计网八股（1）

jask

09/09/2024

HTTP 基本概念

HTTP 是超文本传输协议，也就是 HyperText Transfer Protocol。

五大类状态码：

1xx： 提示信息，表示协议处理的中间状态，还需要后续操作。

2xx： 成功，报文已经收到并被正确处理。

3xx： 重定向，资源位置发生变动，需要客户端重新发送请求。

4xx： 客户端错误，请求报文有误，服务器无法处理。

5xx： 服务器错误，服务器在处理请求内部发生错误。

http 常见字段：

Host 字段：

客户端发送请求时，用来服务器的域名。有了 Host 字段，就可以将请求发往「同一台」服务器上的不同网站。

Content-Length 字段：

服务器在返回数据时，会有 Content-Length 字段，表明本次回应的数据长度。

Connection 字段：

Connection 字段最常用于客户端要求服务器使用 TCP 持久连接，以便其他请求复用。

HTTP/1.1 版本的默认连接都是持久连接，但为了兼容老版本的 HTTP，需要指定 Connection 首部字段的值为 Keep-Alive 。Connection: keep-alive

一个可以复用的 TCP 连接就建立了，直到客户端或服务器主动关闭连接。但是，这不是标准字段。

Content-Type 字段：

Content-Type 字段用于服务器回应时，告诉客户端，本次数据是什么格式。

Content-Encoding 字段：

Content-Encoding 字段说明数据的压缩方法。表示服务器返回的数据使用了什么压缩格式。

GET 和 POST 区别

Get 方法的含义是请求从服务器获取资源，这个资源可以是静态的文本、页面、图片视频等。

而 POST 方法则是相反操作，它向 URI 指定的资源提交数据，数据就放在报文的 body 里。

GET 和 POST 都是安全和幂等的吗？

先说明下安全和幂等的概念：

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。

那么很明显 GET 方法就是安全且幂等的，因为它是「只读」操作，无论操作多少次，服务器上的数据都是安全的，且每次的结果都是相同的。

POST 因为是「新增或提交数据」的操作，会修改服务器上的资源，所以是不安全的，且多次提交数据就会创建多个资源，所以不是幂等的。

HTTP 特性

简单、灵活和易于扩展、应用广泛和跨平台

简单

HTTP 基本的报文格式就是 Header+Body，头部信息也是 Key-Value 这样的简单文本形式

灵活和易于拓展

HTTP 协议里的各类请求方法、URI/URL、状态码、头字段等每个组成要求都没有被固定死，都允许开发人员自定义和扩充。

同时 HTTP 由于是工作在应用层（OSI 第七层），则它下层可以随意变化。HTTPS 也就是在 HTTP 与 TCP 层之间增加了 SSL/TLS 安全传输层，HTTP/3 甚至把 TCP 层换成了基于 UDP 的 QUIC。

应用广泛且跨平台

无须赘述

HTTP 缺点

HTTP 协议里有优缺点一体的双刃剑，分别是「无状态、明文传输」，同时还有一大缺点「不安全」。

无状态

无状态的好处，因为服务器不会去记忆 HTTP 的状态，所以不需要额外的资源来记录状态信息，这能减轻服务器的负担，能够把更多的 CPU 和内存用来对外提供服务。

无状态的坏处，既然服务器没有记忆能力，它在完成有关联性的操作时会非常麻烦。

例如登录-> 添加购物车-> 下单-> 结算-> 支付，这系列操作都要知道用户的身份才行。但服务器不知道这些请求是有关联的，每次都要问一遍身份信息。

对于无状态的问题，解法方案有很多种，其中比较简单的方式用 Cookie 技术。

Cookie 通过在请求和响应报文中写入 Cookie 信息来控制客户端的状态。

相当于，在客户端第一次请求后，服务器会下发一个装有客户信息的「小贴纸」，后续客户端请求服务器的时候，带上「小贴纸」，服务器就能认得了了。

明文传输

明文意味着在传输过程中的信息，是可方便阅读的，通过浏览器的 F12 控制台或 Wireshark 抓包都可以直接肉眼查看，为我们调试工作带了极大的便利性。

但是这正是这样，HTTP 的所有信息都暴露在了光天化日下，相当于信息裸奔。在传输的漫长的过程中，信息的内容都毫无隐私可言，很容易就能被窃取，如果里面有你的账号密码信息，那你号没了。

不安全

1. 通信明文，内容可能被窃听
2. 不验证通信方的身份，可能遭遇伪装
3. 无法证明报文的完整，可能已经被篡改

- HTTP 的安全问题，可以用 HTTPS 的方式解决，也就是通过引入 SSL/TLS 层，使得在安全上达到了极致 *。

性能

HTTP 协议是基于 TCP/IP，并且使用了「请求 - 应答」的通信模式，所以性能的关键就在这两点里。

长连接

早期 HTTP/1.0 性能上的一个很大的问题，那就是每发起一个请求，都要新建一次 TCP 连接（三次握手），而且是串行请求，做了无谓的 TCP 连接建立和断开，增加了通信开销。

为了解决上述 TCP 连接问题,HTTP/1.1 提出了长连接的通信方式，也叫持久连接。这种方式的好处在于减少了 TCP 连接的重复建立和断开所造成的额外开销，减轻了服务器端的负载。

持久连接的特点是，只要任意一端没有明确提出断开连接，则保持 TCP 连接状态。

管道网络传输

HTTP/1.1 采用了长连接的方式，这使得管道（pipeline）网络传输成为了可能。

即可在同一个 TCP 连接里面，客户端可以发起多个请求，只要第一个请求发出去了，不必等其回来，就可以发第二个请求出去，可以减少整体的响应时间。

举例来说，客户端需要请求两个资源。以前的做法是，在同一个 TCP 连接里面，先发送 A 请求，然后等待服务器做出回应，收到后再发出 B 请求。管道机制则是允许浏览器同时发出 A 请求和 B 请求。

但是服务器还是按照顺序，先回应 A 请求，完成后再回应 B 请求。要是前面的回应特别慢，后面就会有许多请求排队等着。这称为「队头堵塞」。

队头阻塞

「请求 - 应答」的模式加剧了 HTTP 的性能问题。

因为当顺序发送的请求序列中的一个请求因为某种原因被阻塞时，在后面排队的所有请求也一同被阻塞了，会招致客户端一直请求不到数据，这也就是「队头阻塞」。好比上班的路上塞车。

HTTP 与 HTTPS

有哪些区别?

1.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。HTTPS 则解决 HTTP 不安全的缺陷，在 TCP 和 HTTP 网络层之间加入了 SSL/TLS 安全协议，使得报文能够加密传输。

2.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。而 HTTPS 在 TCP 三次握手之后，还需进行 SSL/TLS 的握手过程，才可进入加密报文传输。

3.HTTP 的端口号是 80,HTTPS 的端口号是 443。

4.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。

HTTPS 解决了 HTTP 的哪些问题

HTTP 由于是明文传输，所以安全上存在以下三个风险：

窃听风险，比如通信链路上可以获取通信内容，用户号容易没。

篡改风险，比如强制植入垃圾广告，视觉污染，用户眼容易瞎。

冒充风险，比如冒充淘宝网站，用户钱容易没。

HTTPS 在 HTTP 与 TCP 层之间加入了 SSL/TLS 协议，可以很好的解决了上述的风险：

信息加密：交互信息无法被窃取，但你的号会因为「自身忘记」账号而没。

校验机制：无法篡改通信内容，篡改了就不能正常显示，但百度「竞价排名」依然可以搜索垃圾广告。

身份证书：证明淘宝是真的淘宝网，但你的钱还是会因为「剁手」而没。

HTTPS 是如何解决上述的三个风险的

混合加密的方式实现信息的机密性，解决了窃听的风险。

摘要算法的方式来实现完整性，它能够为数据生成独一无二的「指纹」，指纹用于校验数据的完整性，解决了篡改的风险。

将服务器公钥放入到数字证书中，解决了冒充的风险。

混合加密

HTTPS 采用的是对称加密和非对称加密结合的「混合加密」方式：

在通信建立前采用非对称加密的方式交换「会话秘钥」，后续就不再使用非对称加密。

在通信过程中全部使用对称加密的「会话秘钥」的方式加密明文数据。

采用「混合加密」的方式的原因：

对称加密只使用一个密钥，运算速度快，密钥必须保密，无法做到安全的密钥交换。

非对称加密使用两个密钥：公钥和私钥，公钥可以任意分发而私钥保密，解决了密钥交换问题但速度慢。

摘要算法

摘要算法用来实现完整性，能够为数据生成独一无二的「指纹」，用于校验数据的完整性，解决了篡改的风险。

客户端在发送明文之前会通过摘要算法算出明文的「指纹」，发送的时候把「指纹 + 明文」一同加密成密文后，发送给服务器，服务器解密后，用相同的摘要算法算出发送过来的明文，通过比较客户端携带的「指纹」和当前算出的「指纹」做比较，若「指纹」相同，说明数据是完整的。

数字证书

客户端先向服务器端索要公钥，然后用公钥加密信息，服务器收到密文后，用自己的私钥解密。

这就存在些问题，如何保证公钥不被篡改和信任度？ 所以这里就需要借助第三方权威机构 CA（数字证书认证机构），将服务器公钥放在数字证书（由数字证书认证机构颁发）中，只要证书是可信的，公钥就是可信的。

HTTPS 是如何建立连接的？其间交互了什么？

SSL/TLS 协议基本流程：

客户端向服务器索要并验证服务器的公钥。

双方协商生产「会话秘钥」。

双方采用「会话秘钥」进行加密通信。

前两步也就是 SSL/TLS 的建立过程，也就是握手阶段。

SSL/TLS 的「握手阶段」涉及四次通信：

ClientHello 首先，由客户端向服务器发起加密通信请求，也就是 ClientHello 请求。

在这一步，客户端主要向服务器发送以下信息：

(1) 客户端支持的 SSL/TLS 协议版本，如 TLS 1.2 版本。

(2) 客户端生产的随机数（Client Random），后面用于生产「会话秘钥」。

(3) 客户端支持的密码套件列表，如 RSA 加密算法。

ServerHello 服务器收到客户端请求后，向客户端发出响应，也就是 SeverHello 。服务器回应的内容有如下内容：

(1) 确认 SSL/ TLS 协议版本，如果浏览器不支持，则关闭加密通信。

(2) 服务器生产的随机数（Server Random），后面用于生产「会话秘钥」。

(3) 确认的密码套件列表，如 RSA 加密算法。

(4) 服务器的数字证书。

客户端回应 客户端收到服务器的回应之后，首先通过浏览器或者操作系统中的 CA 公钥，确认服务器的数字证书的真实性。

如果证书没有问题，客户端会从数字证书中取出服务器的公钥，然后使用它加密报文，向服务器发送如下信息：

(1) 一个随机数（pre-master key）。该随机数会被服务器公钥加密。

(2) 加密通信算法改变通知，表示随后的信息都将用「会话秘钥」加密通信。

(3) 客户端握手结束通知，表示客户端的握手阶段已经结束。这一项同时把之前所有内容的发生的数据做个摘要，用来供服务端校验。

上面第一项的随机数是整个握手阶段的第三个随机数，这样服务器和客户端就同时有三个随机数，接着就用双方协商的加密算法，各自生成本次通信的「会话秘钥」。

服务器最后回应 服务器收到客户端的第三个随机数 （ pre-master key ）之后，通过协商的加密算法，计算出本次通信的「会话秘钥」。然后，向客户端发生最后的信息：

（1）加密通信算法改变通知，表示随后的信息都将用「会话秘钥」加密通信。

（2）服务器握手结束通知，表示服务器的握手阶段已经结束。这一项同时把之前所有内容的发生的数据做个摘要，用来供客户端校验。

至此，整个 SSL/TLS 的握手阶段全部结束。接下来，客户端与服务器进入加密通信，就完全是使用普通的 HTTP 协议，只不过用「会话秘钥」加密内容。

HTTP/1.1，HTTP/2，HTTP/3 演变

HTTP/1.1 相比 HTTP/1.0 性能上的改进：

使用 TCP 长连接的方式改善了 HTTP/1.0 短连接造成的性能开销。

支持管道（pipeline）网络传输，只要第一个请求发出去了，不必等其回来，就可以发第二个请求出去，可以减少整体的响应时间。

但 HTTP/1.1 还是有性能瓶颈：

请求 / 响应头部（Header）未经压缩就发送，首部信息越多延迟越大。只能压缩 Body 的部分；

发送冗长的首部。每次互相发送相同的首部造成的浪费较多；

服务器是按请求的顺序响应的，如果服务器响应慢，会招致客户端一直请求不到数据，也就是队头阻塞；

没有请求优先级控制；

请求只能从客户端开始，服务器只能被动响应。

HTTP/2 的优化

头部压缩：

HTTP/2 会压缩头（Header）如果你同时发出多个请求，他们的头是一样的或是相似的，那么，协议会帮你消除重复的部分。

这就是所谓的 HPACK 算法：在客户端和服务器同时维护一张头信息表，所有字段都会存入这个表，生成一个索引号，以后就不发送同样字段了，只发送索引号，这样就提高速度了。

二进制格式

HTTP/2 不再像 HTTP/1.1 里的纯文本形式的报文，而是全面采用了二进制格式，头信息和数据体都是二进制，并且统称为帧（frame）：头信息帧和数据帧。

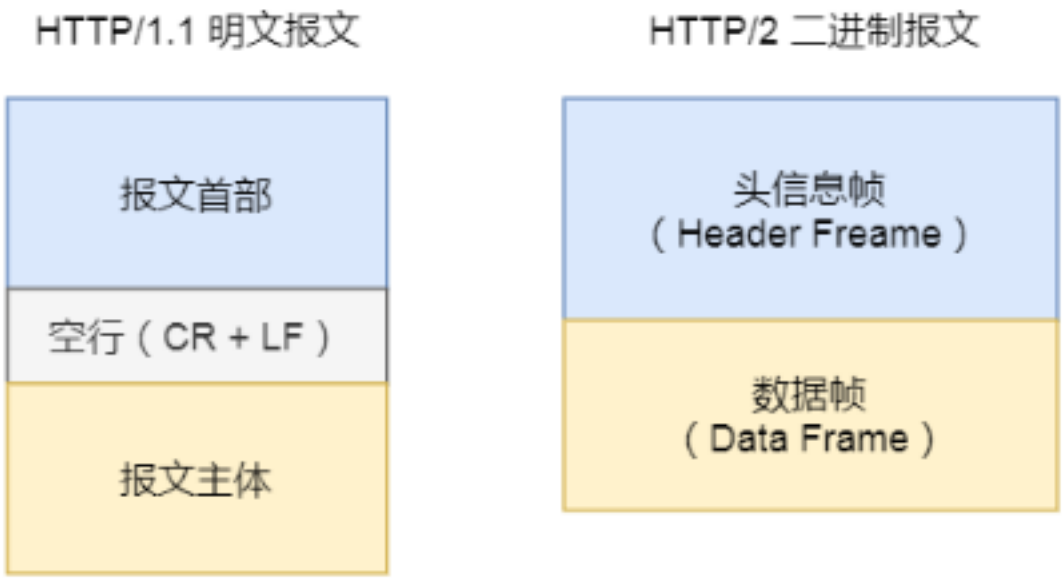

Figure 1: 格式比较

数据流

HTTP/2 的数据包不是按顺序发送的，同一个连接里面连续的数据包，可能属于不同的回应。因此，必须要对数据包做标记，指出它属于哪个回应。

每个请求或回应的所有数据包，称为一个数据流（Stream）。每个数据流都标记着一个独一无二的编号，其中规定客户端发出的数据流编号为奇数，服务器发出的数据流编号为偶数。

客户端还可以指定数据流的优先级。优先级高的请求，服务器就先响应该请求。

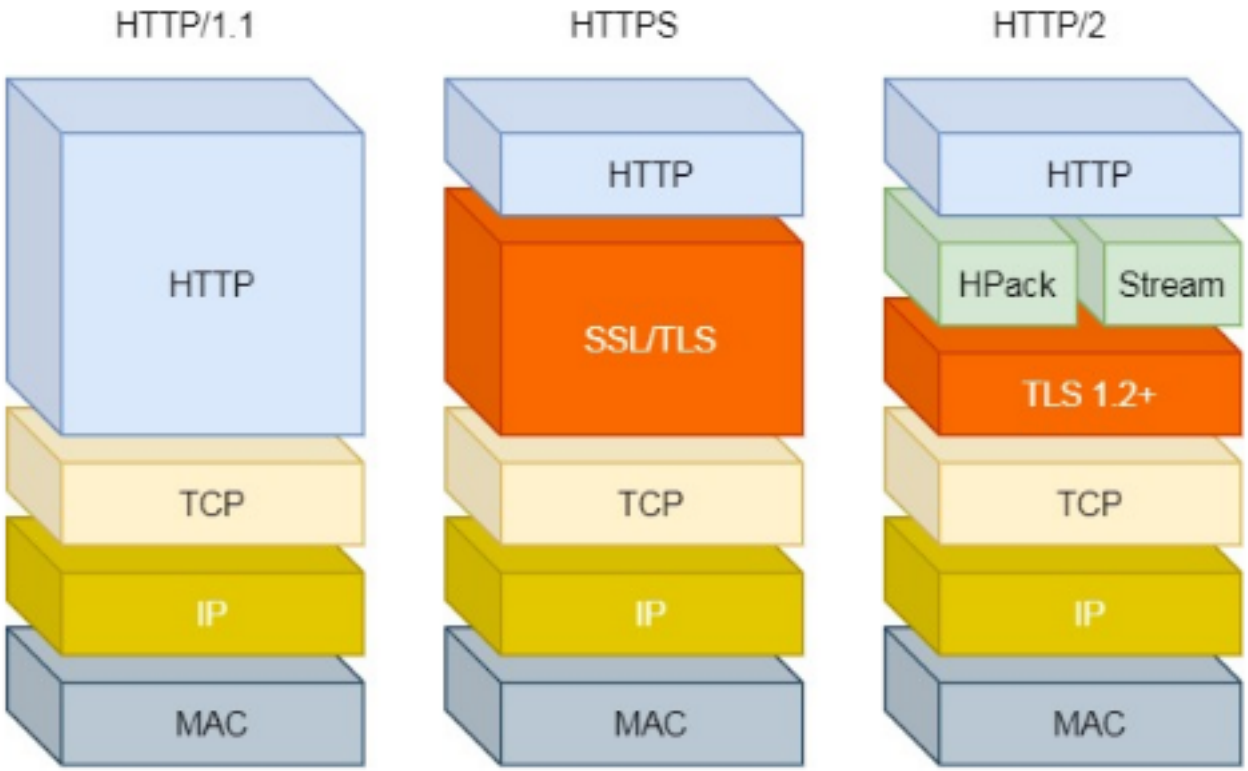

Figure 2：三种请求形式

多路复用

HTTP/2 是可以在一个连接中并发多个请求或回应，而不用按照顺序一一对应。

移除了 HTTP/1.1 中的串行请求，不需要排队等待，也就不会再出现「队头阻塞」问题，降低了延迟，大幅度提高了连接的利用率。

服务器推送

HTTP/2 还在一定程度上改善了传统的「请求 - 应答」工作模式，服务不再是被动地响应，也可以主动向客户端发送消息。

举例来说，在浏览器刚请求 HTML 的时候，就提前把可能会用到的 JS、CSS 文件等静态资源主动发给客户端，减少延时的等待，也就是服务器推送（Server Push，也叫 Cache Push）。

HTTP/2 的缺陷？HTTP/3 的优化？

HTTP/2 主要的问题在于，多个 HTTP 请求在复用一个 TCP 连接，下层的 TCP 协议是不知道有多少个 HTTP 请求的。所以一旦发生了丢包现象，就会触发 TCP 的重传机制，这样在一个 TCP 连接中的所有的 HTTP 请求都必须等待这个丢了的包被重传回来。

HTTP/1.1 中的管道（pipeline）传输中如果有一个请求阻塞了，那么队列后请求也统统被阻塞住了。

HTTP/2 多个请求复用一个 TCP 连接，一旦发生丢包，就会阻塞住所有的 HTTP 请求。

这都是基于 TCP 传输层的问题，所以 HTTP/3 把 HTTP 下层的 TCP 协议改成了 UDP！

DP 发生是不管顺序，也不管丢包的，所以不会出现 HTTP/1.1 的队头阻塞和 HTTP/2 的一个丢包全部重传问题。

大家都知道 UDP 是不可靠传输的，但基于 UDP 的 QUIC 协议可以实现类似 TCP 的可靠性传输。

QUIC 有自己的一套机制可以保证传输的可靠性的。当某个流发生丢包时，只会阻塞这个流，其他流不会受到影响。

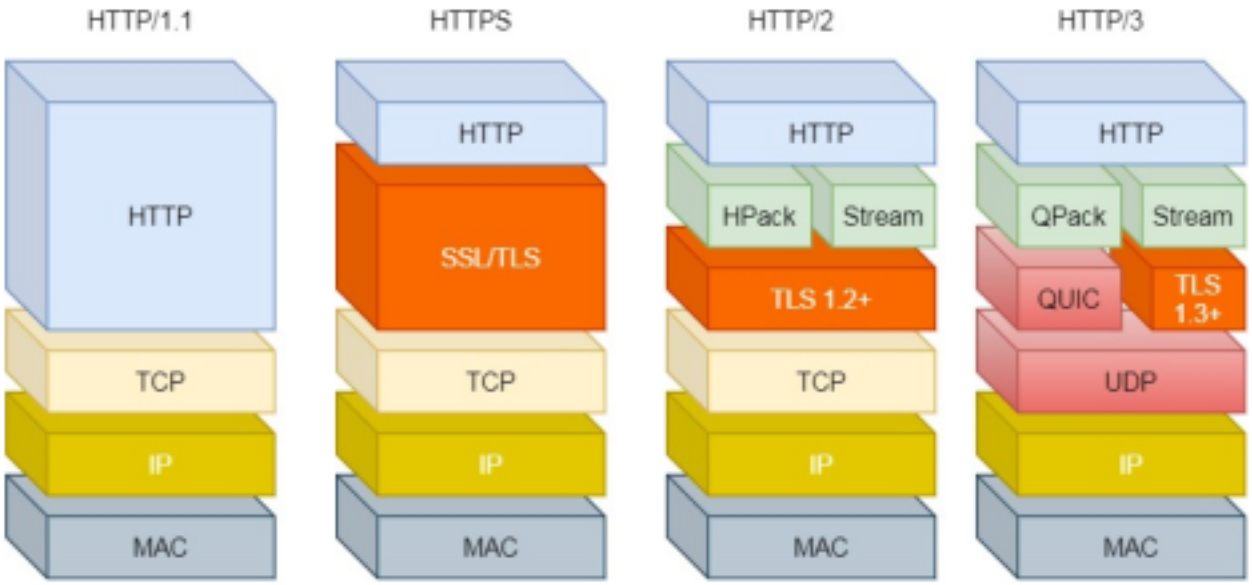

Figure 3: 全部 HTTP 协议的比较

TLS3 升级成了最新的 1.3 版本，头部压缩算法也升级成了 QPack 。

HTTPS 要建立一个连接，要花费 6 次交互，先是建立三次握手，然后是 TLS/1.3 的三次握手。

QUIC 直接把以往的 TCP 和 TLS/1.3 的 6 次交互合并成了 3 次，减少了交互次数。

所以，QUIC 是一个在 UDP 之上的伪 TCP + TLS + HTTP/2 的多路复用的协议。

为什么 ssl 握手是 4 次

SSL/TLS 1.2 需要 4 握手，需要 2 个 RTT 的时延，我文中的图是把每个交互分开画了，实际上把他们合在一起发送，就是 4 次握手。

SSL/TLS 1.3 优化了过程，只需要 1 个 RTT 往返时延，也就是只需要 3 次握手。

IP 协议

IP 在 TCP/IP 参考模型中处于第三层，也就是网络层。

网络层的主要作用是：实现主机与主机之间的通信，也叫点对点（end to end）通信。

IP 的作用是主机之间通信用的，而 MAC 的作用则是实现「直连」的两个设备之间通信，而 IP 则负责在「没有直连」的两个网络之间进行通信传输。

源 IP 地址和目标 IP 地址在传输过程中是不会变化的，只有源 MAC 地址和目标 MAC 一直在变化。

IP

P 地址（IPv4 地址）由 32 位正整数来表示,IP 地址在计算机是以二进制的方式处理的。而人类为了方便记忆采用了点分十进制的标记方式，也就是将 32 位 IP 地址以每 8 位为组，共分为 4 组，每组以「. 」隔开，再将每组转换成十进制。

所以 IP 协议最大允许 2 的 32 次方个计算机连接到网络。

实际上,IP 地址并不是根据主机台数来配置的，而是以网卡。像服务器、路由器等设备都是有 2 个以上的网卡，也就是它们会有 2 个以上的 IP 地址。

因为会根据一种可以更换 IP 地址的技术 NAT ，使得可连接计算机数超过 43 亿台。NAT 技术后续会进一步讨论和说明。

IP 地址分类成了 5 种类型，分别是 A 类、B 类、C 类、D 类、E 类。

其中对于 A、B、C 类主要分为两个部分，分别是网络号和主机号。

C 类不是 256 而是 254 是因为有两个特殊 IP(主机号全 0，主机号全 1)

主机号全为 1 指定某个网络下的所有主机，用于广播

主机号全为 0 指定某个网络

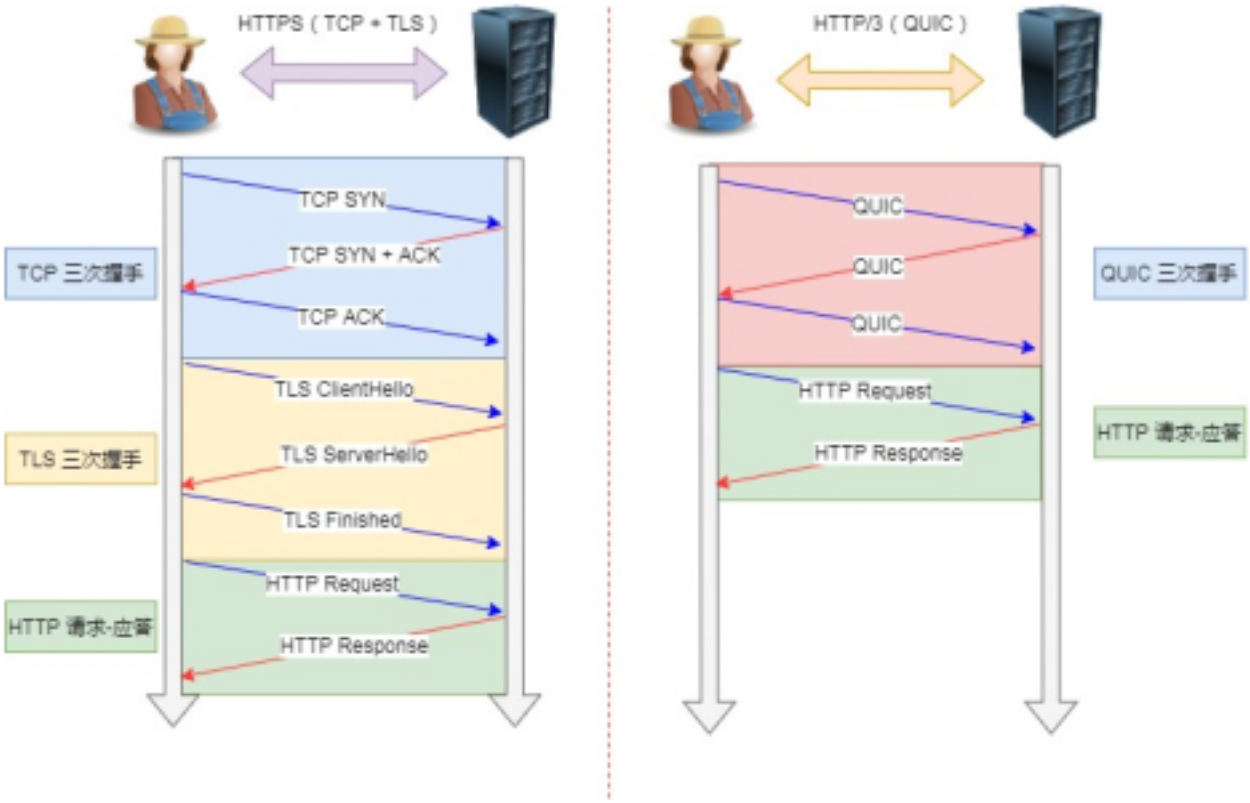

Figure 4: QUIC 与 TCP+TLS 比较

Figure 5: 如图

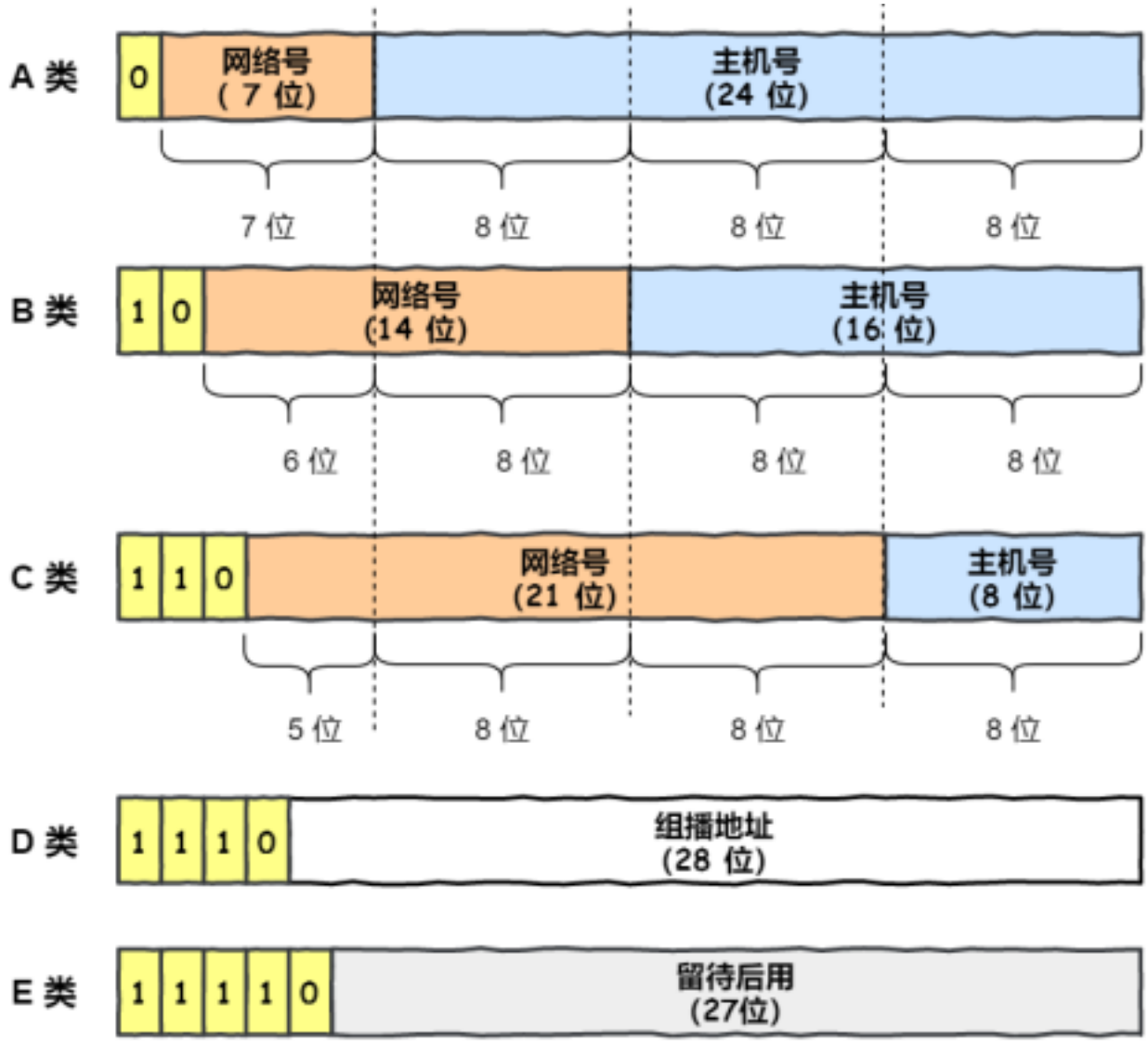

Figure 6: 五类 ip

类别	IP 地址范围	最大主机数
A	0.0.0.0 ~ 127.255.255.255	16777214
B	128.0.0.0 ~ 191.255.255.255	65534
C	192.0.0.0 ~ 223.255.255.255	254

Figure 7: A,B,C 三类

广播地址用于在同一个链路中相互连接的主机之间发送数据包。

广播地址可以分为本地广播和直接广播两种。

在本网络内广播的叫做本地广播。例如网络地址为 192.168.0.0/24 的情况下，广播地址是 192.168.0.255 。因为这个广播地址的 IP 包会被路由器屏蔽，所以不会到达 192.168.0.0/24 以外的其他链路上。

在不同网络之间的广播叫做直接广播。例如网络地址为 192.168.0.0/24 的主机向 192.168.1.255/24 的目标地址发送 IP 包。收到这个包的路由器，将数据转发给 192.168.1.0/24，从而使得所有 192.168.1.1~192.168.1.254 的主机都能收到这个包（由于直接广播有一定的安全问题，多数情况下会在路由器上设置为不转发。）。

而 D 类和 E 类地址是没有主机号的，所以不可用于主机 IP,D 类常被用于多播,E 类是预留的分类，暂时未使用。

多播用于将包发送给特定组内的所有主机。

IP 分类的优点

不管是路由器还是主机解析到一个 IP 地址时候，我们判断其 IP 地址的首位是否为 0，为 0 则为 A 类地址，那么就能很快的找出网络地址和主机地址。简单明了、选路（基于网络地址）简单。

IP 分类的缺点

同一网络下没有地址层次，比如一个公司里用了 B 类地址，但是可能需要根据生产环境、测试环境、开发环境来划分地址层次，而这种 IP 分类是没有地址层次划分的功能，所以这就缺少地址的灵活性。

A、B、C 类有个尴尬处境，就是不能很好的与现实网络匹配。C 类地址能包含的最大主机数量实在太少了，只有 254 个，估计一个网吧都不够用。而 B 类地址能包含的最大主机数量又太多了,6 万多台机器放在一个网络下面，一般的企业基本达不到这个规模，闲着的地址就是浪费。

这两个缺点，都可以在 CIDR 无分类地址解决。

无分类地址 CIDR

正因为 IP 分类存在许多缺点，所以后面提出了无分类地址的方案，即 CIDR 。

这种方式不再有分类地址的概念,32 比特的 IP 地址被划分为两部分，前面是网络号，后面是主机号。

如何划分网络号和主机号

表示形式 a.b.c.d/x ，其中 /x 表示前 x 位属于网络号，x 的范围是 0 ~ 32 ，这就使得 IP 地址更加具有灵活性。

比如 10.100.122.2/24，这种地址表示形式就是 CIDR,/24 表示前 24 位是网络号，剩余的 8 位是主机号。

还有另一种划分网络号与主机号形式，那就是子网掩码，掩码的意思就是掩盖掉主机号，剩余的就是网络号。

将子网掩码和 IP 地址按位计算 AND，就可得到网络号。

为什么要分离网络号和主机号

因为两台计算机要通讯，首先要判断是否处于同一个广播域内，即网络地址是否相同。如果网络地址相同，表明接受方在本网络上，那么可以把数据包直接发送到目标主机。

路由器寻址工作中，也就是通过这样的方式来找到对应的网络号的，进而把数据包转发给对应的网络内。

怎么进行子网划分

在上面我们知道可以通过子网掩码划分出网络号和主机号，那实际上子网掩码还有一个作用，那就是划分子网。子网划分实际上是将主机地址分为两个部分：子网网络地址和子网主机地址。形式如下：

未做子网划分的 ip 地址：网络地址 + 主机地址

做子网划分后的 ip 地址：网络地址 +(子网网络地址 + 子网主机地址)

假设对 C 类地址进行子网划分，网络地址 192.168.1.0，使用子网掩码 255.255.255.192 对其进行子网划分。

C 类地址中前 24 位是网络号，最后 8 位是主机号，根据子网掩码可知从 8 位主机号中借用 2 位作为子网号。

由于子网网络地址被划分成 2 位，那么子网地址就有 4 个，分别是 00、01、10、11，具体划分如下图：

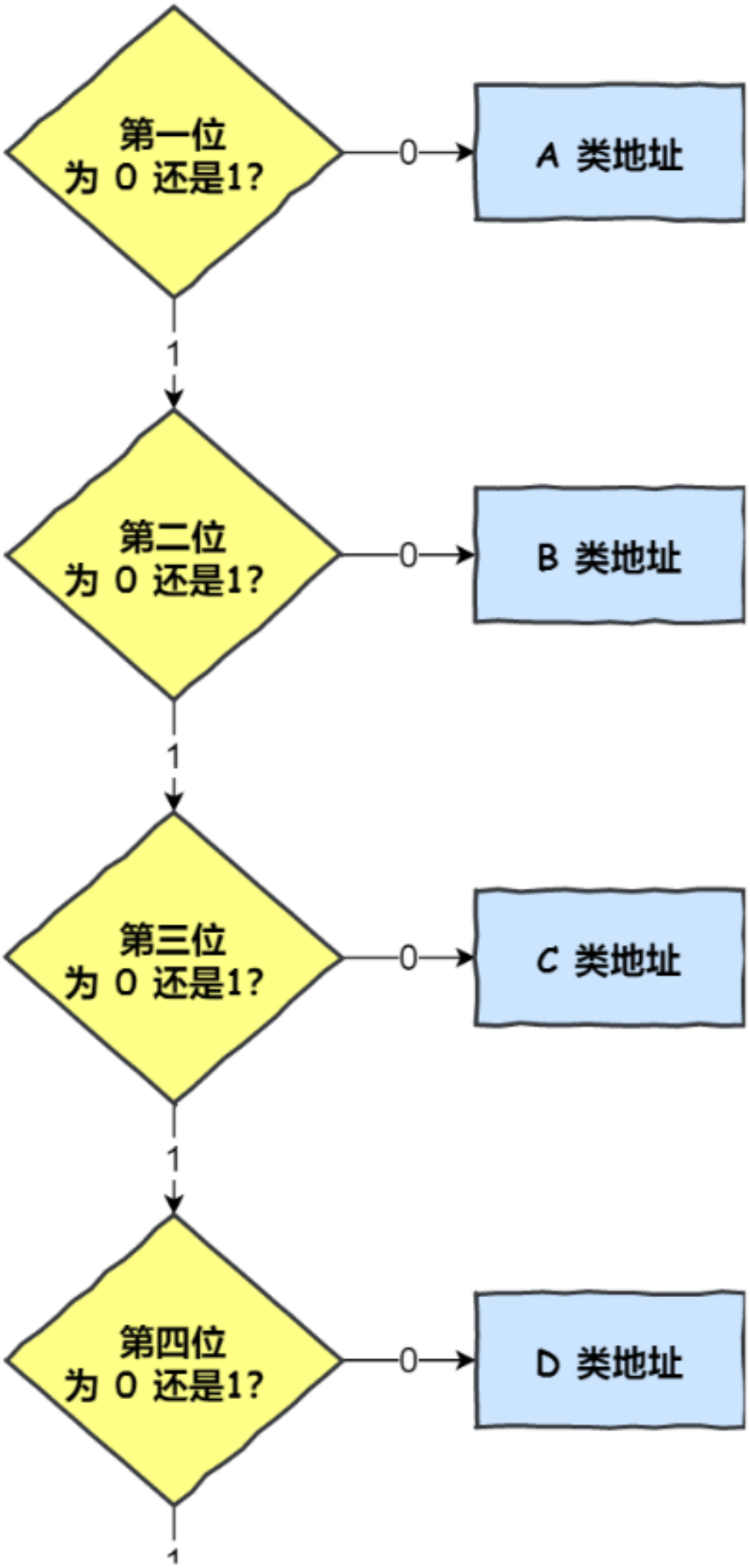
第一位
为 0 还是1?
0
A 类地址
1
第二位
为 0 还是1?
0
B 类地址
1
第三位
为 0 还是1?
0
C 类地址
1
第四位
为 0 还是1?
0
D 类地址
1

Figure 8: 判断

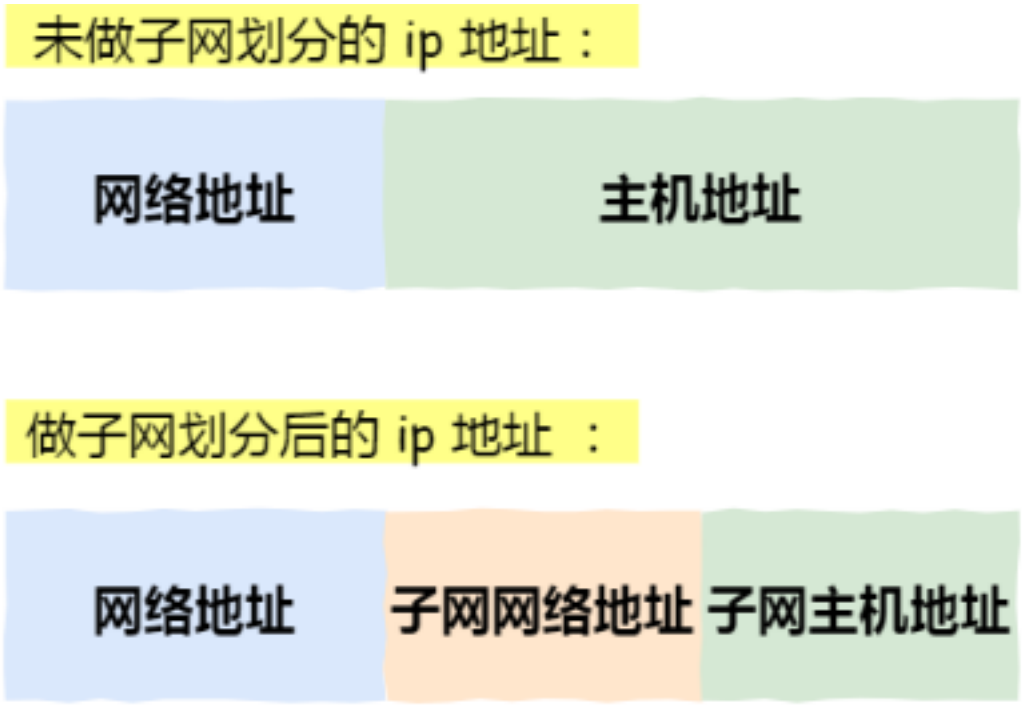

Figure 9: 如图

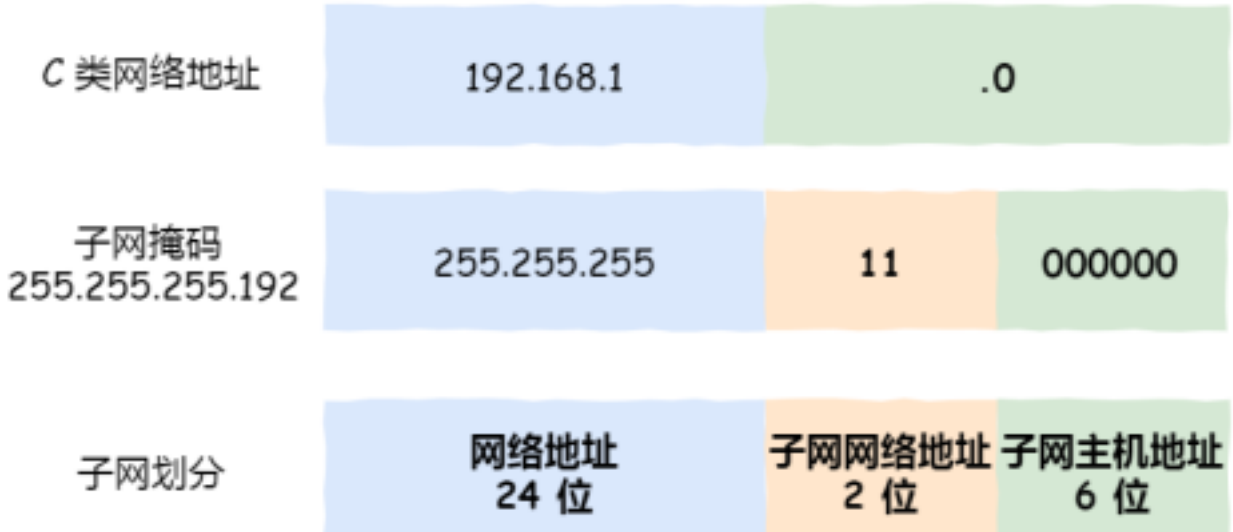

Figure 10: 划分

划分后的 4 个子网如下表格：

子网号	网络地址	主机地址范围	广播地址
0	192.168.1.0	192.168.1.1 ~ 192.168.1.62	192.168.1.63
1	192.168.1.64	192.168.1.65 ~ 192.168.1.126	192.168.1.127
2	192.168.1.128	192.168.1.129 ~ 192.168.1.190	192.168.1.191
3	192.168.1.192	192.168.1.193 ~ 192.168.1.254	192.168.1.255

Figure 11： 划分后

类别	IP 地址范围	最大主机数	私有 IP 地址范围
A	0.0.0.0 ~ 127.255.255.255	16777214	10.0.0.0 ~ 10.255.255.255
B	128.0.0.0 ~ 191.255.255.255	65534	172.16.0.0 ~ 172.31.255.255
C	192.0.0.0 ~ 223.255.255.255	254	192.168.0.0 ~ 192.168.255.255

Figure 12： 如图

公有 ip 和私有 ip

ip 地址与路由控制

IP 地址的网络地址这一部分是用于进行路由控制。

路由控制表中记录着网络地址与下一步应该发送至路由器的地址。在主机和路由器上都会有各自的路由器控制表。

在发送 IP 包时，首先要确定 IP 包首部中的目标地址，再从路由控制表中找到与该地址具有相同网络地址的记录，根据该记录将 IP 包转发给相应的下一个路由器。如果路由控制表中存在多条相同网络地址的记录，就选择相同位数最多的网络地址，也就是最长匹配。

换回地址是不会流向网络

环回地址是在同一台计算机上的程序之间进行网络通信时所使用的一个默认地址。

计算机使用一个特殊的 IP 地址 127.0.0.1 作为环回地址。与该地址具有相同意义的是一个叫做 localhost 的主机名。使用这个 IP 或主机名时，数据包不会流向网络。

ipv6

IPv4 的地址是 32 位的，大约可以提供 42 亿个地址，但是早在 2011 年 IPv4 地址就已经被分配完了。

但是 IPv6 的地址是 128 位的，这可分配的地址数量是大的惊人，说个段子 IPv6 可以保证地球上的每粒沙子都能被分配到一个 IP 地址。

但 IPv6 除了有更多的地址之外，还有更好的安全性和扩展性，说简单点就是 IPv6 相比于 IPv4 能带来更好的网络体验。

但是因为 IPv4 和 IPv6 不能相互兼容，所以不但要我们电脑、手机之类的设备支持，还需要网络运营商对现有的设备进行升级，所以这可能是 IPv6 普及率比较慢的一个原因。

亮点

可自动配置，即使没有 DHCP 服务器也可以自动分配 IP 地址。

包头包首部长度采用固定的值 40 字节，去掉了包头校验和，简化了首部结构，提高性能。

IPv6 有应对伪造 IP 地址的网络安全功能以及防止线路窃听的功能，大大提升了安全性。

IPv6 的地址主要有以下类型地址： 单播地址，用于一对一的通信

组播地址，用于一对多的通信

任播地址，用于通信最近的节点，最近的节点是由路由协议决定

没有广播地址

单播地址类型

对于一对一通信的 IPv6 地址，主要划分了三类单播地址，每类地址的有效范围都不同。

在同一链路单播通信，不经过路由器，可以使用链路本地单播地址,IPv4 没有此类型

在内网里单播通信，可以使用唯一本地地址，相当于 IPv4 的私有 IP

在互联网通信，可以使用全局单播地址，相当于 IPv4 的公有 IP

首部改进

取消了首部校验和字段。因为在数据链路层和传输层都会校验，因此 IPv6 直接取消了 IP 的校验。

取消了分片/重新组装相关字段。分片与重组是耗时的过程,IPv6 不允许在中间路由器进行分片与重组，这种操作只能在源与目标主机，这将大大提高了路由器转发的速度。

取消选项字段。选项字段不再是标准 IP 首部的一部分了，但它并没有消失，而是可能出现在 IPv6 首部中的「下一个首部」指出的位置上。删除该选项字段使的 IPv6 的首部成为固定长度的 40 字节。

IP 相关技术

DNS 域名解析 ARP 与 RARP 协议 DHCP 动态获取 IP 地址 NAT 网络地址转换 ICMP 互联网控制报文协议 IGMP 因特网组管理协议

DNS

我们在上网的时候，通常使用的方式是域名，而不是 IP 地址，因为域名方便人类记忆。那么实现这一技术的就是 DNS 域名解析,DNS 可以将域名网址自动转换为具体的 IP 地址。

DNS 中的域名都是用句点来分隔的，比如 www.server.com ，这里的句点代表了不同层次之间的界限。

在域名中，越靠右的位置表示其层级越高。

所以域名的层级关系类似一个树状结构：

根 DNS 服务器

顶级域 DNS 服务器（com）

权威 DNS 服务器（server.com）

域名解析的工作流程 浏览器首先看一下自己的缓存里有没有，如果没有就向操作系统的缓存要，还没有就检查本机域名解析文件 hosts ，如果还是没有，就会 DNS 服务器进行查询，查询的过程如下：

1. 客户端首先会发出一个 DNS 请求，问 www.server.com 的 IP 是啥，并发给本地 DNS 服务器（也就是客户端的 TCP/IP 设置中填写的 DNS 服务器地址）。
2. 本地域名服务器收到客户端的请求后，如果缓存里的表格能找到 www.server.com，则它直接返回 IP 地址。如果没有，本地 DNS 会去问它的根域名服务器：“老大，能告诉我 www.server.com 的 IP 地址吗?”根域名服务器是最高层次的，它不直接用于域名解析，但能指明一条道路。
3. 根 DNS 收到来自本地 DNS 的请求后，发现后置是.com，说：“www.server.com 这个域名归.com 区域管理”，我给你.com 顶级域名服务器地址给你，你去问问它吧。”
4. 本地 DNS 收到顶级域名服务器的地址后，发起请求问”老二，你能告诉我 www.server.com 的 IP 地址吗?”
5. 顶级域名服务器说：“我给你负责 www.server.com 区域的权威 DNS 服务器的地址，你去问它应该能问到”。
6. 本地 DNS 于是转向问权威 DNS 服务器：“老三,www.server.com 对应的 IP 是啥呀?” server.com 的权威 DNS 服务器，它是域名解析结果的原出处。为啥叫权威呢？就是我的域名我做主。
7. 权威 DNS 服务器查询后将对应的 IP 地址 X.X.X.X 告诉本地 DNS。
8. 本地 DNS 再将 IP 地址返回客户端，客户端和目标建立连接。

ARP

在传输一个 IP 数据报的时候，确定了源 IP 地址和目标 IP 地址后，就会通过主机「路由表」确定 IP 数据包下一跳。然而，网络层的下一层是数据链路层，所以我们还要知道「下一跳」的 MAC 地址。

由于主机的路由表中可以找到下一跳的 IP 地址，所以可以通过 ARP 协议，求得下一跳的 MAC 地址。

ARP 是借助 ARP 请求与 ARP 响应两种类型的包确定 MAC 地址的。

ARP 请求与响应 主机会通过广播发送 ARP 请求，这个包中包含了想要知道的 MAC 地址的主机 IP 地址。

当同个链路中的所有设备收到 ARP 请求时，会去拆开 ARP 请求包里的内容，如果 ARP 请求包中的目标 IP 地址与自己的 IP 地址一致，那么这个设备就将自己的 MAC 地址塞入 ARP 响应包返回给主机。

操作系统通常会把第一次通过 ARP 获取的 MAC 地址缓存起来，以便下次直接从缓存中找到对应 IP 地址的 MAC 地址。

RARP 协议? ARP 协议是已知 IP 地址求 MAC 地址，那 RARP 协议正好相反，它是已知 MAC 地址求 IP 地址。例如将打印机服务器等小型嵌入式设备接入到网络时就经常会用得到。

通常这需要架设一台 RARP 服务器，在这个服务器上注册设备的 MAC 地址及其 IP 地址。然后再将这个设备接入到网络，接着：

该设备会发送一条「我的 MAC 地址是 XXXX，请告诉我，我的 IP 地址应该是什么」的请求信息。

RARP 服务器接到这个消息后返回「MAC 地址为 XXXX 的设备,IP 地址为 XXXX」的信息给这个设备。

最后，设备就根据从 RARP 服务器所收到的应答信息设置自己的 IP 地址。

DHCP

客户端首先发起 DHCP 发现报文（DHCP DISCOVER）的 IP 数据报，由于客户端没有 IP 地址，也不知道 DHCP 服务器的地址，所以使用的是 UDP 广播通信，其使用的广播目的地址是 255.255.255.255(端口 67) 并且使用 0.0.0.0(端口 68) 作为源 IP 地址。DHCP 客户端将该 IP 数据报传递给链路层，链路层然后将帧广播到所有的网络中设备。

DHCP 服务器收到 DHCP 发现报文时，用 DHCP 提供报文（DHCP OFFER）向客户端做出响应。该报文仍然使用 IP 广播地址 255.255.255.255，该报文信息携带服务器提供可租约的 IP 地址、子网掩码、默认网关、DNS 服务器以及 IP 地址租用期。

客户端收到一个或多个服务器的 DHCP 提供报文后，从中选择一个服务器，并向选中的服务器发送 DHCP 请求报文（DHCP REQUEST 进行响应，回显配置的参数。

最后，服务端用 DHCP ACK 报文对 DHCP 请求报文进行响应，应答所要求的参数。

一旦客户端收到 DHCP ACK 后，交互便完成了，并且客户端能够在租用期内使用 DHCP 服务器分配的 IP 地址。

如果租约的 DHCP IP 地址快期后，客户端会向服务器发送 DHCP 请求报文：

服务器如果同意继续租用，则用 DHCP ACK 报文进行应答，客户端就会延长租期。

服务器如果不同意继续租用，则用 DHCP NACK 报文，客户端就要停止使用租约的 IP 地址。

可以发现,DHCP 交互中，全程都是使用 UDP 广播通信。

有了 DHCP 中继代理以后，对不同网段的 IP 地址分配也可以由一个 DHCP 服务器统一进行管理。

NAT

不赘述

ICMP

互联网控制报文协议：

ICMP 主要的功能包括：确认 IP 包是否成功送达目标地址、报告发送过程中 IP 包被废弃的原因和改善网络设置等。

在 IP 通信中如果某个 IP 包因为某种原因未能达到目标地址，那么这个具体的原因将由 ICMP 负责通知。

ICMP 大致可以分为两大类：

一类是用于诊断的查询消息，也就是「查询报文类型」

另一类是通知出错原因的错误消息，也就是「差错报文类型」

IGMP

在前面我们知道了组播地址，也就是 D 类地址，既然是组播，那就说明是只有一组的主机能收到数据包，不在一组的主机不能收到数组包，怎么管理是否是在一组呢？那么，就需要 IGMP 协议了。

IGMP 是因特网组管理协议，工作在主机（组播成员）和最后一跳路由之间，如上图中的蓝色部分。

IGMP 报文向路由器申请加入和退出组播组，默认情况下路由器是不会转发组播包到连接中的主机，除非主机通过 IGMP 加入到组播组，主机申请加入到组播组时，路由器就会记录 IGMP 路由器表，路由器后续就会转发组播包到对应的主机了。

IGMP 报文采用 IP 封装,IP 头部的协议号为 2，而且 TTL 字段值通常为 1，因为 IGMP 是工作在主机与连接的路由器之间。