面试-计算机网络

github: https://github.com/Harry-Yu-Shuhang

2025 年 1 月 11 日

1 基础篇

1.1 TCP/IP 网络模型有哪几层?

1.1.1 应用层

最上层的，也是我们能直接接触到的就是应用层（Application Layer），我们电脑或手机使用的应用软件都是在应用层实现。

1.1.2 传输层

应用层的数据包会传给传输层，传输层（Transport Layer）是为应用层提供网络支持的。

在传输层会有两个传输协议，分别是 **TCP** 和 **UDP**。

UDP 相对来说就很简单，简单到只负责发送数据包，不保证数据包是否能抵达对方，但它实时性相对更好，传输效率也高。TCP 相比 UDP 多了很多特性，比如流量控制、超时重传、拥塞控制等，这些都是为了保证数据包能可靠地传输给对方。

当设备作为接收方时，传输层则要负责把数据包传给应用，但是一台设备上可能会有很多应用在接收或者传输数据，因此需要用一个编号将应用区分开来，这个编号就是**端口**。比如 80 端口通常是 Web 服务器用的，22 端口通常是远程登录服务器用的。

1.1.3 网络层

我们不希望传输层协议处理太多的事情，只需要服务好应用即可，让其作为应用间数据传输的媒介，帮助实现应用到应用的通信，而实际的传输功能就交给下一层，也就是网络层。(Internet Layer). 网络层最常使用的是 IP 协议（Internet Protocol）。

1.1.4 网络接口层

生成了 IP 头部之后，接下来要交给网络接口层（Link Layer）在 IP 头部的前面加上 MAC 头部，并封装成数据帧（Data frame）发送到网络上。

以太网就是一种在「局域网」内，把附近的设备连接起来，使它们之间可以进行通讯的技术。在以太网进行通讯要用到 MAC 地址。

1.1.5 总结

示意图如下。

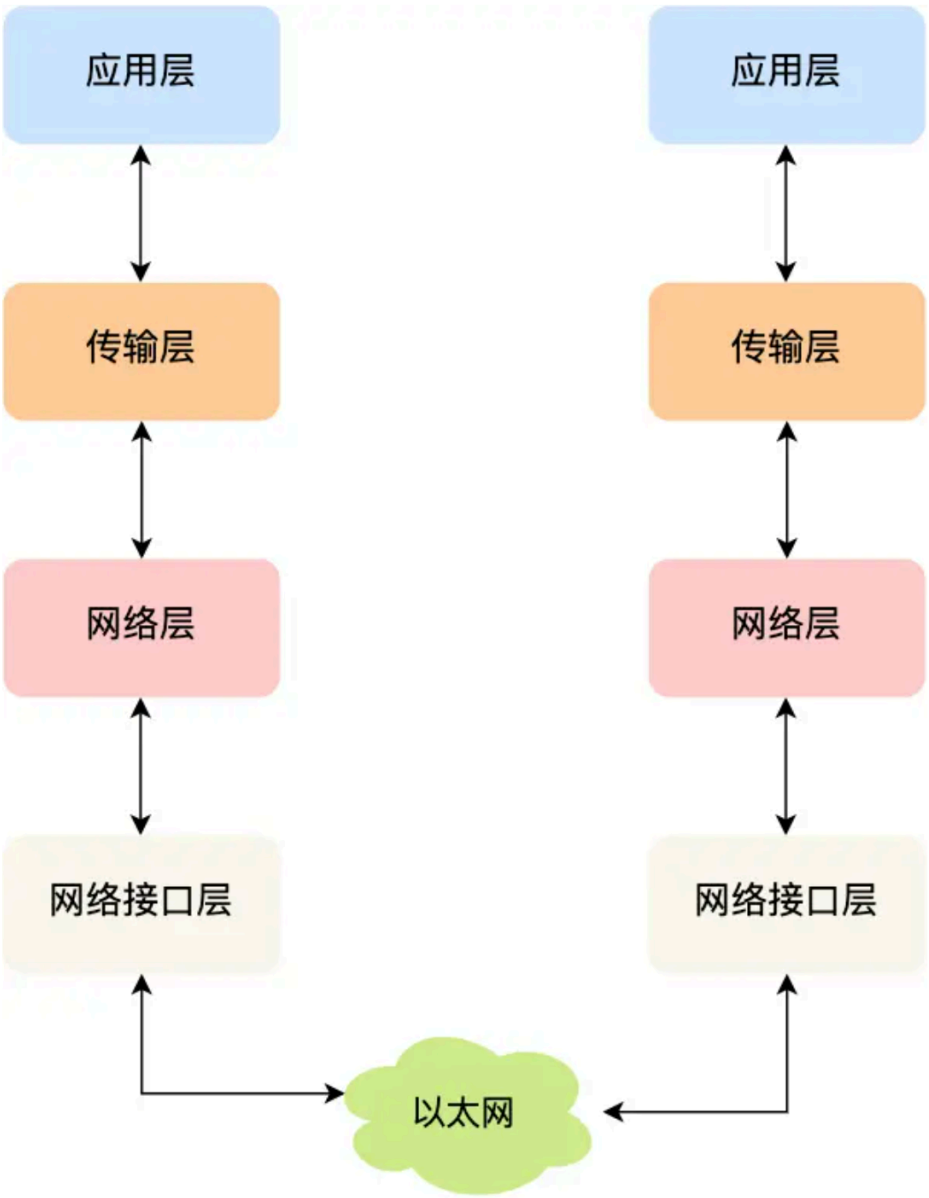

图 1: TCP/IP 结构图

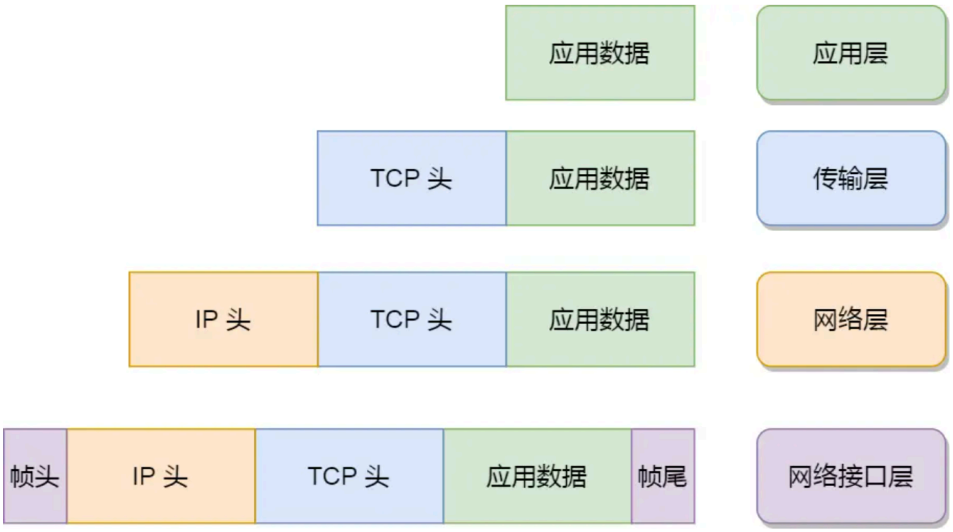

图 2: 每一层的封装格式

1.2 真实地址查询——DNS

通过浏览器解析 URL 并生成 HTTP 消息后，需要委托操作系统将消息发送给 Web 服务器。但是服务器域名对应的 ip 地址很难记，因此 DNS 服务器保存了 web 服务器域名与 ip 地址的对应关系，就好像电话号码记不住，就在通讯录里添加一个联系人。过程就是只指路不带路。

示意图如图 3。

2 http 篇

2.1 安全与幂等

在 HTTP 协议里，所谓的「安全」是指请求方法不会「破坏」服务器上的资源。

所谓的「幂等」，意思是多次执行相同的操作，结果都是「相同」的。

2.2 http 和 https 的区别

HTTP 是超文本传输协议，信息是明文传输，存在安全风险的问题。HTTP 是超文本传输协议，信息是明文传输，存在安全风险的问题。

HTTP 连接建立相对简单，TCP 三次握手之后便可进行 HTTP 的报文传输。HTTP 连接建立相对简单，TCP 三次握手之后便可进行 HTTP 的报文传输。

两者的默认端口不一样，HTTP 默认端口号是 80，HTTPS 默认端口号是 443。

HTTPS 协议需要向 CA（证书权威机构）申请数字证书，来保证服务器的身份是可信的。

2.3 HTTP 和 RPC

主流的 HTTP/1.1 传输 json，有很多冗余字段。RPC 可以用 Protobuf 或其他序列化协议去保存结构体数据，同时也不需要像 HTTP 那样考虑各种浏览器行为，因此性能也会更好一些，这也是在公司内部微服务中抛弃 HTTP，选择使用 RPC 的最主要原因。

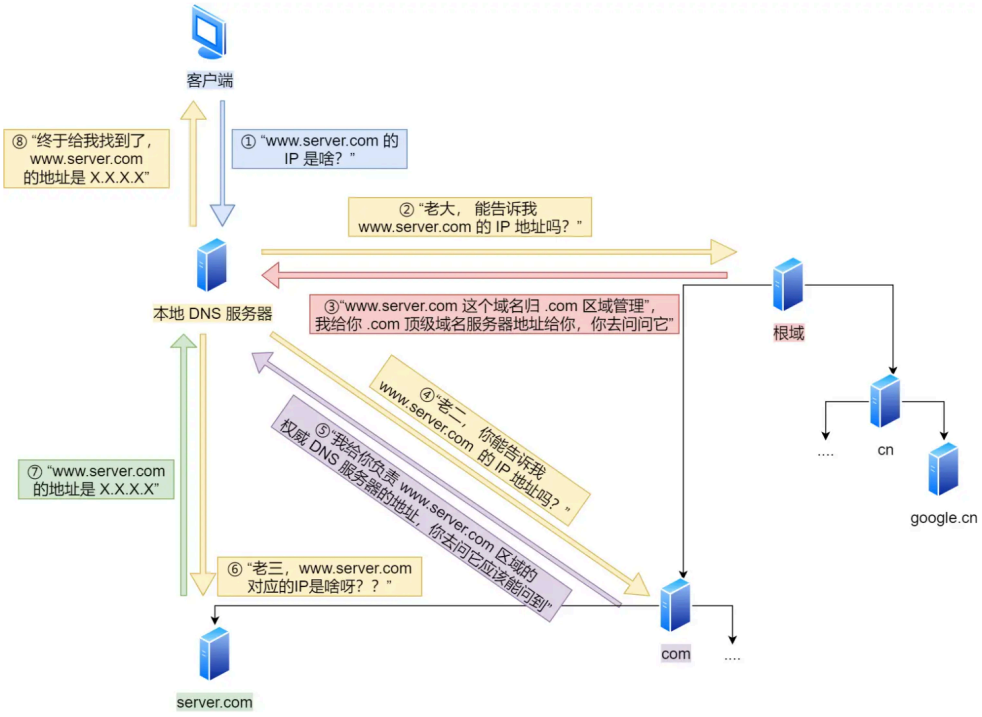

图 3: DNS 解析过程示意图

HTTP/2.0 是 2015 年才出的，性能可能比很多 RPC 协议还要好，甚至连 gRPC 底层都直接用的 HTTP/2。

2.4 为什么要用 Websocket

2.4.1 HTTP 轮询

怎么样才能在用户不做任何操作的情况下，网页能收到消息并发生变更?

最常见的解决方案是，网页的前端代码里不断定时发 HTTP 请求到服务器，服务器收到请求后给客户端响应消息。

用这种方式的场景有很多，最常见的就是扫码登录。

比如，某信公众号平台，登录页面二维码出现之后，前端网页根本不知道用户扫没扫，于是不断去向后端服务器询问，看有没有人扫过这个码。

但是，这样会消耗带宽，使用起来也很卡顿。如何改进? 于是就有了长轮询。

2.4.2 长轮询

我们知道，HTTP 请求发出后，一般会给服务器留一定的时间做响应，比如 3 秒，规定时间内没返回，就认为是超时。

如果我们的 HTTP 请求将超时设置的很大，比如 30 秒，在这 30 秒内只要服务器收到了扫码请求，就立马返回给客户端网页。如果超时，那就立马发起下一次请求。

像这种发起一个请求，在较长时间内等待服务器响应的机制，就是所谓的长轮询机制。们常用的消息队列 RocketMQ 中，消费者去取数据时，也用到了这种方式。

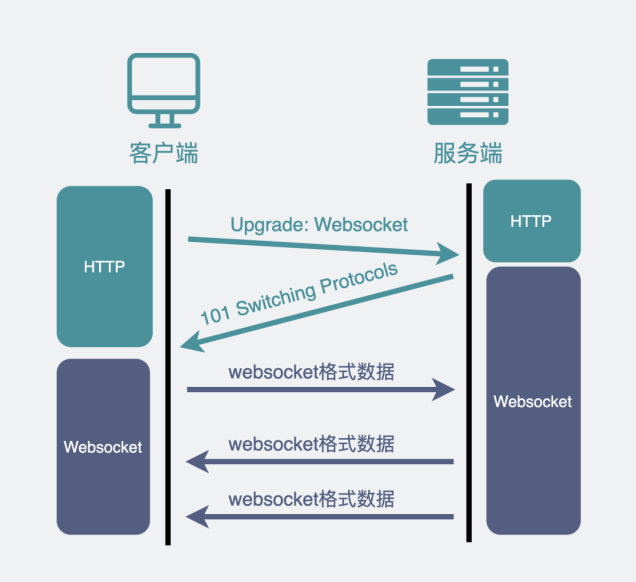

图 4: Websocket 请求示意图

上面提到的两种解决方案（不断轮询和长轮询），本质上，其实还是客户端主动去取数据。对于像扫码登录这样的简单场景还能用用。但如果是网页游戏呢，游戏一般会有大量的数据需要从服务器主动推送到客户端。

2.4.3 WebSocket

建立 websocket 之前，为了兼容性，会先用 http 请求发送升级成 websocket 的请求。示意图如图 4。WebSocket 只有在建立连接时才用到了 HTTP，升级完成之后就跟 HTTP 没有任何关系了。

这就好像你喜欢的女生通过你要到了你大学室友的微信，然后他们自己就聊起来了。你能说这个女生是通过你去跟你室友沟通的吗？不能。你跟 HTTP 一样，都只是个工具人。

2.4.4 WebSocket 的使用场景

它适用于需要服务器和客户端（浏览器）频繁交互的大部分场景，比如网页/小程序游戏，网页聊天室，以及一些类似飞书这样的网页协同办公软件。

3 TCP 篇

3.1 TCP 和 UDP 区别

3.1.1 连接

TCP 是面向连接的传输层协议，传输数据前先要建立连接。

UDP 是不需要连接，即刻传输数据。

3.1.2 服务对象

TCP 是一对一的两点服务，即一条连接只有两个端点。

UDP 支持一对一、一对多、多对多的交互通信

3.1.3 可靠性

TCP 是可靠交付数据的，数据可以无差错、不丢失、不重复、按序到达。

UDP 是尽最大努力交付，不保证可靠交付数据。但是我们可以基于 UDP 传输协议实现一个可靠的传输协议，比如 QUIC 协议。

TCP 是打电话，UDP 是发短信

3.1.4 拥塞控制、流量控制

TCP 有拥塞控制和流量控制机制，保证数据传输的安全性。

UDP 则没有，即使网络非常拥堵了，也不会影响 UDP 的发送速率。

3.1.5 首部开销

TCP 首部长度较长，会有一定的开销，首部在没有使用「选项」字段时是 20 个字节，如果使用了「选项」字段则会变长的。

UDP 首部只有 8 个字节，并且是固定不变的，开销较小。

3.1.6 传输方式

TCP 是流式传输，没有边界，但保证顺序和可靠。

UDP 是一个包一个包的发送，是有边界的，但可能会丢包和乱序。

3.1.7 分片不同

TCP 的数据大小如果大于 MSS 大小，则会在传输层进行分片，目标主机收到后，也同样在传输层组装 TCP 数据包，如果中途丢失了一个分片，只需要传输丢失的这个分片。

UDP 的数据大小如果大于 MTU 大小，则会在 IP 层进行分片，目标主机收到后，在 IP 层组装完数据，接着再传给传输层。

3.2 TCP 和 UDP 可以使用同一个端口吗?

可以。

在数据链路层中，通过 MAC 地址来寻找局域网中的主机。在网际层中，通过 IP 地址来寻找网络中互连的主机或路由器。在传输层中，需要通过端口进行寻址，来识别同一计算机中同时通信的不同应用程序。

所以，传输层的「端口号」的作用，是为了区分同一个主机上不同应用程序的数据包。

传输层有两个传输协议分别是 TCP 和 UDP，在内核中是两个完全独立的软件模块。

当主机收到数据包后，可以在 IP 包头的「协议号」字段知道该数据包是 TCP/UDP，所以可以根据这个信息确定送给哪个模块（TCP/UDP）处理，送给 TCP/UDP 模块的报文根据「端口号」确定送给哪个应用程序处理。

因此，TCP/UDP 各自的端口号也相互独立，如 TCP 有一个 80 号端口，UDP 也可以有一个 80 号端口，二者并不冲突。

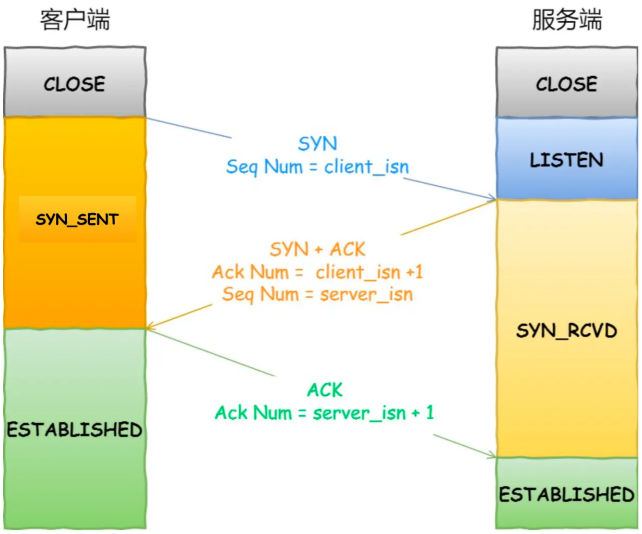

图 5: TCP 三次握手示意图

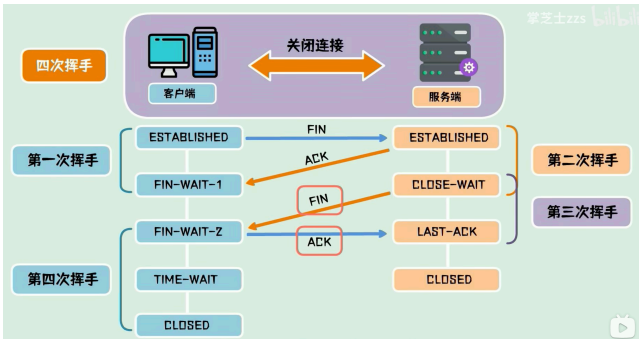

图 6: TCP 四次挥手示意图

3.3 TCP 三次握手/四次挥手

如图 5.

为什么是三次握手？不是两次？为了在不可靠的信道建立可靠的连接。比如客户端发一个 SYN 数据包，因为网络阻塞，数据包没发过去，因此又发了一个 SYN，这次成功了，服务端回复 SYN+ACK。如果此时直接建立连接，则之前阻塞的 SYN 到达服务端后会再建立一个新的连接，把历史理解为创建。

断开连接需要四次挥手，示意图如图 6。服务端在等待 ack 超时的情况下会重发 FIN 包。