一、 常见面试题

1. IP 地址和端口号

1）IP 地址

用来标志网络中的一个通信实体的地址。通信实体可以是计算机，路由器等。

2）IP 地址分类

IPV4：32 位地址，以点分十进制表示，如 192.168.0.1

IPV6：128 位（16 个字节）写成 8 个 16 位的无符号整数，每个整数用四个十六进制位表示，数之间用冒号（:）分开，如：3ffe:3201:1401:1280:c8ff:fe4d:db39:1984

3）特殊的 IP 地址

127.0.0.1 本机地址

192.168.0.0--192.168.255.255 私有地址，属于非注册地址，专为组织机构内部使用。

4）端口 port

IP 地址用来标志一台计算机，但是一台计算机上可能提供多种应用程序，使用端口来区分这些应用程序。 端口是虚拟的概念，并不是说在主机上真的有若干个端口。通过端口，可以在一个主机上运行多个网络应用程序。 端口范围 0---65535,16 位整数

5）端口分类

公认端口 0—1023 比如 80 端口分配给 WWW，21 端口分配给 FTP，22 端口分配给 SSH,23 端口分配给 telnet，25 端口分配给 smtp

注册端口 1024—49151 分配给用户进程或应用程序

动态/私有端口 49152--65535

6）理解 IP 和端口的关系

IP 地址好比每个人的地址（门牌号），端口好比是房间号。必须同时指定 IP 地址和端口号才能够正确的发送数据

IP 地址好比为电话号码，而端口号就好比为分机号。

2. 常用网络通信协议

TCP/IP 协议

TCP 传输控制协议:TCP 协议是一种可靠的端对端协议，重发一切没有收到的数据，进行数据内容准确性检查并保证分组的正确顺序。

IP 网际协议:规定数据传输格式

HTTP 协议/HTTPS

HTTP 超文本传输协议,基于请求和响应模式。

HTTPS 协议是由 SSL+HTTP 协议构建的可进行加密传输、身份认证的网络协议，要比 http 协议安全，可防止数据在传输过程中被窃取、改变，确保数据的完整性，它大幅增加了中间人攻击的成本。

FTP 协议

文件传输协议

SMTP 协议

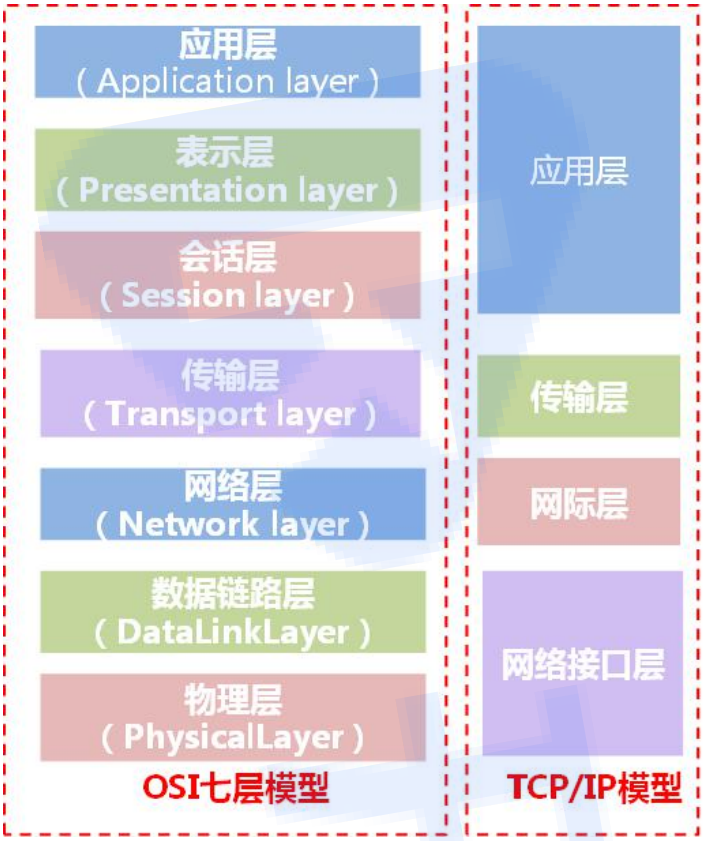
简单邮件传输协议

POP3/IMAP 协议

POP3 邮局协议版本 3

IMAP:Internet 消息访问协议

3. OSI 七层模型



这个模型推出的最开始，是是因为美国人有两台机器之间进行通信的需求。

需求 1：

科学家要解决的第一个问题是，两个硬件之间怎么通信。具体就是一台发些比特流，然后另一台能收到。

于是，科学家发明了物理层：

主要定义物理设备标准，如网线的接口类型、光纤的接口类型、各种传输介质的传输速率等。它的主要作用是传输比特流(就是由 1、0 转化为电流强弱来进行传输，到达目的地后在转化为 1、0，也就是我们常说的数模转换与模数转换)。这一层的数据叫做比特。

需求 2：

现在通过电线我能发数据流了，但是还希望通过无线电波，通过其它介质来传输。然后我还要保证传输过去的比特流是正确的，要有纠错功能。

于是，发明了数据链路层 ：

定义了如何让格式化数据以进行传输，以及如何让控制对物理介质的访问。这一层通常还提供错误检测和纠正，以确保数据的可靠传输。

需求 3：

现在我能发正确的发比特流数据到另一台计算机了，但是当我发大量数据时候，可能需要好长时间，例如一个视频格式的，网络会中断好多次（事实上，即使有了物理层和数据链路层，网络还是经常中断，只是中断的时间是毫秒级别的）。

那么，我还须要保证传输大量文件时的准确性。于是，我要对发出去的数据进行封装。就像发快递一样，一个个地发。

于是，先发明传输层（传输层在 OSI 模型中，是在网络层上面）

例如 TCP，是用于发大量数据的，我发了 1 万个包出去，另一台电脑就要告诉我是否接受到了 1 万个包，如果缺了 3 个包，就告诉我是第 1001，234，8888 个包丢

了，那我再发一次。这样，就能保证对方把这个视频完整接收了。
例如 UDP，是用于发送少量数据的。我发 20 个包出去，一般不会丢包，所以，我不管你收到多少个。在多人互动游戏，也经常用 UDP 协议，因为一般都是简单的信息，而且有广播的需求。如果用 TCP，效率就很低，因为它会不停地告诉主机我收到了 20 个包，或者我收到了 18 个包，再发我两个！如果同时有 1 万台计算机都这样做，那么用 TCP 反而会降低效率，还不如用 UDP，主机发出去就算了，丢几个包你就卡一下，算了，下次再发包你再更新。
TCP 协议是会绑定 IP 和端口的协议，下面会介绍 IP 协议。

需求 4：

传输层只是解决了打包的问题。但是如果我有多台计算机，怎么找到我要发的那台？或者，A 要给 F 发信息，中间要经过 B，C，D,E，但是中间还有好多节点如 K.J.Z.Y。我怎么选择最佳路径？这就是路由要做的事。
于是，发明了网络层 。即路由器，交换价那些具有寻址功能的设备所实现的功能。这一层定义的是 IP 地址，通过 IP 地址寻址。所以产生了 IP 协议。

需求 5：

现在我们已经保证给正确的计算机，发送正确的封装过后的信息了。但是用户级别的体验好不好？难道我每次都要调用 TCP 去打包，然后调用 IP 协议去找路由，自己去发？当然不行，所以我们要建立一个自动收发包，自动寻址的功能。
于是，发明了会话层 。会话层的作用就是建立和管理应用程序之间的通信。

需求 6：

现在我能保证应用程序自动收发包和寻址了。但是我要用 Linux 给 window 发包，两个系统语法不一致，就像安装包一样，exe 是不能在 linux 下用的，shell 在 window 下也是不能直接运行的。于是需要表示层（presentation），帮我们解决不同系统之间的通信语法问题。

需求 7：

现在所有必要条件都准备好了，我们可以写个 android 程序，web 程序去实现需求。

4. TCP/IP 协议有了解吗

OSI 是一个理论上的七层网络通信模型，而 TCP/IP 则是实际运行的四层网络协议。
TCP/IP 包含：
网络接口层，主机必须使用某种协议与网络相连
网络层，使主机可以把分组发往任何网络，并使分组独立地传向目标。IP 协议
传输层，使源端和目的端机器上的对等实体可以进行会话。TCP 和 UDP 协议
应用层，包含所有的高层协议，FTP/TELNET/SMTP/DNS/NNTP/HTTP

5. Http 协议

HTTP 是 TCP/IP 协议中的应用层协议。它不涉及数据包传输，主要规定了客户端和服务器之间的通信格式，是互联网信息交互中最常用的协议

特点：

简单快速。只需要传请求方法与资源路径就能确定资源
灵活，传输任意类型的数据
无连接，一般一次连接只处理一个请求，结束后主动释放连接，但在 HTTP1.1 中可以使用 keep-alive 来复用相同的 TCP 连接发送多个请求
无状态，客户端向服务器发送 HTTP 请求之后，服务器会给我们发送数据过来，但不会记录任何信息。所以 Cookie、Session 产生了。

6. HTTP 状态码

1xx（临时响应）
2xx（成功）
3xx（重定向）：表示要完成请求需要进一步操作
4xx（错误）：表示请求可能出错，妨碍了服务器的处理
5xx（服务器错误）：表示服务器在尝试处理请求时发生内部错误
常见 HTTP 协议的状态码：
200（成功）
302 （重定向）：请求重定向到指定网页
304（未修改）：自从上次请求后，请求的网页未修改过。服务器返回此响应时，不会返回网页内容
401（未授权）：请求要求身份验证
403（禁止）：服务器拒绝请求（比如死循环了，一直访问）
404（未找到）：服务器找不到请求的网页
405 （方法禁用）:Post 请求当成了 Get 请求直接访问
500 （服务器内部错误）：有 bug 导致程序嗝屁了
502 （错误网关）：服务器从上游接到了无效响应
504 （ 网关超时）：nginx 请求超时，请求一直没有返回

7. http 协议和 https 协议区别

端口 ：HTTP 的 URL 由“http://”起始且默认使用端口 80，而 HTTPS 的 URL 由“https://”起始且默认使用端口 443。

安全性和资源消耗： HTTP 协议运行在 TCP 之上，所有传输的内容都是明文，客户端和服务器端都无法验证对方的身份。HTTPS 是运行在 SSL/TLS 之上的 HTTP 协议，SSL/TLS 运行在 TCP 之上。所有传输的内容都经过加密，加密采用对称加密，但对称加密的密钥用服务器方的证书进行了非对称加密。所以说，HTTP 安全性没有 HTTPS 高，但是 HTTPS 比 HTTP 耗费更多服务器资源。

对称加密：密钥只有一个，加密解密为同一个密码，且加解密速度快，典型的对称加密算法有 DES、AES 等；

非对称加密：密钥成对出现（且根据公钥无法推知私钥，根据私钥也无法推知公钥），加密解密使用不同密钥（公钥加密需要私钥解密，私钥加密需要公钥解密），相对对称加密速度较慢，典型的非对称加密算法有 RSA、DSA 等。

8. TCP 和 UDP 有什么区别

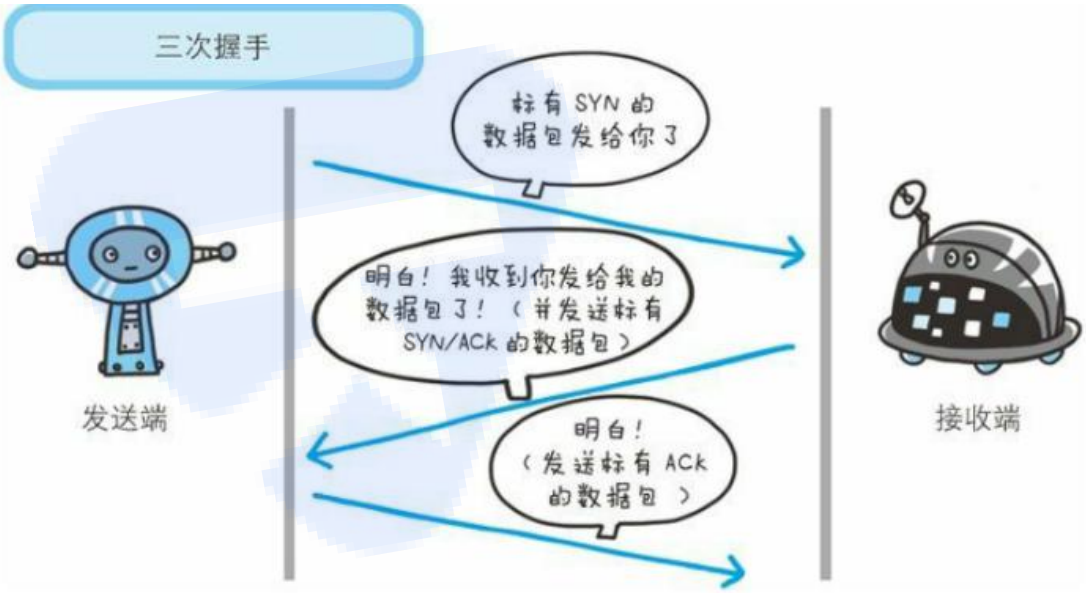
UDP 在传送数据之前不需要先建立连接，远地主机在收到 UDP 报文后，不需要给出任何确认。虽然 UDP 不提供可靠交付，但在某些情况下 UDP 确是一种最有效的工作方式（一般用于即时通信），比如： QQ 语音、 QQ 视频 、直播等等

TCP 提供面向连接的服务。在传送数据之前必须先建立连接，数据传送结束后要释放连接。 TCP 不提供广播或多播服务。由于 TCP 要提供可靠的，面向连接的传输服务（TCP 的可靠体现在 TCP 在传递数据之前，会有三次握手来建立连接，而且在数据传递时，有确认、窗口、重传、拥塞控制机制，在数据传完后，还会断开连接用来节约系统资源），这一难以避免增加了许多开销，如确认，流量控制，计时器以及连接管理等。这不仅使协议数据单元的首部增大很多，还要占用许多处理机资源。TCP 一般用于文件传输、发送和接收邮件、远程登录等场景。

9. TCP 的三次握手过程和四次挥手

三次握手

下面的两个机器人通过 3 次握手确定了对方能正确接收和发送消息



客户端 - 发送带有 SYN 标志的数据包 - 一次握手 - 服务端
服务端 - 发送带有 SYN/ACK 标志的数据包 - 二次握手 - 客户端
客户端 - 发送带有带有 ACK 标志的数据包 - 三次握手 - 服务端

为什么要三次握手?

三次握手的目的是建立可靠的通信信道，说到通讯，简单来说就是数据的发送与接收，而三次握手最主要的目的就是双方确认自己与对方的发送与接收是正常的。

第一次握手：Client 什么都不能确认；Server 确认了对方发送正常，自己接收正常

第二次握手：Client 确认了：自己发送、接收正常，对方发送、接收正常；Server 确认了：对方发送正常，自己接收正常

第三次握手：Client 确认了：自己发送、接收正常，对方发送、接收正常；Server 确认了：自己发送、接收正常，对方发送、接收正常

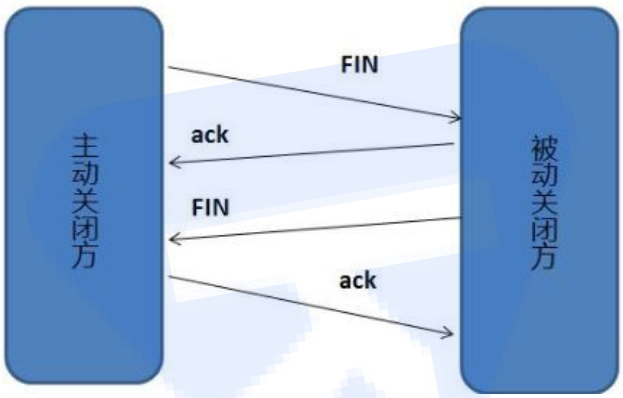
所以三次握手就能确认双发收发功能都正常，缺一不可。

第 2 次握手传回了 ACK，为什么还要传回 SYN？

接收端传回发送端所发送的 ACK 是为了告诉客户端，我接收到的信息确实就是你所发送的信号了，这表明从客户端到服务端的通信是正常的。而回传 SYN 则是为了建立并确认从服务端到客户端的通信。”

SYN 同步序列编号(Synchronize Sequence Numbers) 是 TCP/IP 建立连接时使用的握手信号。在客户机和服务器之间建立正常的 TCP 网络连接时，客户机首先发出一个 SYN 消息，服务器使用 SYN-ACK 应答表示接收到了这个消息，最后客户机再以 ACK(Acknowledgement）消息响应。这样在客户机和服务器之间才能建立起可靠的 TCP 连接，数据才可以在客户机和服务器之间传递。

为什么要四次挥手



任何一方都可以在数据传送结束后发出连接释放的通知，待对方确认后进入半关闭状态。当另一方也没有数据再发送的时候，则发出连接释放通知，对方确认后就完全关闭了 TCP 连接。

举个例子：A 和 B 打电话，通话即将结束后，A 说“我没啥要说的了”，B 回答“我知道了”，但是 B 可能还会有要说的话，A 不能要求 B 跟着自己的节奏结束通话，于是 B 可能又巴拉巴拉说了一通，最后 B 说“我说完了”，A 回答“知道了”，这样通话才算结束。

10. 在浏览器中输入 url 地址到显示主页的过程

DNS 解析

TCP 连接

发送 HTTP 请求

服务器处理请求并返回 HTTP 报文

浏览器解析渲染页面

连接结束

11. 什么是长连接、短连接？

在 HTTP/1.0 中默认使用短连接。也就是说，客户端和服务器每进行一次 HTTP 操作，就建立一次连接，任务结束就中断连接。当客户端浏览器访问的某个 HTML 或其他类型的 Web 页中包含有其他的 Web 资源（如 JavaScript 文件、图像文件、CSS 文件等），每遇到这样一个 Web 资源，浏览器就会重新建立一个 HTTP 会话。

而从 HTTP/1.1 起，默认使用长连接，用以保持连接特性。使用长连接的 HTTP 协议，会在响应头加入这行代码：

Connection:keep-alive

Copy to clipboardErrorCopied

在使用长连接的情况下，当一个网页打开完成后，客户端和服务器之间用于传输 HTTP 数据的 TCP 连接不会关闭，客户端再次访问这个服务器时，会继续使用这一条已经建立的连接。Keep-Alive 不会永久保持连接，它有一个保持时间，可以在不同的服务器软件（如 Apache）中设定这个时间。实现长连接需要客户端和服务端都支持长连接。

什么时候用长连接，短连接？

长连接多用于操作频繁，点对点的通讯，而且连接数不能太多情况，。每个 TCP 连接都需要三步握手，这需要时间，如果每个操作都是先连接，再操作的话那么处理速度会降低很多，所以每个操作完后都不断开，次处理时直接发送数据包就 OK 了，不用建立 TCP 连接。例如：数据库的连接用长连接， 如果用短连接频繁的通信会造成 socket 错误，而且频繁的 socket 创建也是对资源的浪费。

而像 WEB 网站的 http 服务一般都用短链接，因为长连接对于服务端来说会耗费一定的资源，而像 WEB 网站这么频繁的成千上万甚至上亿客户端的连接用短连接会更省一些资源，如果用长连接，而且同时有成千上万的用户，如果每个用户都占用一个连接的话，那可想而知吧。所以并发量大，但每个用户无需频繁操作情况下需用短连好。