 模型见注[2]。

3.3.2 Linux 的 IO 模型

以下内容是注[1]博文的全文拷贝：

1、阻塞 **IO** 模型场景描述：

在饭店领完号后，前面还有 n 桌，不知道什么时候到我们，但是又不能离开，因为过号之后必须重新取号。只好在饭店里等，一直等到叫号到我们才吃完晚饭，然后去逛街。中间等待的时间什么事情都不能做。

网络模型：

在这个模型中，应用程序为了执行这个 read 操作，会调用相应的一个 system call，将系统控制权交给内核，然后就进行等待（这个等待的过程就是被阻塞了），内核开始执行这个 system call，执行完毕后会向应用程序返回响应，应用程序得到响应后，就不再阻塞，并进行后面的工作。

优点：

能够及时返回数据，无延迟。

缺点：

对用户来说处于等待就要付出性能代价。

2、非阻塞 **IO**

场景描述：

等待过程是在太无聊，于是我们就去逛商场，每隔一段时间就回来询问服务员，叫号是否到我们了，整个过程来来回回好多次。这就是非阻塞，但是需要不断的询问。网络模型：

当用户进程发出 read 操作时，调用相应的 system call，这个 system call 会立即从内核中返回。但是在返回的这个时间点，内核中的数据可能还没有准备好，也就是说内核只是很快就返回了 system call，只有这样才不会阻塞用户进程，对于应用

程序，虽然这个 IO 操作很快就返回了，但是它并不知道这个 IO 操作是否真的成功了，为了知道 IO 操作是否成功，应用程序需要主动的循环去问内核。

优点：

能够在等待的时间里去做其他的事情。

缺点：

任务完成的响应延迟增大了，因为每过一段时间去轮询一次 read 操作，而任务可能在两次轮询之间的任意时间完成，这对导致整体数据吞吐量的降低。

3、IO 多路复用

场景描述：

与第二个经常类似，饭店安装了电子屏幕，显示叫号的状态，所以在逛街的时候，就不用去询问服务员，而是看下大屏幕就可以了。（不仅仅是我们不用询问服务员，其他所有的人都可以不用询问服务员）

网络模型：

和第二种一样，调用 system call 之后，并不等待内核的返回结果而是立即返回。虽然返回结果的调用函数是一个异步的方式，但应用程序会被像 select、poll 和 epoll 等具有多个文件描述符的函数阻塞住，一直等到这个 system call 有结果返回了，再通知应用程序。这种情况，从 IO 操作的实际效果来看，异步阻塞 IO 和第一种同步阻塞 IO 是一样的，应用程序都是一直等到 IO 操作成功之后（数据已经被写入或者读取），才开始进行下面的工作。不同点在于异步阻塞 IO 用一个 select 函数可以为多个文件描述符提供通知，提供了并发性。举个例子：例如有一万个并发的 read 请求，但是网络上仍然没有数据，此时这一万个 read 会同时各自阻塞，现在用 select、poll、epoll 这样的函数来专门负责阻塞同时监听这一万个请求的状态，一旦有数据到达了就负责通知，这样就将一万个等待和阻塞转化为一个专门的函数来负责与管理。

多路复用技术应用于 JAVA NIO 的核心类库多路复用器 Selector 中，目前 Linux 中支持 I/O 多路复用的系统调用有 select、pselect、poll、epoll，在 linux

编程中有一段时间一直在使用 select 做轮询和网络事件通知的，但是 select 支持一个进程打开的 socket 描述符（FD）收到了限制，一般为 1024，由于这一限制，

现在使用了 epoll 代替了 select，而 epoll 支持一个进程打开的 FD 不受限制。

异步 IO 与同步 IO 的区别在于：同步 IO 是需要应用程序主动地循环去询问是否有数据，而异步 IO 是通过像 select 等 IO 多路复用函数来同时检测多个事件句柄来告知应用程序是否有数据。

了解了前面三种 IO 模式，在用户进程进行系统调用的时候，他们在等待数据到来的时候，处理的方式是不一样的，直接等待、轮询、select 或 poll 轮询，两个阶段过程：第一个阶段有的阻塞，有的不阻塞，有的可以阻塞又可以不阻塞。

第二个阶段都是阻塞的。

从整个 IO 过程来看，他们都是顺序执行的，因此可以归为同步模型，都是进程自动等待且向内核检查状态。

高并发的程序一般使用同步非阻塞模式，而不是多线程+同步阻塞模式。要理解这点，先弄明白并发和并行的区别：比如去某部门办事需要依次去几个窗口，办事大厅的人数就是并发数，而窗口的个数就是并行度。就是说并发是同时进行的任务数（如同时服务的 http 请求），而并行数就是可以同时工作的物理资源数量（如 cpu 核数）。通过合理调度任务的不同阶段，并发数可以远远大于并行度。这就是区区几个 CPU 可以支撑上万个用户并发请求的原因。在这种高并发的情况下，为每个用户请求创建一个进程或者线程的开销非常大。而同步非阻塞方式可以把多个 **IO** 请求丢到后台去，这样一个 **CPU** 就可以服务大量的并发 **IO** 请求。

IO 多路复用究竟是同步阻塞还是异步阻塞模型，这里来展开说说：

同步是需要主动等待消息通知，而异步则是被动接受消息通知，通过回调、通知、状态等方式来被动获取消息。IO 多路复用在阻塞到 select 阶段时，用户进程是主动等待并调用 select 函数来获取就绪状态消息，并且其进程状态为阻塞。

所以 IO 多路复用是同步阻塞模式。

4、信号驱动式 IO

应用程序提交 read 请求，调用 system call，然后内核开始处理相应的 IO 操作，而同时，应用程序并不等内核返回响应，就会开始执行其他的处理操作（应用程序没有被 IO 阻塞），当内核执行完毕，返回 read 响应，就会产生一个信号或执行一个基于线程的回调函数来完成这次 IO 处理过程。在这里 IO 的读写操作是在 IO 事件发生之后由应用程序来完成。异步 IO 读写操作总是立即返回，而不论 IO 是否阻塞，因为真正的读写操作已经有内核掌管。也就是说同步 IO 模型要求用户代码自行执行 IO 操作（将数据从内核缓冲区移动用户缓冲区或者相反），而异步操作机制则是由内核来执行 IO 操作（将数据从内核缓冲区移动用户缓冲区或者相反）。可以这样认为，同步 IO 向应用程序通知的是 IO 就绪事件，而异步 IO 向应用程序通知的是 IO 完成事件。

5、异步 IO

异步 IO 与上面的异步概念是一样的， 当一个异步过程调用发出后，调用者不能立刻得到结果，实际处理这个调用的函数在完成后，通过状态、通知和回调来通知调用者的输入输出操作。异步 IO 的工作机制是：告知内核启动某个操作，并让内核在整个操作完成后通知我们，这种模型与信号驱动的 **IO** 区别在于，信号驱动 **IO** 是由内核通知我们何时可以启动一个 **IO** 操作，这个 **IO** 操作由用户自定义的信号函数来实现，而异步 **IO** 模型是由内核告知我们 **IO** 操作何时完成。

3.4 运输层协议之 HTTP

超文本传输协议（HTTP，HyperText Transfer Protocol)是互联网上应用最为广泛的一种网络协议。所有的 WWW 文件都必须遵守这个标准。设计 HTTP 最初的目的是为了提供一种发布和接收 HTML 页面的方法。1960 年美国人 Ted Nelson 构思了一种通过计算机处理文本信息的方法，并称之为超文本（hypertext）,这成为了 HTTP 超文本传输协议标准架构的发展根基。Ted Nelson

组织协调万维网协会（World Wide Web Consortium）和互联网工程工作小组（Internet Engineering Task Force ）共同合作研究，最终发布了一系列的 RFC，其中著名的 RFC 2616 定义了 HTTP 1.1。

3.4.1 HTTP 的状态码[3]

HTTP 状态码（英语：HTTP Status Code）是用以表示网页服务器超文本传输协议响应状态的 3 位数字代码。它由 RFC 2616 规范定义的，并得到 RFC 2518、RFC 2817、RFC 2295、RFC 2774 与 RFC 4918 等规范扩展。所有状态码的第一个数字代表了响应的五种状态之一。所示的消息短语是典型的，但是可以提供任何可读取的替代方案。 除非另有说明，状态码是 HTTP / 1.1 标准（RFC 7231）的一部分。

HTTP 状态码的官方注册表由互联网号码分配局（Internet Assigned Numbers Authority）维护。

微软互联网信息服务 （Microsoft Internet Information Services）有时会使用额外的十进制子代码来获取更多具体信息，但是这些子代码仅出现在响应有效内容和文档中，而不是代替实际的 HTTP 状态代码。

表 2 HTTP 协议的状态码

目录

1 消息
- 100 Continue
- 101 Switching Protocols
- 102 Processing

2 成功
- 200 OK
- 201 Created
- 202 Accepted
- 203 Non-Authoritative Information
- 204 No Content
- 205 Reset Content
- 206 Partial Content
- 207 Multi-Status

3 重定向
- 300 Multiple Choices
- 301 Moved Permanently
- 302 Move Temporarily
- 303 See Other
- 304 Not Modified
- 305 Use Proxy
- 306 Switch Proxy
- 307 Temporary Redirect

4 请求错误
- 400 Bad Request
- 401 Unauthorized
- 402 Payment Required
- 403 Forbidden
- 404 Not Found
- 405 Method Not Allowed
- 406 Not Acceptable
- 407 Proxy Authentication Required
- 408 Request Timeout
- 409 Conflict
- 410 Gone
- 411 Length Required
- 412 Precondition Failed
- 413 Request Entity Too Large
- 414 Request-URI Too Long
- 415 Unsupported Media Type
- 416 Requested Range Not Satisfiable
- 417 Expectation Failed
- 418 I'm a teapot
- 421 Too Many Connections
- 422 Unprocessable Entity
- 423 Locked
- 424 Failed Dependency
- 425 Unordered Collection
- 426 Upgrade Required
- 449 Retry With
- 451 Unavailable For Legal Reasons

5 服务器错误
- 500 Internal Server Error
- 501 Not Implemented
- 502 Bad Gateway
- 503 Service Unavailable
- 504 Gateway Timeout
- 505 HTTP Version Not Supported
- 506 Variant Also Negotiates
- 507 Insufficient Storage
- 509 Bandwidth Limit Exceeded
- 510 Not Extended
- 600 Unparseable Response Headers

3.5 SSL/TLS[4]

传输层安全性协议（英语：Transport Layer Security，缩写作 TLS），及其前身安全套接层（Secure Sockets Layer，缩写作 SSL）是一种安全协议，目的是为互联网通信提供安全及数据完整性保障。网景公司（Netscape）在 1994 年推出首版网页浏览器，网景导航者时，推出 HTTPS 协议，以 SSL 进行加密，这是 SSL 的起源。IETF 将 SSL 进行标准化，1999 年公布第一版 TLS 标准文件。随后又公布 RFC 5246 （2008 年 8 月）与 RFC 6176（2011 年 3 月）。在浏览器、邮箱、即时通信、VoIP、网络传真等应用程序中，广泛支持这个协议。主要的网站，如 Google、 Facebook 等也以这个协议来创建安全连线，发送数据。目前已成为互联网上保密通信的工业标准。

SSL 包含记录层（Record Layer）和传输层，记录层协议确定传输层数据的封装格式。传输层安全协议使用 X.509 认证，之后利用非对称加密演算来对通信方做身份认证，之后交换对称密钥作为会谈密钥（Session key）。这个会谈密钥是用来将通信两方交换的数据做加密，保证两个应用间通信的保密性和可靠性，使客户与服务器应用之间的通信不被攻击者窃听。

3.5.1 HTTPS[5]

HTTPS（全称：Hyper Text Transfer Protocol over Secure Socket Layer 或 Hypertext Transfer Protocol Secure，超文本传输安全协议），是以安全为目标的 HTTP 通道，简单讲是 HTTP 的安全版。即 HTTP 下加入 SSL 层，HTTPS 的安全基础是 SSL，因此加密的详细内容就需要 SSL。 它是一个 URI scheme（抽象标识符体系），句法类同 http:体系。用于安全的 HTTP 数据传输。https:URL 表明它使用了 HTTP，但 HTTPS 存在不同于 HTTP 的默认端口及一个加密/身份验证层（在 HTTP 与 TCP 之间）。这个系统的最初研发由网景公司(Netscape)进行，并内置于其浏览器 Netscape Navigator 中，提供了身份验证与加密通讯方法。现在它被广泛用于万维网上安全敏感的通讯，例如交易支付方面。

四、实验内容

4.1 TCP Client

使用网线将开发板与 PC 连接；

在 PC 上运行网络调试助手，建立 TCP Server；

编写运行于开发板的程序，向上述 TCP Server 发送数据。

4.2 TCP Server

使用网线将开发板与 PC 连接；

编写运行于 PC 的程序，建立 TCP Server；

使用开发板向上述 TCP Server 发送数据。

五、实验步骤

实验 3.1 要求：发送“学号姓名”及时间戳（年月日时分秒，可选毫秒，具体格式不限）至网络调试助手；

实验 3.2 要求：完成上位机（PC）的服务器软件，实现开发板作为客户端，PC 作为服务器端，开发板发送内容给 PC，PC 端可以打印出发送的数据。服务端不限语言和平台。

实验 3.3：完成登录功能。以实验 3.2 为基础，在客户端连接上服务器时，服务器告知客户端输入登录信息。客户端手动输入账号、密码登录（abc，def 分别为正确的账号和密码）。服务端需校验账号密码是否正确，正确则返回 login successfully，错误则返回 login failed。 若正确，后续同 3.2，客户端可以给服务端发消息，服务端需要打印出客户端发送的消息；若错误，断开连接。

六、实验注意事项

1、 互联网是当今人类最大的计算机网络，几乎覆盖全球；嵌入式设备连接至互联网有很大的意义。

2、 连接至东南大学无线 wifi 网络 seu-wlan 的主机之间的通信被防火墙屏蔽，因此无法直接相互通信。使用东南大学 vpn 是一种解决方案， vpn 服务器可与 seuwlan 网络下的主机通信，当一台主机使用东南大学 vpn 时，即可访问 seu-wlan 下的主机。

3、 以太网方式接入东南大学校园网的主机之间可以直接通信，可以与连接 seuwlan 的主机直接通信。

4、2 和 3 是对 2019 年 1 月 1 日的东南大学网络环境的描述。

注：

[1] https://www.cnblogs.com/renxs/p/3683189.html

[2] Java IO 模型有:BIO、NIO、 AIO；windows 平台上的 IO 模型有：选择（Select)、异步选择（WSAAsyncSelect)、事件选择（WSAEventSelect)、 重叠 I/O（Overlapped

I/O）和完成端口（Completion Port)共五种 I/O 模型；

Linux 平台上的 IO 模型有：阻塞 IO、非阻塞 IO、多路复用 IO、信号驱动 IO、异步 IO 。

[3]https://baike.baidu.com/item/HTTP%E7%8A%B6%E6%80%81%E7%A0%81/5053660?fr=aladdin

[4] https://baike.baidu.com/item/TLS/2979545?fr=aladdin

[5] https://baike.baidu.com/item/https/285356?fr=aladdin [6] https://baike.baidu.com/item/TCP/33012?fr=aladdin

[7] https://baike.baidu.com/item/UDP/571511?fr=aladdin 题库：

1、简述计算机网络的五层网络模型

2、简述 Linux 五种 IO 模型

3、简述 HTTP 状态码的分类

4、简述 SSL/TLS 基本原理

5、简述域名与 IP 关系

6、简述 HTTP 常用方法（如 GET POST）

7、简述 TCP 的三次握手与四次握手

8、简述 UDP 的应用场景

9、简述程序流程