计算机网络

IP/TCP

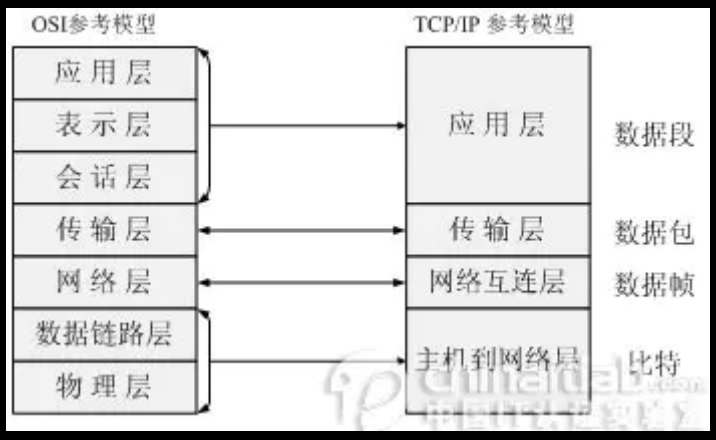
TCP/IP 协议栈

TCP/IP 协议，或称为 TCP/IP 协议栈，或互联网协议系列。TCP/IP 协议栈
TCP/IP 分为 4 层，不同于 OSI，他将 OSI 中的会话层、表示层规划到应用层。
1. 应用层 FTP SMTP HTTP ...
2. 传输层 TCP UDP
3. IP 网络层 IP ICMP IGMP
4. 网络接口层 ARP RARP 以太网令牌环 FDDI ...

协议实际上就是一种约定。
而 TCP/IP 协议并不是指某一个具体的协议，它是指代一系列的协议栈，因此也叫 TCP/IP 协议栈或者 TCP/IP 协议簇。

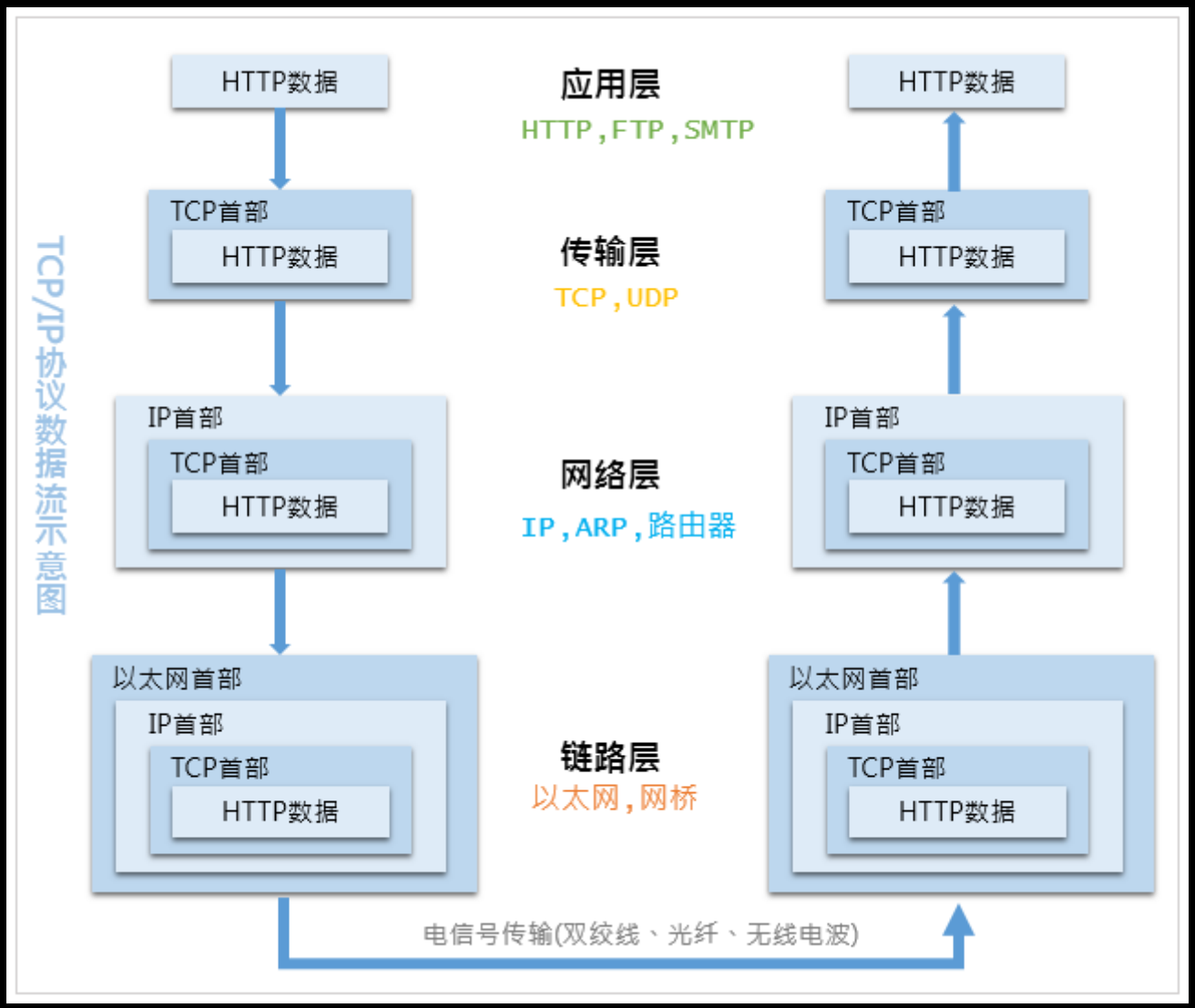
所以广义上，我们说的 TCP/IP 指的是 4 层的总和。
而狭义上来说，指的是 4 层中的传输层和网络互联层。

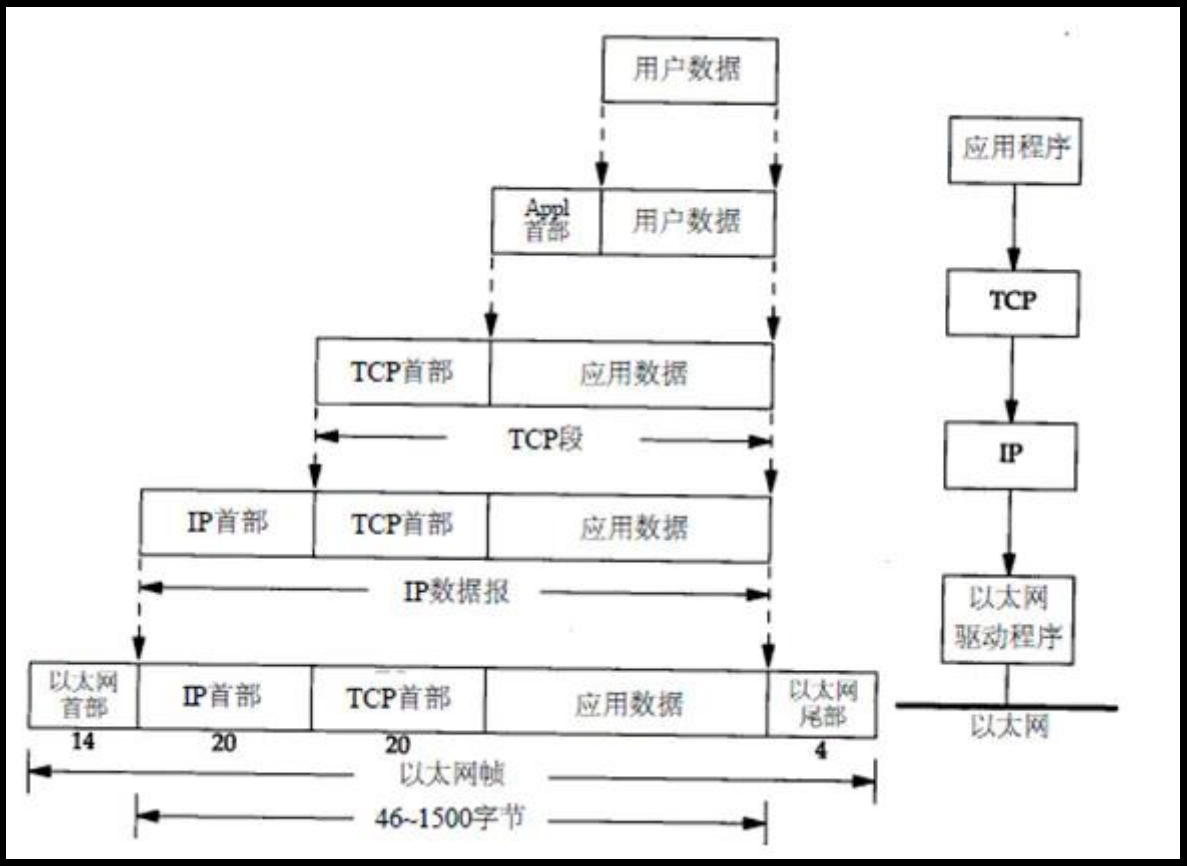
在 TCP/IP 协议簇 中，定义了包含对应 OSI 模型的每一层。但同时对 OSI 模型层做了简化处理。
看看这种图理解一下：



在 TCP/IP 协议簇中每一层都有对应的协议，最终组成协议簇。

应用层	FTP、TELNET、HTTP			SNMP、TFTP、NTP
传输层	TCP			UDP
网络互连层	IP			
主机到网络层	以太网	令牌环网	802.2	HDLC、PPP、FRAME-RELAY
			802.3	EIA/TIA-232，449、V.35、V.21

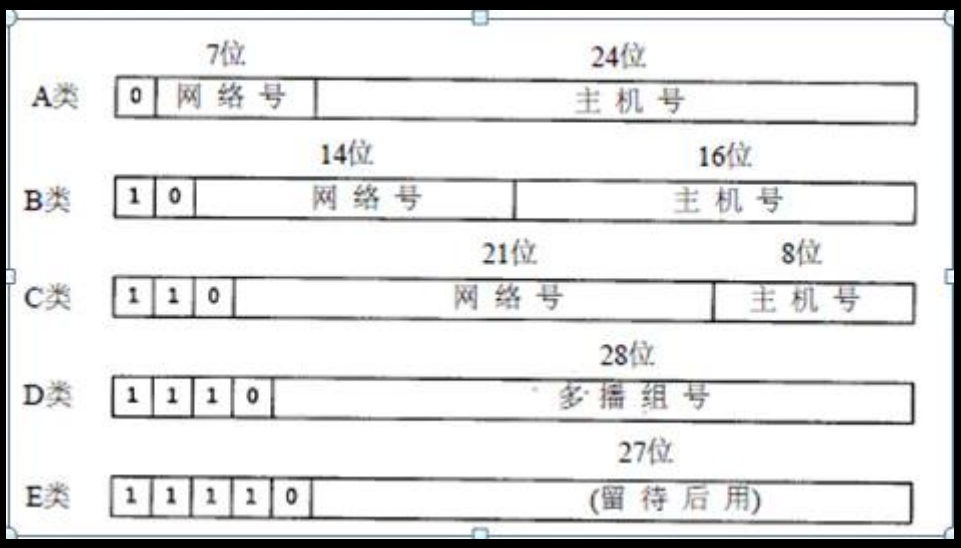




IP 协议

IP 协议处于 TCP/IP 协议簇的网络互联层。它提供不可靠、无连接的服务，也即依赖其他层的协议进行差错控制。在局域网环境，IP 协议往往被封装在以太网帧中传送。而所有的 TCP、UDP、ICMP、IGMP 数据都被封装在 IP 数据报中传送。

IPV4 的 32bit 地址中,分为两个部分：网络号和主机号。同时根据不同的内容开头，又分为 A、B、C、D、E 类。



Where some see coincidence, I see consequence. Where others see chance, I see cost!

网络号用于区分不同的网络点，比如一个公司是一个网络集群，我们可以通过他的网络号确定该公司网关，再通过主机号确定每一台计算。
通过掩码能够改变网络号和主机号的位数。
通常，我们看到的掩码类似：255.255.255.0
二进制表示：11111111.11111111.11111111.00000000

如果一个 IPV4 地址为：192.168.1.12

那么 **IP 地址和掩码经过与运算之后**的结果为：192.168.1.0（192.168.001.000），这就是我们常说的**网关**！

而从 192.168.1.1~192.168.1.254 **都可作为主机号。也即是这个网关下，可以容纳 254 台机器。**

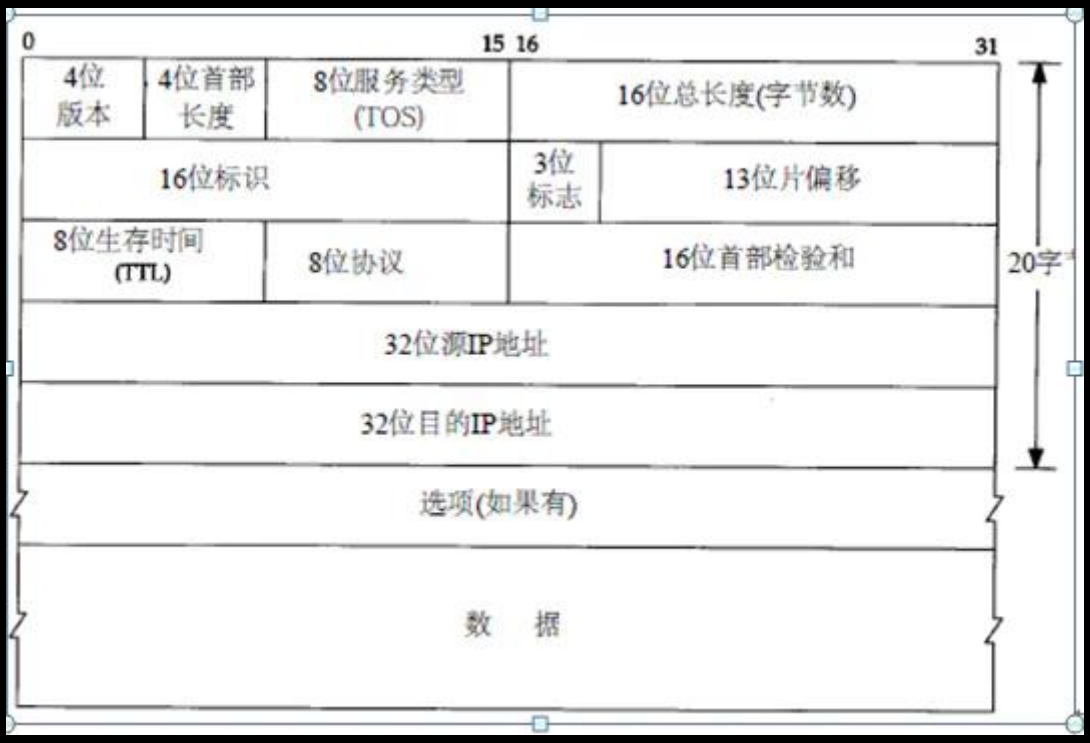
IP 寻址

当一个 IP 包从一台计算机被发送，它会到达一个 IP 路由器。
IP 路由器负责将这个包路由至它的目的地，直接地或者通过其他的路由器。
在一个相同的通信中，一个包所经由的路径可能会和其他的包不同。而路由器负责根据通信量、网络中的错误或者其他参数来进行正确地寻址。

IPV4 中普通的 IP 首部长 20 个字节。其中有 32 位的源 IP 地址和 32 位的目的 IP 地址。

TTL：生存时间。代表了数据包可以经过的**最多路由器数**。比如 TTL 为 10，意思是如果经过 10 次路由器转发，仍然未找到目的地址，则报文丢弃

16 位总长度：指 IP 数据包的最大长度。16bit 那么最长可达 65535 字节。但是通过链路的 MTU 不会有这么大。因此如果数据包长度超过了 MTU，数据包会被分片。如果发生了分片，则需要用到 16 位标识以及 13 位片偏移来找到分片的报文。



TCP 协议

TCP 协议作用

TCP 协议位于协议栈的传输层。当应用层向 TCP 层发送用于网间传输的、用 8 位字节表示的数据流，TCP 则把数据流分割成适当长度的报文段，最大传输段大小（MSS）通常受该计算机连接的网络的数据链路层的最大传送单元（MTU）限制。之后 TCP 把数据包传给 IP 层，由它来通过网络将包传送给接收端实体的 TCP 层。

TCP 为了保证报文传输的可靠，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字节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（假设丢失了）将会被重传。

在拥塞控制上，采用广受好评的 **TCP 拥塞控制**算法（也称 **AIMD** 算法）。
该算法主要包括三个主要部分：

（1）加性增、乘性减；
（2）慢启动；
（3）对超时事件做出反应。

IP & TCP & UDP 报文

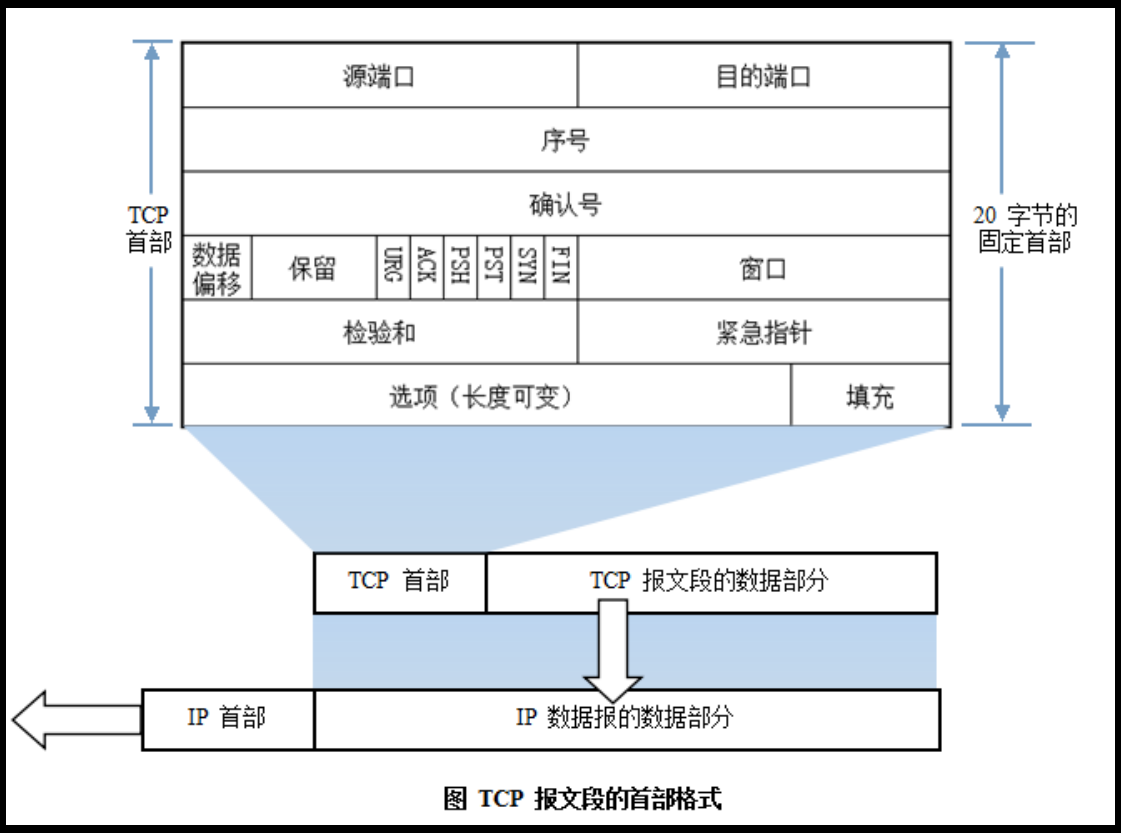
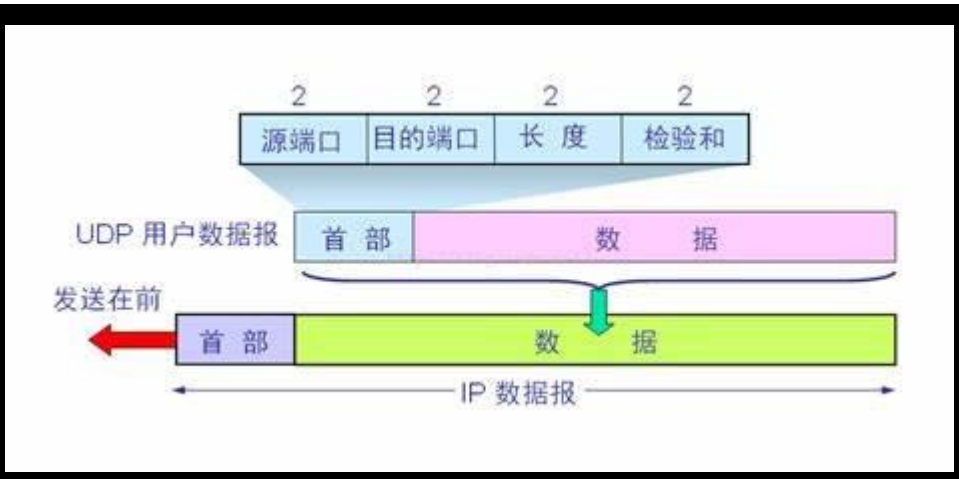
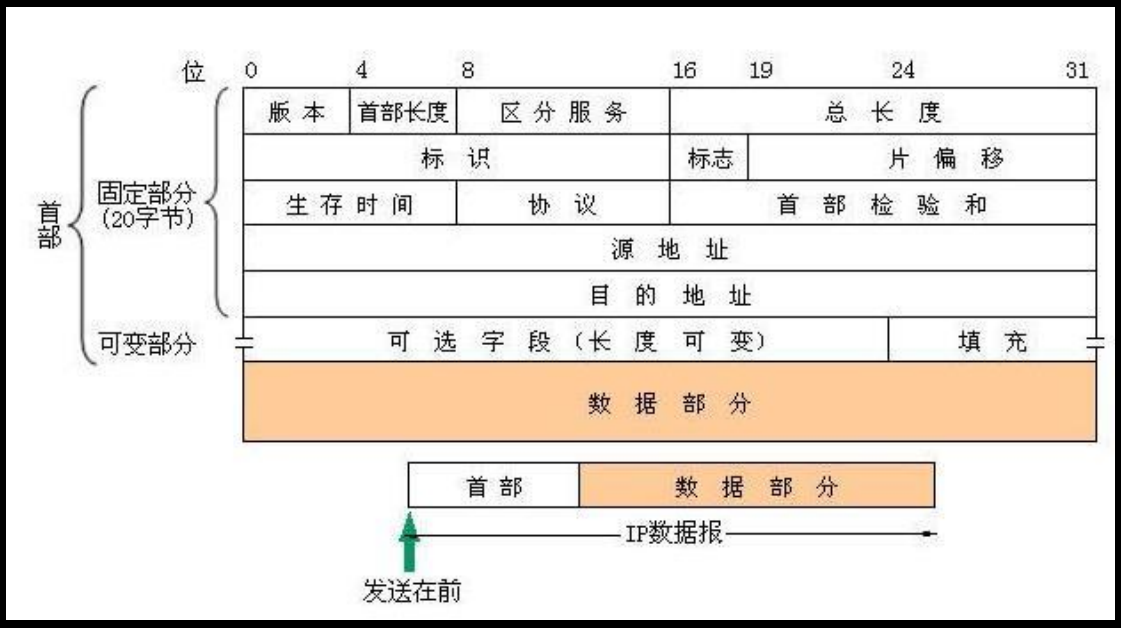


图 TCP 报文段的首部格式





Where some see coincidence, I see consequence. Where others see chance, I see cost!

IP 报文
= 首部+数据部分
= 首部(20 字节固定部分+可变部分) + 数据部分(TCP/UDP 报文)

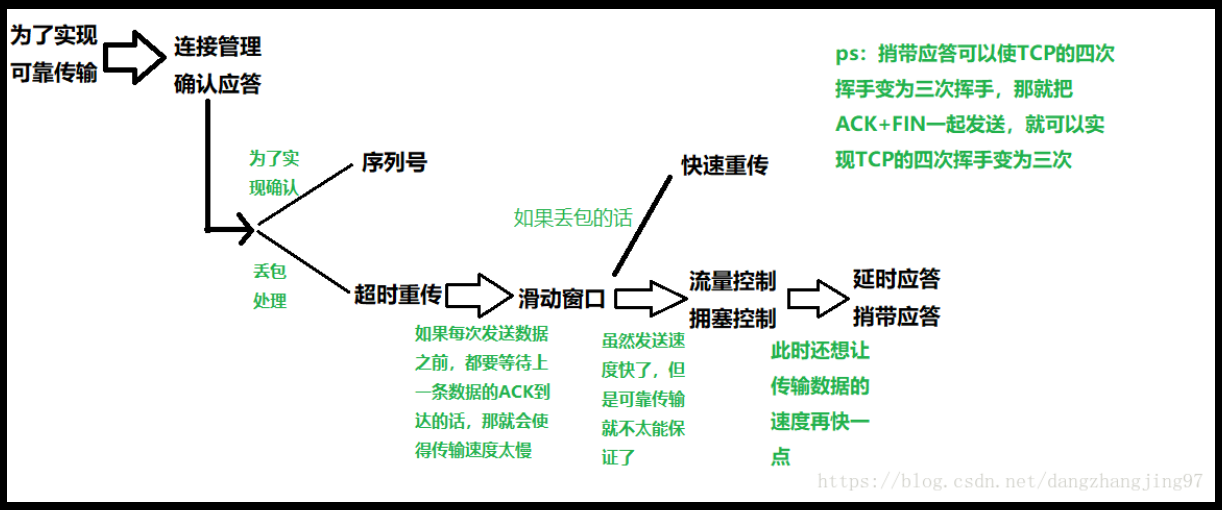
TCP 报文 = 20 字节首部 + 数据
UDP 报文 = 8 字节首部 + 数据

注意:

1. 三者首部不一样: ip: 20 固定+可变; TCP: 20,; UDP: 8
2. 封装顺序: IP 报文的数据部分包含 TCP/UDP 报文。
3. IP 首部最长是多少? IP 数据报首部中有一个首部长度字段, 4 位长 du, 可表示的最大十进制数是 15。因此首部长度的最大值是 15 个 4 字节长的字, 即 60 字节。(一行一行的, 以 4 字节为单位)
4. IP 地址在 IP 报文的首部, 端口在 IP 报文的数据部分, 也是 UDP/TCP 的首部。 为什么不放在一起? 因为解析的时候是先解析 IP 报文, 得到 IP 地址, 然后再解析 TCP 报文, 发送到具体端口的应用。

TCP 拥塞控制

Q: TCP 是怎么保证可靠传输的?



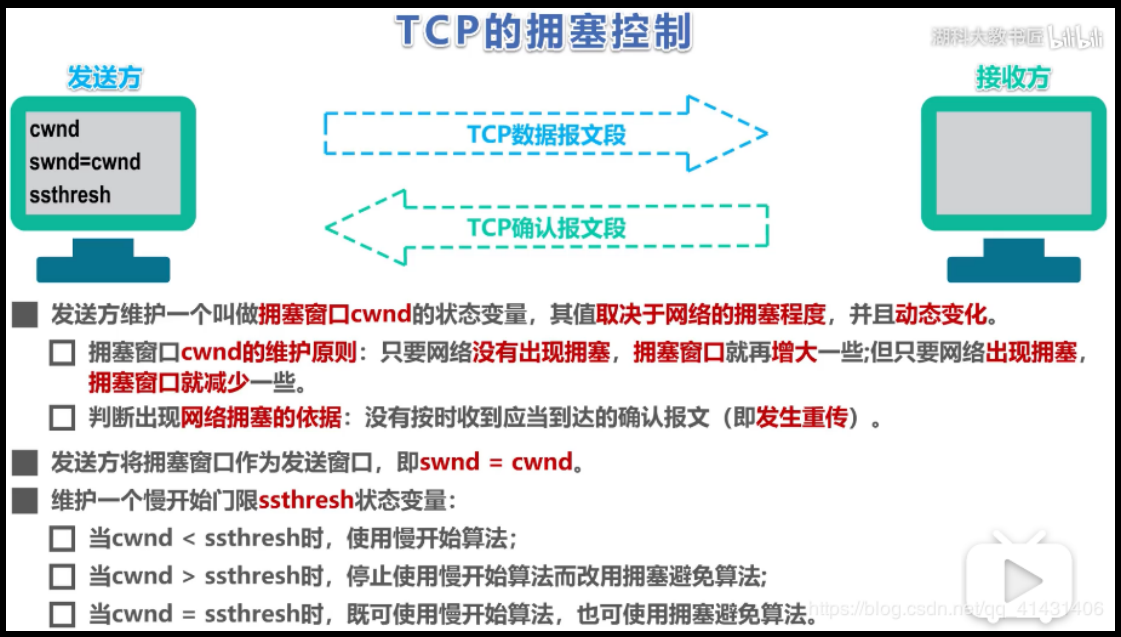
TCP 的四种拥塞控制算法

1.慢开始
2.拥塞控制
3.快重传
4.快恢复

假定:

1.数据是单方向传送, 而另一个方向只传送确认
2.接收方总是有足够大的缓存空间, 因而发送发发送窗口的大小由网络的拥塞程度来决定

3.以 TCP 报文段的个数为讨论问题的单位，而不是以字节为单位



面向报文（UDP）和面向字节流（TCP）的区别

面向报文的传输方式是应用层交给 UDP 多长的报文，UDP 就照样发送，即一次发送一个报文。因此，应用程序必须选择合适大小的报文。若报文太长，则 IP 层需要分片，降低效率。若太短，会是 IP 太小。UDP 对应用层交下来的报文，既不合并，也不拆分，而是保留这些报文的边界。这也就是说，应用层交给 UDP 多长的报文，UDP 就照样发送，即一次发送一个报文。

面向字节流的话，虽然应用程序和 TCP 的交互是一次一个数据块（大小不等），但 **TCP 把应用程序看成是一连串的无结构的字节流**。TCP 有一个缓冲，当应用程序传送的数据块太长，TCP 就可以把它划分短一些再传送。如果应用程序一次只发送一个字节，TCP 也可以等待积累有足够多的字节后再构成报文段发送出去。

应用层协议	应用	传输层协议
SMTP	电子邮件	TCP
TELNET	远程终端接入	
HTTP	万维网	
FTP	文件传输	
DNS	名字转换	UDP
TFTP	文件传输	
RIP	路由选择协议	
BOOTP，DHCP	IP 地址配置	
SNMP	网络管理	
NFS	远程文件服务器	
专用协议	IP 电话	
专用协议	流式多媒体通信	

	TCP	UDP
可靠性	可靠	不可靠
连接性	面向连接	无连接
报文	面向字节流	面向报文（保留报文的边界）
效率	传输效率低	传输效率高
双工性	全双工	一对一、一对多、多对一、多对多
流量控制	有（滑动窗口）	无
拥塞控制	有（慢开始、拥塞避免、快重传、快恢复）	无

ARP

“Address Resolution Protocol”（地址解析协议）

作用是在以太网环境中，数据的传输所依懒的是 MAC 地址而非 IP 地址，而将已知 IP 地址转换为 MAC 地址的工作是由 ARP 协议来完成的。

局域网中，网络中实际传输的是“帧”，帧里面是有目标主机的 MAC 地址的。在以太网中，一个主机和另一个主机进行直接通信，必须要知道目标主机的 MAC 地址。但这个目标 MAC 地址是如何获得的呢？它就是通过地址解析协议获得的。所谓**“地址解析”就是主机在发送帧前将目标 IP 地址转换成目标 MAC 地址的过程**。ARP 协议的基本功能就是通过目标设备的 IP 地址，查询目标设备的 MAC 地

址，以保证通信的顺利进行。

ARP 映射方式:静态、动态

静态映射的意思是要手动创建一张 ARP 表，把逻辑（IP）地址和物理地址关联起来。这个 ARP 表储存在网络中的每一台机器上。例如，知道其机器的 IP 地址但不知道其物理地址的机器就可以通过查 ARP 表找出对应的物理地址。这样做有一定的局限性，因为物理地址可能发生变化:

（1）机器可能更换 NIC（网络适配器），结果变成一个新的物理地址。

（2）在某些局域网中，每当计算机加电时，他的物理地址都要改变一次。

（3）移动电脑可以从一个物理网络转移到另一个物理网络，这样会时物理地址改变。

动态映射:

动态映射时，每次只要机器知道另一台机器的逻辑（IP）地址，就可以使用协议找出相对应的物理地址。已经设计出的实现了动态映射协议的有 ARP 和 RARP 两种。ARP 把逻辑（IP）地址映射为物理地址。RARP 把物理地址映射为逻辑（IP）地址。

ARP 原理及流程:

在任何时候，一台主机有 IP 数据报文发送给另一台主机，它都要知道接收方的逻辑（IP）地址。但是 IP 地址必须封装成帧才能通过物理网络。这就意味着发送方必须有接收方的物理（MAC）地址，因此需要完成逻辑地址到物理地址的映射。而 ARP 协议可以接收来自 IP 协议的逻辑地址，将其映射为相应的物理地址，然后把物理地址递交给数据链路层。

ARP 请求:

任何时候，当主机需要找出这个网络中的另一个主机的物理地址时，它就可以发送一个 ARP 请求报文，这个报文包好了发送方的 MAC 地址和 IP 地址以及接收方的 IP 地址。因为发送方不知道接收方的物理地址，所以这个查询分组会在网络层中进行广播。

ARP 响应:

局域网中的每一台主机都会接受并处理这个 ARP 请求报文，然后进行验证，查看接收方的 IP 地址是不是自己的地址，只有验证成功的主机才会返回一个 ARP 响应报文，这个响应报文包含接收方的 IP 地址和物理地址。这个报文**利用收到的 ARP 请求报文中的请求方物理地址以单播**的方式直接发送给 ARP 请求报文的请求方。(不是同一网络：利用网关)

报文格式:

硬件类型		协议类型
硬件长度	协议长度	操作码（请求为1，响应为2）
源硬件地址		
源逻辑地址		
目的硬件地址		
目的逻辑地址		

（图3）ARP报文格式

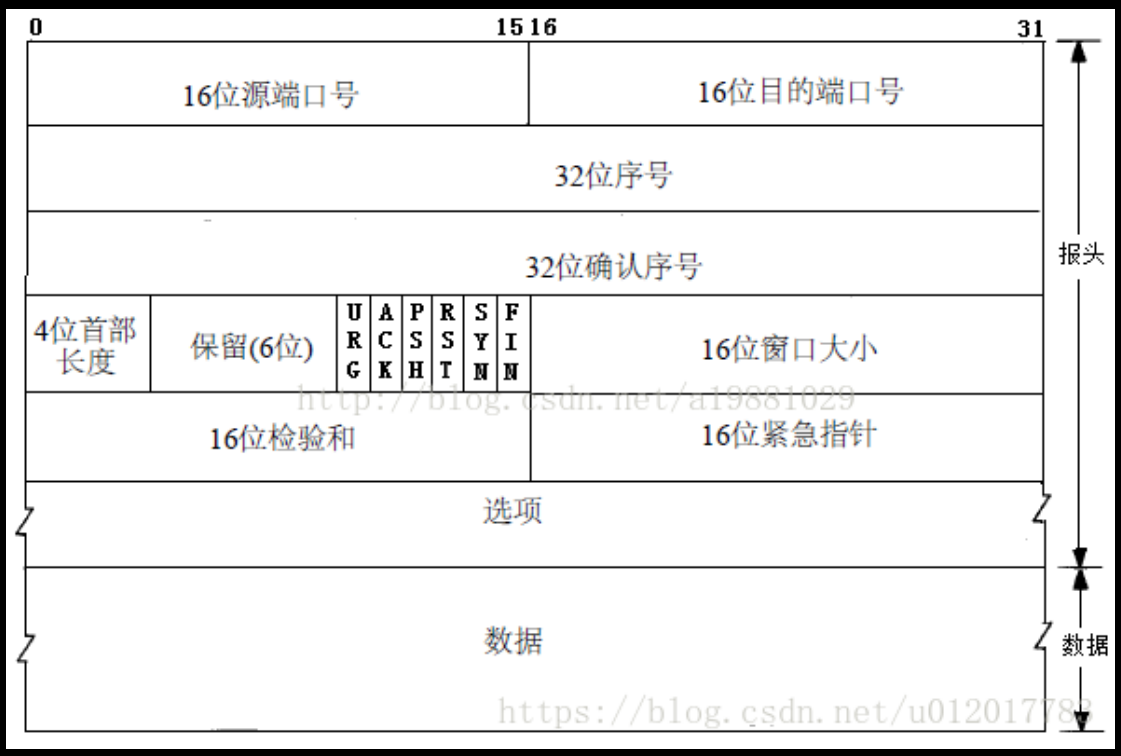
ARP 报文的总长度为 64 字节。
ARP 报文直接封装在数据链路帧中

TCP vs. UDP

UDP 不提供复杂的控制机制，利用 IP 提供面向无连接的通信服务。它是将应用程序发来的数据在收到的那一刻，立刻按照原样发送到网络上的一种机制。
TCP 作为一种面向有连接的协议，只有在确认通信对端存在时才会发送数据，从而可以控制通信流量的浪费。**TCP 通过检验和、序列号、确认应答、重发控制、连接管理以及窗口控制等机制实现可靠性传输**。

TCP 与 UDP 区别总结：
1、TCP 面向连接;UDP 是无连接的，即发送数据之前不需要建立连接
2、TCP 提供可靠的服务。也就是说，通过 TCP 连接传送的数据，无差错，不丢失，不重复，且按序到达；UDP 尽最大努力交付，即不保 证可靠交付
3、TCP 面向字节流，实际上是 TCP 把数据看成一连串无结构的字节流;UDP 是面向报文的
　UDP 没有拥塞控制，因此网络出现拥塞不会使源主机的发送速率降低（对实时应用很有用，如 IP 电话，实时视频会议等）
4、每一条 TCP 连接只能是点到点的;UDP 支持一对一，一对多，多对一和多对多的交互通信
5、TCP 首部开销 20 字节;UDP 的首部开销小，只有 8 个字节
6、TCP 的逻辑通信信道是全双工的可靠信道，UDP 则是不可靠信道

TCP 报头：



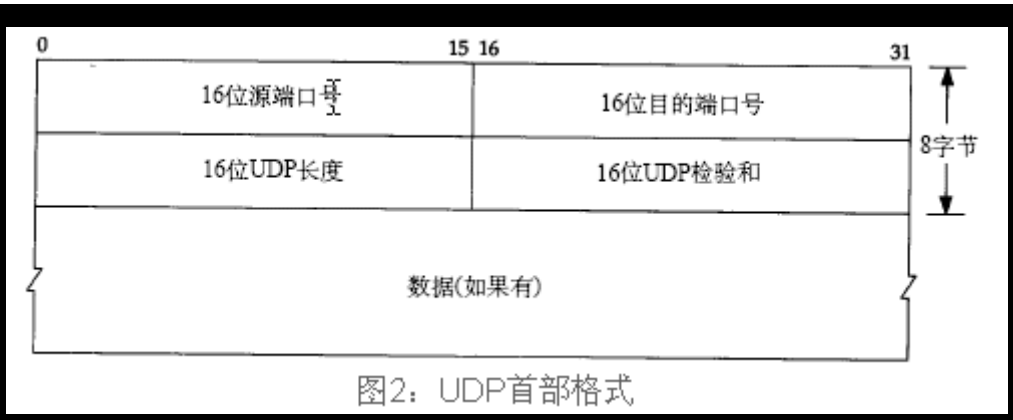
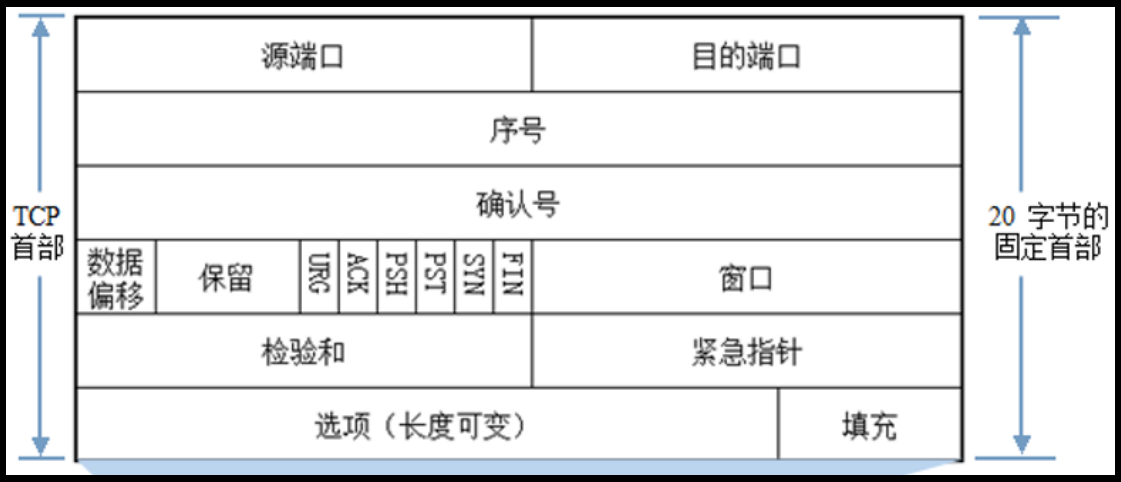


图2：UDP首部格式

TCP 三次握手,四次挥手

TCP 报文：



序号（sequence number）：Seq 序号，占 32 位，用来标识从 TCP 源端向目的端发送的字节流，发

起方发送数据时对此进行标记。
确认号（acknowledgement number）：Ack 序号，占 32 位，只有 ACK 标志位为 1 时，确认序号字段才有效，**Ack=Seq+1**。
标志位（Flags）：共 6 个，即 URG、ACK、PSH、RST、SYN、FIN 等，具体含义如下：
1. URG：紧急指针（urgent pointer）有效。
2. ACK：确认序号有效。
3. PSH：接收方应该尽快将这个报文交给应用层。
4. RST：重置连接。
5. SYN：发起一个新连接。
6. **FIN：释放一个连接。**

TCP 报头中的源端口号和目的端口号同 IP 数据报中的源 IP 与目的 IP 唯一确定一条 TCP 连接。

- 三次握手
  是指建立一个 TCP 连接时，需要客户端和服务器总共发送 3 个包。
  三次握手的目的是连接服务器指定端口，建立 TCP 连接，并同步连接双方的序列号和确认号并交换 TCP 窗口大小信息。

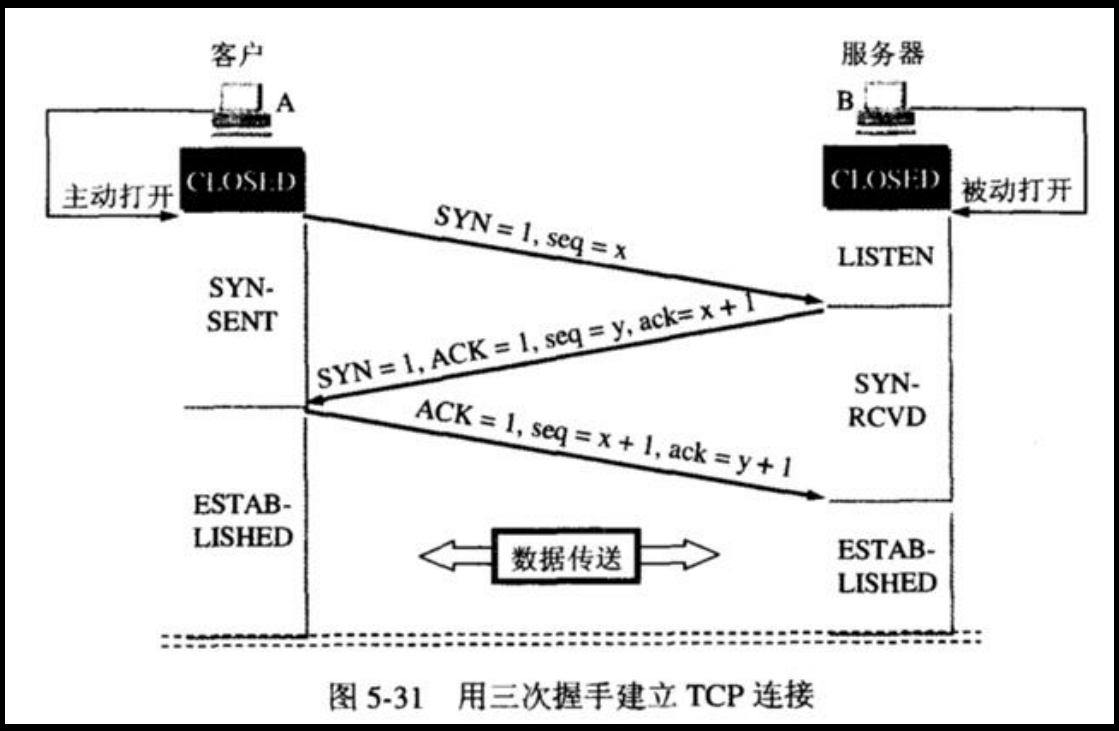


图 5-31　用三次握手建立 TCP 连接

- 四次挥手

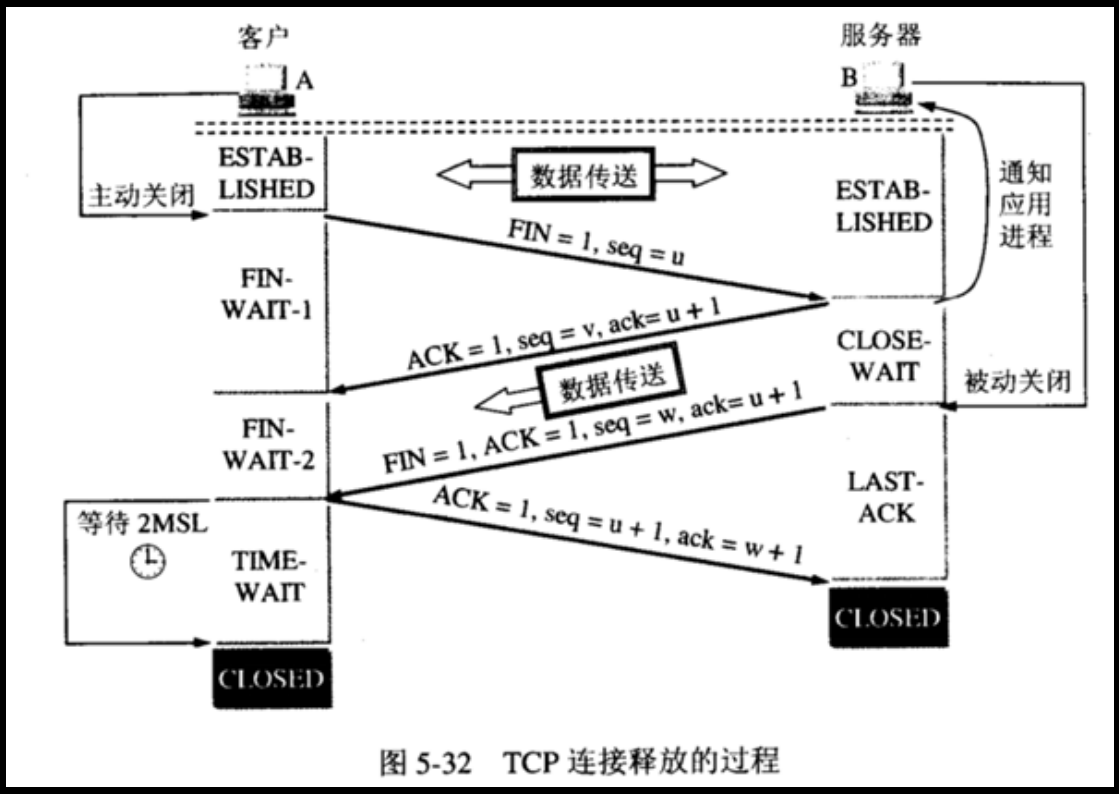
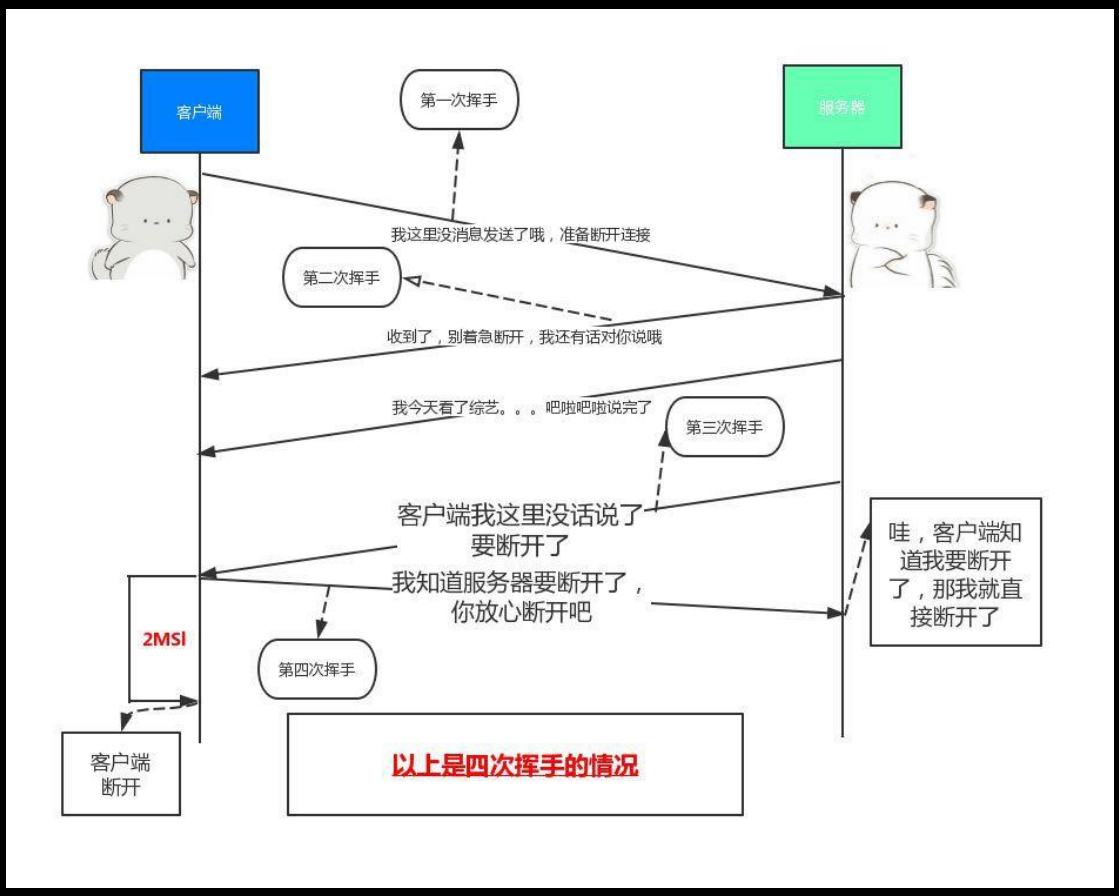


图 5-32　TCP 连接释放的过程



MSL 是 Maximum Segment Lifetime 英文的缩写，中文可以译为“报文最大生存时间”，他是任何报文在网络上存在的最长时间，超过这个时间报文将被丢弃。

为什么 C 需要等待 2MSL？

考虑第四次 C 发送的 TCP 报文丢掉了，这时会发生什么？服务端会一直等待，不能关闭连接！

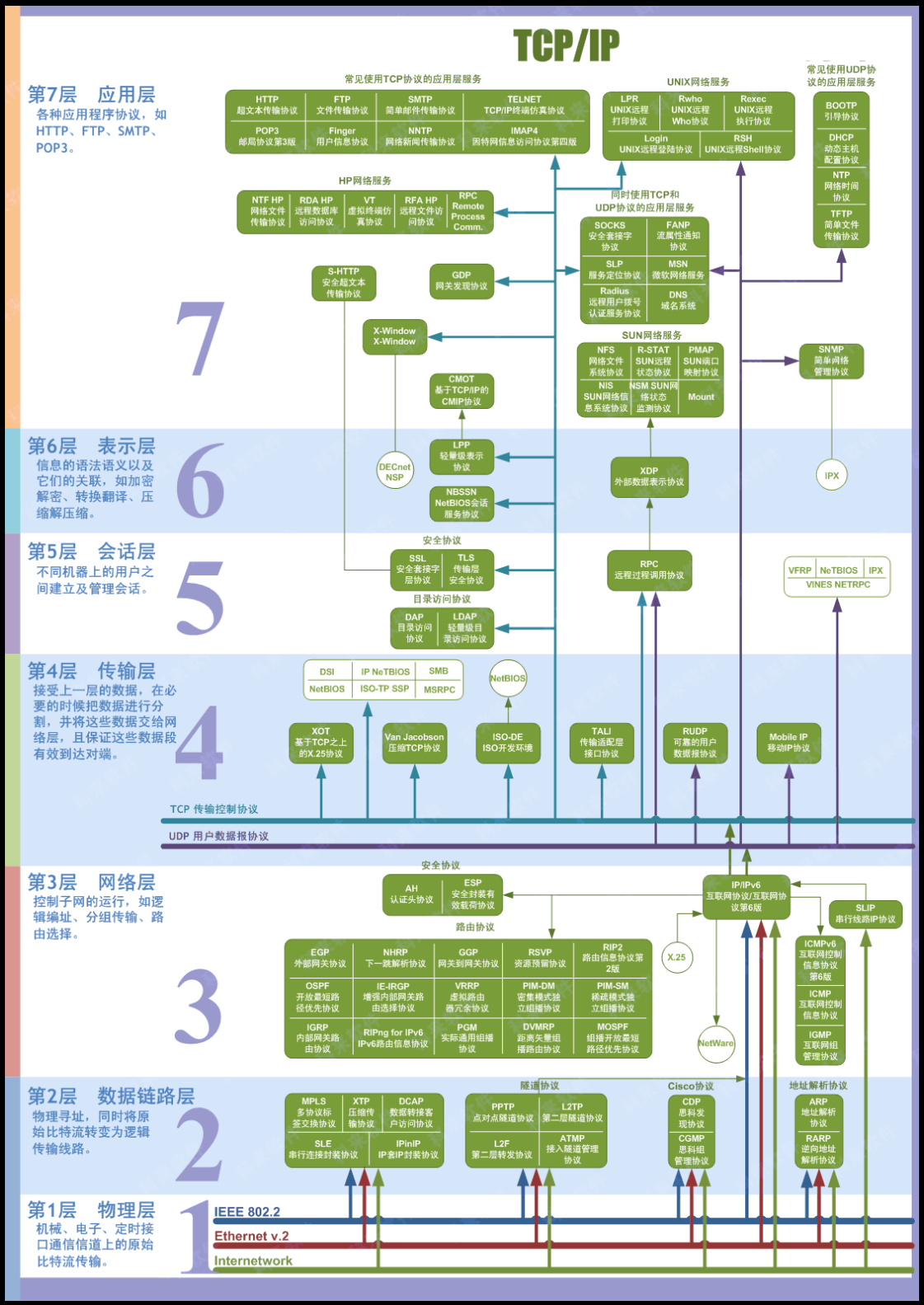
Where some see coincidence, I see consequence. Where others see chance, I see cost!

于是：如果服务端发送第三次挥手报文，过了 2MSL 之后还没有收到客服端的响应，那就表明丢包了，可能是第三次也可能是第四次，那就**超时重传**，在重新发送第三个挥手报文。

为什么是 2MSL？
个人感觉一方面是：一来一回需要 2MSL。
二是：**等待本次连接的所有数据消失**

如果 Client（客户端）直接 CLOSED（关闭），然后又再向 Server（服务器端）发起一个新连接，我们不能保证这个新连接与刚关闭的连接的端口号是不同的。也就是说有可能新连接和老连接的端口号是相同的。一般来说不会发生什么问题，但是还是有特殊情况出现：假设新连接和已经关闭的老连接端口号是一样的，如果前一次连接的某些数据仍然滞留在网络中，这些延迟数据在建立新连接之后才到达 Server，**由于新连接和老连接的端口号是一样的，于是，TCP 协议就认为那个延迟的数据是属于新连接的，这样就和真正的新连接的数据包发生混淆了。所以 TCP 连接还要在 TIME_WAIT 状态等待 2 倍 MSL，这样可以保证本次连接的所有数据都从网络中消失。**

7 层



三次握手过程可以携带数据吗

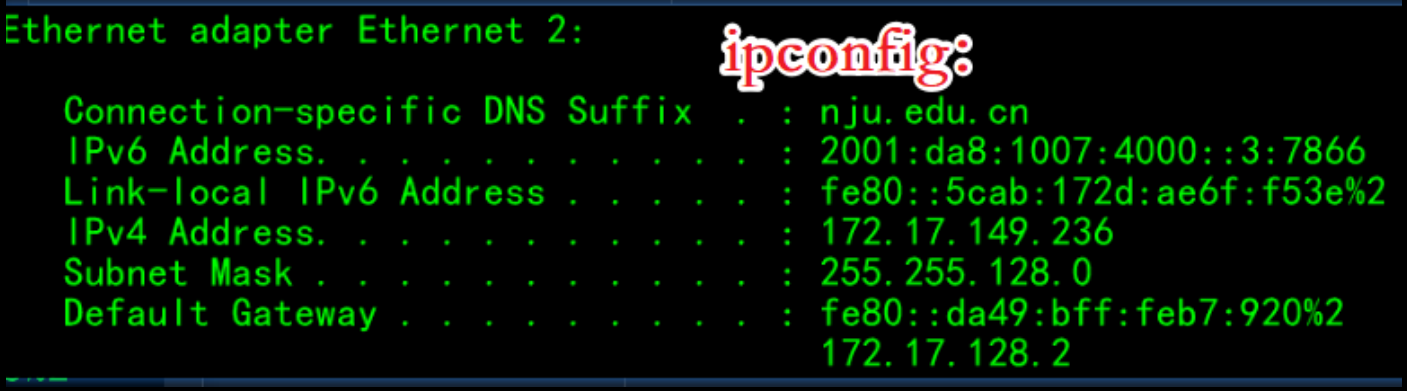
TCP 协议建立连接的三次握手过程中的第三次握手允许携带数据。

RFC793 文档里**带有 SYN 标志的过程包是不可以携带数据**的，也就是说三次握手的前两次是不可以携带数据的

Where some see coincidence, I see consequence. Where others see chance, I see cost!

公网 Ip 和私网 ip

IP 可以分为 Public IP 和 Private IP,出现这种规划的原因在于 IPv4 所能表示的 IP 太少而电脑太多以至于不够用，然而只有 Public IP 才能直接连接上网络，所以对于那些公司，学校，政府机构等场所，就可以**集中使用私有的 IP 进行管理**，而大家可以**共用一个 IP 去连接上公网**，这样，就省下了许多宝贵的 Public IP。你有没有发现，你每次使用 ipconfig 查到的地址，要么就是 172.开头的，要么就是 192.开头的，为什么？难道没有其他开头的嘛，答:基本没有。因为一个 Ip 分给我们一个 Pc 端太不划算了，一般都是很多人共享一个 ip，我们用 ipconfig 查询到的其实只是我们的局域网 Ip,172.开头的意味着我们是 B 类型的局域网,以１９２.开头的意味着你是ｃ类的局域网。所以我们一直都没有接触到正真的ｉｐ，我们一直打交道的都是私有ｉｐ又叫 private ip.



ipconfig 查出来的是你本机的 IP 地址，也就是内网私有地址，此类地址仅在局域网使用，不能联通外网。
百度查出来的地址是你上网的共有地址，也许并不是你主机的地址，而是电信或联通分给你的地址，用于连接互联网。

- Public IP : 公共 IP ，经由 INTERNIC 所统一规划的 IP，有这种 IP 才可以连上 Internet ；
- Private IP : 私有 IP 或保留 IP，不能直接连上 Internet 的 IP ，主要用于局域网络内的主机联机规划。

早在 IPv4 规划的时候就担心 IP 会有不足的情况，而且为了应付某些企业内部的网络设定，于是就有了私有 IP (Private IP) 的产生了。**私有 IP 也分别在 A, B, C 三个 Class 当中各保留一段作为私有 IP 网段**，那就是:

1. Class A: 10.0.0.0 - 10.255.255.255

2. Class B: 172.16.0.0 - 172.31.255.255

3. Class C: 192.168.0.0 - 192.168.255.255

类别	IP范围	私有地址范围	保留地址
A	1.0.0.0~126.255.255.255	10.0.0.0~10.255.255.255	127.0.0.0~127.255.255.255
B	128.0.0.0~191.255.255.255	172.16.0.0~172.31.255.255	169.254.0.0~169.254.255.255
C	192.0.0.0~223.255.255.255	192.168.0.0~192.168.255.255	无

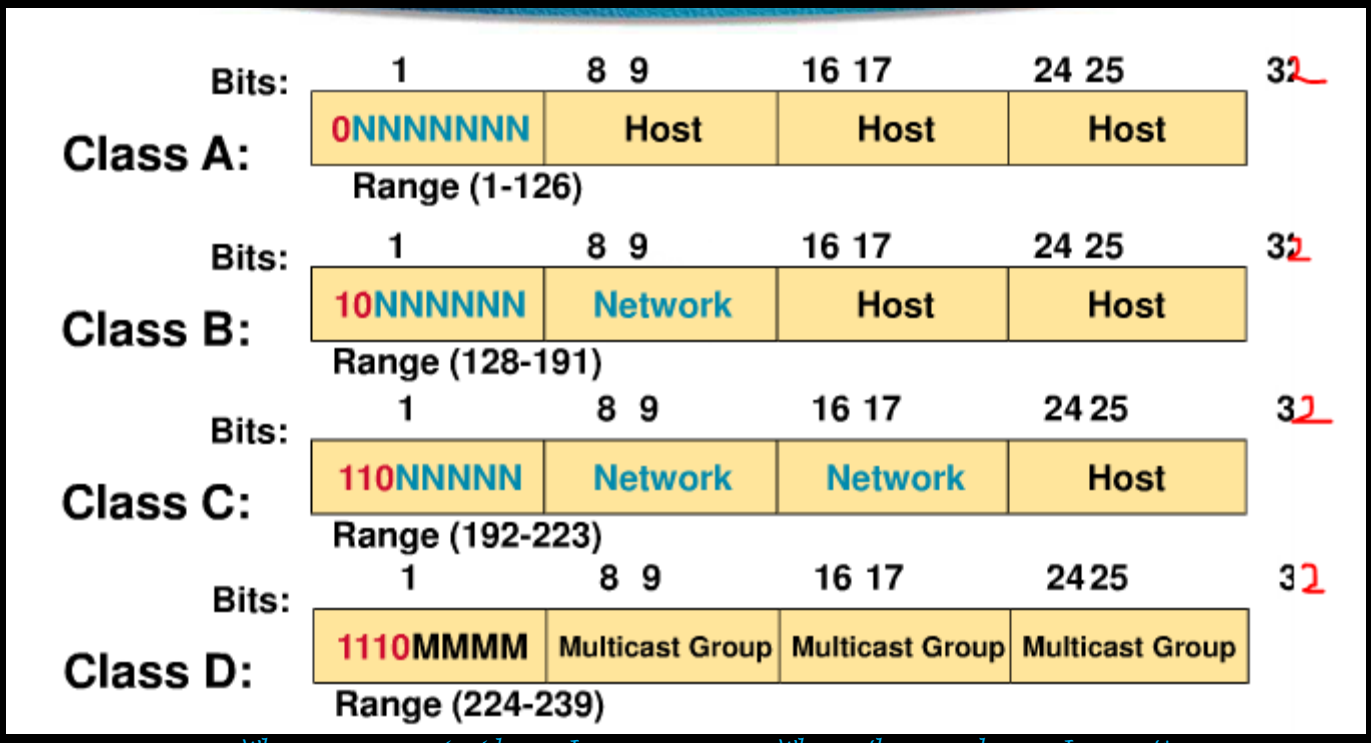
这三个 IP 网段就只做为内部私有网域的 IP 沟通之用

一般说来，我们用 Ipcofig 是无法查到公网 ip 的，大多数时候，你使用 ipconfig 查到的一般就只是以 172.开头的 b 类私有 Ip，或者以 **192.168 开头的 c 类私有 Ip**.简单的说，私有 ip 有底下的几个限制：

1. 私有 IP 的路由信息不能对外散播 (只能存在内部网络)；
2. 使用私有 IP 作为来源或目的地址的封包，不能透过 Internet 来转送 (不然网络会混乱)；
3. 关于私有 IP 的参考纪录(如 DNS)，只能限于内部网络使用

IP 分类

ip 常见有 5 种分类，而我们在实际生活中和生产中一般只会遇到 A,B,C 这三类地址



Where some see coincidence, I see consequence. Where others see chance, I see cost!

类别	最大网络数	IP地址范围	最大主机数	私有IP地址范围
A	126（2^7-2)	0.0.0.0-127.255.255.255	16777214	10.0.0.0-10.255.255.255
B	16384(2^14)	128.0.0.0-191.255.255.255	65534	172.16.0.0-172.31.255.255
C	2097152(2^21)	192.0.0.0-223.255.255.255	254	192.168.0.0-192.168.255.255

类别	默认子网掩码	支持主机数
A	255.0.0.0	2^24-2
B	255.255.0.0	2^16-2
C	255.255.255.0	2^8-2

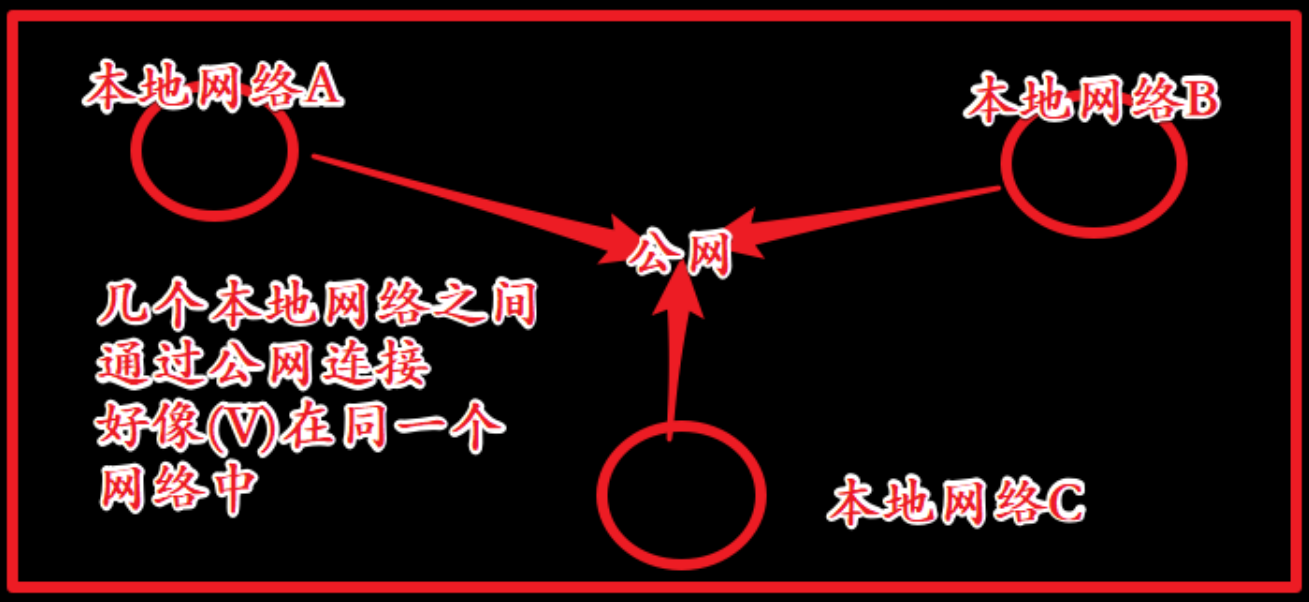
VPN

虚拟专用网络

一个公司、组织使用需要的 ip 地址多余能分配的 ip 地址，于是使用私有 ip 构建局域网(本地网络、专用互联网)。
该局域网内 IP 是 ABC 三类中的私有 ip，他们使用共同的公网 ip 访问公网。

如果一个不在该局域网中的 主机想要访问该局域网，就需要 VPN

VPN 使用公网作为给专用网络通信的载体。
通过公网来连接 分布在各地的 本地网络。



每一个本地网络至少需要一个路由器
本地网络间传输的数据会经过

还有另外一种 VPN：**远程接入 VPN**
有些公司并没有分布在不同的场所
但是有员工在外流动
这时在外员工通过 VPN 软件接入公司本地网络。

数据链路层协议

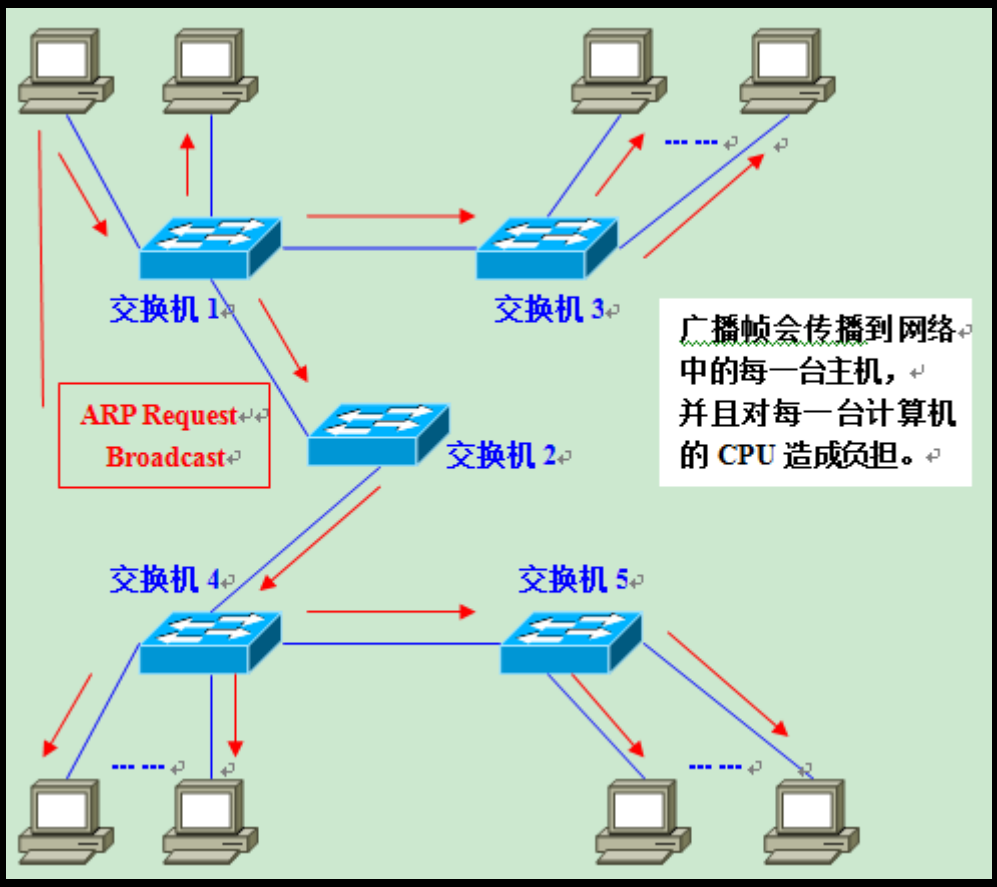
VLAN

Virtual Local Area Network
VLAN（虚拟局域网）是对连接到的第二层交换机端口的网络用户的逻辑分段，不受网络用户的物理位置限制而根据用户需求进行网络分段。一个 VLAN 可以在一个交换机或者跨交换机实现。VLAN 可以根据网络用户的位置、作用、部门或者根据网络用户所使用的应用程序和协议来进行分组。基于交换机的虚拟局域网能够为局域网解决冲突域、广播域、带宽问题。

VLAN 相当于 OSI 参考模型的第二层的广播域，能够将广播风暴控制在一个 VLAN 内部，划分 VLAN 后，由于广播域的缩小，网络中广播包消耗带宽所占的比例大大降低，网络的性能得到显著的提高。不同的 VLAN 之间的数据传输是通过第三层（网络层）的路由来实现的，因此使用 VLAN 技术，结合数据链路层和网络层的交换设备可搭建安全可靠的网络。网络管理员通过控制交换机的每一个端口来控制网络用户对网络资源的访问，同时 VLAN 和第三层第四层的交换结合使用能够为网络提供较好的安全措施。

VLAN 的作用是分割广播域，属于第二层数据链路层。

广播域，指的是广播帧(目标 MAC 地址全部为 1)所能传递到的范围，亦即能够直接通信的范围。严格地说，并不仅仅是广播帧，多播帧(Multicast Frame)和目标不明的单播帧(Unknown Unicast Frame)也能在同一个广播域中畅行无阻。
未分割广播域时：如果仅有一个广播域，有可能会影响到网络整体的传输性能。



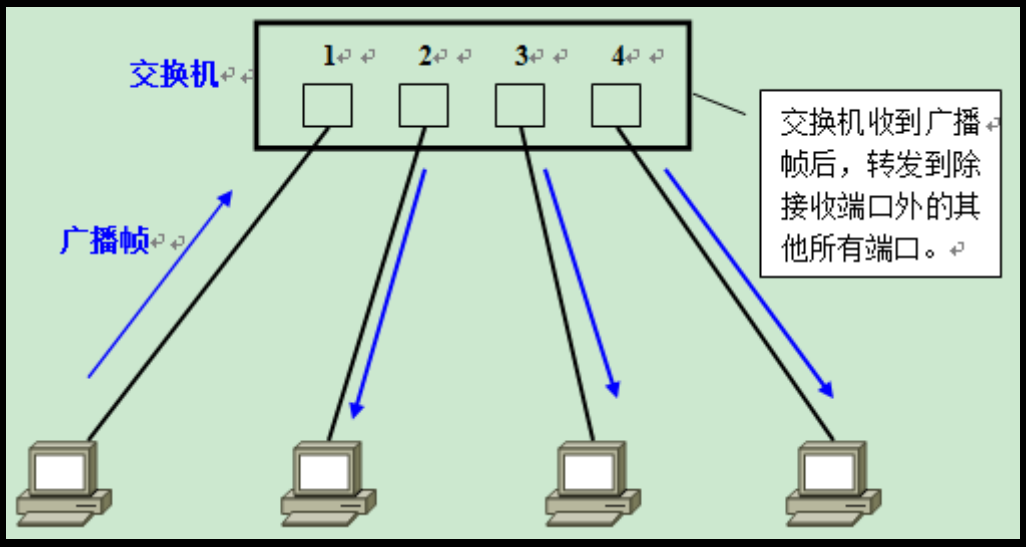
图中，是一个由 5 台二层交换机(交换机 1～5)连接了大量客户机构成的网络。假设这时，计算机 A 需要与计算机 B 通信。在基于以太网的通信中，必须在数据帧中指定目标 MAC 地址才能正常通信，因此计算机 A 必须先广播“ARP 请求(ARP Request)信息”，来尝试获取计算机 B 的 MAC 地址。
交换机 1 收到广播帧(ARP 请求)后，会将它转发给除接收端口外的其他所有端口，也就是 Flooding 了。接着，交换机 2 收到广播帧后也会 Flooding。交换机 3、4、5 也还会 Flooding。最终 ARP 请求会被转发到同一网络中的所有客户机上。
ARP 请求原本是为了获得计算机 B 的 MAC 地址而发出的。也就是说：只要计算机 B 能收到就万事大吉了。可是事实上，数据帧却传遍整个网络，导致所有的计算机都收到了它。如此一来，一方面广播信息消耗了网络整体的带宽，另一方面，收到广播信息的计算机还要消耗一部分 CPU 时间来对它进

行处理。造成了网络带宽和 CPU 运算能力的大量无谓消耗。

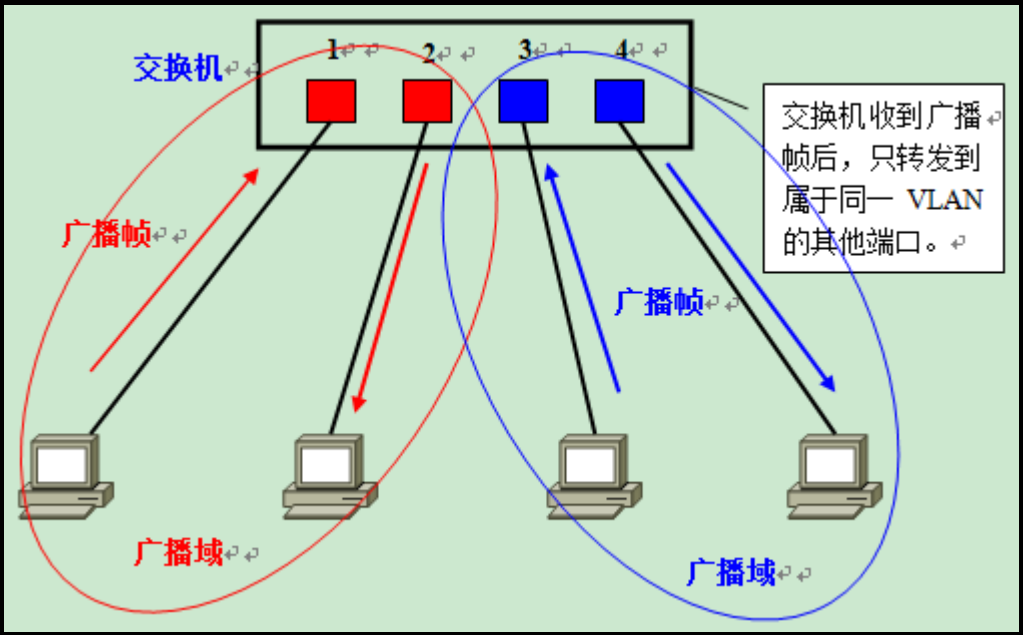
广播信息真是那么频繁出现的吗?
是的!实际上广播帧会非常频繁地出现。利用 TCP/IP 协议栈通信时，除了前面出现的 ARP 外，还有可能需要发出 DHCP、RIP 等很多其他类型的广播信息。

分割广播域时，一般都必须使用到路由器。使用路由器后，可以以路由器上的网络接口(LAN Interface)为单位分割广播域。

交换机是如何使用 VLAN 分割广播域?
首先，在一台未设置任何 VLAN 的二层交换机上，任何广播帧都会被转发给除接收端口外的所有其他端口(Flooding)。例如，计算机 A 发送广播信息后，会被转发给端口 2、3、4。



这时，如果在交换机上生成红、蓝两个 VLAN;同时设置端口 1、2 属于红色 VLAN、端口 3、4 属于蓝色 VLAN。再从 A 发出广播帧的话，交换机就只会把它转发给同属于一个 VLAN 的其他端口——也就是同属于红色 VLAN 的端口 2，不会再转发给属于蓝色 VLAN 的端口。
同样，C 发送广播信息时，只会被转发给其他属于蓝色 VLAN 的端口，不会被转发给属于红色 VLAN 的端口。



在不同的 VLAN 间通信时又该如何是好呢？
VLAN 是广播域。而通常两个广播域之间由路由器连接，广播域之间来往的数据包都是由路由器中继的。因此，VLAN 间的通信也需要路由器提供中继服务，这被称作“VLAN 间路由”。

VLAN 划分与子网划分的区别

其实是分割广播域，把一个大的广播域划分成更多小的广播域，
一个 VLAN 就是一个广播域，既然一个 VLAN 就是一个广播域，
那么计算机之间的通信其实是通过 MAC 地址实现的，
所以一个 VLAN 中的计算机想跟另外一个 VLAN 的计算机通信，
那么他就会广播 ARP 报文，会问“目的主机的 MAC 地址”，
而此时广播的 arp 报文并不能到达另外一个 Vlan，
因为一个 VLAN 就是一个广播域，
广播报文不会广播到另外一个广播域中去，
所以不同 VLAN 之间的通信需要三层设备来实现！
但数据经过交换机的时候会加上 VLAN 的标记，
然后交换机根据 VLAN 标识来转发数据，
当到达目的主机所在的 VLAN 时，
VLAN 标志就会去掉！
VLAN 划分简单的是在一个局域网划分的一个小局域网，
不让内网的计算机来访问这个数据。

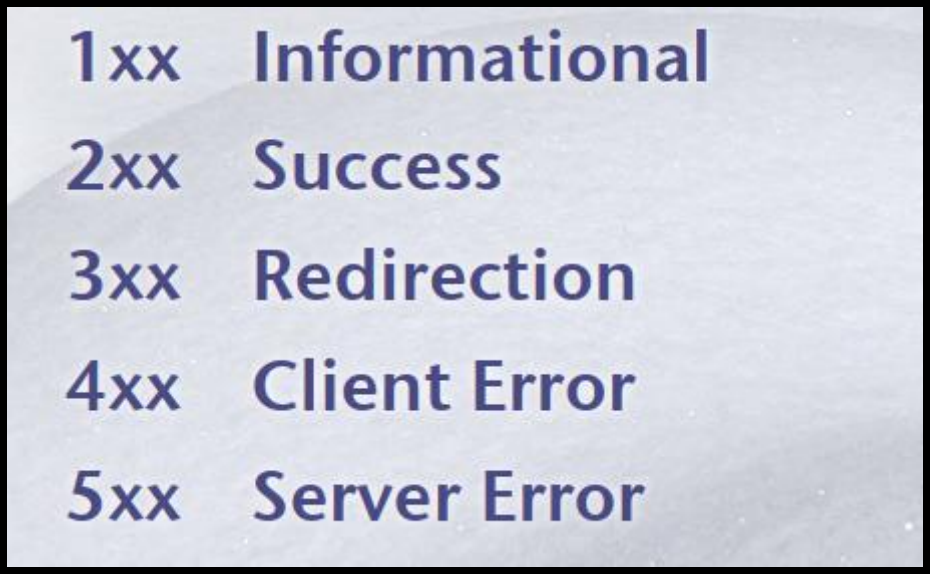
子网划分
是通过借主机位来作为网络位，
实现子网的划分，
也就是**可变长子网掩码**，
划分子网后，

其实就是属于不同的网络了，
他们之间的通信需要三层设备或是路由器的支持才可以通信。

子网划分的好处是：
1、节省 IP 地址。
2、减少广播域。
3、方便管理。

HTTP

状态码



1. 202：请求也接受，还没处理，可能被拒绝
2. **301 是永久重定向，而 302 是临时重定向。** 当然，他们之间也是有共同点的，就是用户都可以看到 url 替换为了一个新的，然后发出请求。
3. **304：如果客户端发送了一个带条件的 GET 请求且该请求已被允许，而文档的内容（自上次访问以来或者根据请求的条件）并没有改变**，则服务器应当返回这个 304 状态码。简单的表达就是：服务端已经执行了 GET，但文件未变化。
4. 400 Bad Request：语义有误，当前请求无法被服务器理解；参数有误
5. 401 Unauthorized
6. 403 Forbidden：服务器已经理解请求，但是拒绝执行它。
7. 404 Not Found：请求失败，请求所希望得到的资源未被在服务器上发现。
8. 406 Not Acceptable：请求的资源的内容特性无法满足请求头中的条件，因而无法生成响应实体。
9. 500 Internal Server Error，服务器遇到了一个未曾预料的状况
10. 502 Bad Gateway：作为网关或者代理工作的服务器尝试执行请求时，从上游服务器接收到无效的响应。
11. 503 Service Unavailable：由于临时的服务器维护或者过载，服务器当前无法处理请求。这个状况是临时的，并且将在一段时间以后恢复。

502

简单来说 502 是报错类型代码 bad gateway 错误的网关。

产生错误的原因

连接超时 我们向服务器发送请求 由于服务器当前链接太多，导致服务器方面无法给于正常的响应，产生此类报错（由于服务器压力过大，不能及时处理 client 的请求导致服务器响应超时）

解救的办法：刷新

刷新也是有两种：

1. 点击刷新或者使用 F5 快捷键
   从本地磁盘上刷新，没有再次请求服务器
2. 访问服务器，或者右键

504

gateway time-out 顾名思义 网关超时

一般计算机中的超时就是配置错了，此处一般指 nginx 做反向代理服务器时，所连接的服务器 tomcat 无响应导致的。

502 已经与后端建立了连接，但超时；504 与后端连接未建立，超时。

500

Internal Server Error(内部服务器错误)

程序出错了时，就会返回 500 错误

501

Not Implemented(未实现)

客户端发起的请求超出服务器的能力范围(比如，使用了服务器不支持的请求方法)时，使用此状态码)。 **一般的 Web 服务器只支持 GET 和 POST 方法。**

实例： 使用 Fiddler Composer 给 www.qq.com， 发送一个 OPTIONS 方法的 Request. 服务器就能返回 501 了。

503

503 是一种 HTTPS 状态码。英文名 503 **Service Unavailable** 与 404（404 Not Found)是同属一种网页状态出错码。前者是服务器出错的一种返回状态，后者是网页程序没有相关的结果后返回的一种状态

301 Moved Permanently

被请求的资源已永久移动到新位置，并且将来任何对此资源的引用都应该使用本响应返回的若干个 URI 之一。如果可能，拥有链接编辑功能的客户端应当自动把请求的地址修改为从服务器反馈回来的地址。除非额外指定，否则这个响应也是可缓存的。

302 Move Temporarily

请求的资源临时从不同的 URI 响应请求。由于这样的重定向是临时的，客户端应当继续向原有地址发送以后的请求。只有在 Cache-Control 或 Expires 中进行了指定的情况下，这个响应才是可缓存的。

POST 和 GET

Where some see coincidence, I see consequence. Where others see chance, I see cost!

这里特指浏览器中非 Ajax 的 HTTP 请求

GET 是获取的意思，顾名思义就是获取信息。
GET 是默认的 HTTP 请求方法。
GET 方法把参数通过 key/value 形式存放在 URL 里面，如果参数是英文数字原样显示，如果是中文或者其他字符加密（Base64）URL 长度一般有限制所以 GET 方法的参数长度不能太长。由于参数显示再地址栏所以不安全，一般需要保密的请求不使用 GET。

POST 是邮件的意思，顾名思义就像一封信一样将参数放在信封里面传输。它用于修改服务器上的数据，一般这些数据是应该保密的，就像信件一样，信的内容只能收信的人看见。例入当用户输入账号和密码登录时账号和密码作为参数通过 HTTP 请求传输到服务器，这时候肯定不能用 GET 方法将账号密码直接显示再 URL 上，这时候就应该用 POST 方法保证数据的保密性。

POST 和 GET 的区别
1. GET 提交的数据放在 URL 中，POST 则不会。这是最显而易见的差别。这点意味着 GET 更不安全（POST 也不安全，因为 HTTP 是明文传输抓包就能获取数据内容，要想安全还得加密）
2. GET 回退浏览器无害，POST 会再次提交请求（GET 方法回退后浏览器再缓存中拿结果，POST 每次都会创建新资源）
3. GET 提交的数据大小有限制（是因为浏览器对 URL 的长度有限制，GET 本身没有限制），POST 没有
4. GET 可以被保存为书签，POST 不可以。这一点也能感受到。
5. GET 能被缓存，POST 不能
6. GET 只允许 ASCII 字符，POST 没有限制
7. GET 会保存再浏览器历史记录中，POST 不会。这点也能感受到。

总之，两者之间没有本质区别，区别就在于数据存储的位置。各自有适用环境，根据需求选择合适的方法即可。

GET 因为是读取，就可以对 GET 请求的数据做缓存。这个缓存可以做到浏览器本身上（彻底避免浏览器发请求），也可以做到代理上（如 nginx），或者做到 server 端（用 Etag，至少可以减少带宽消耗）

在页面里<form> 标签会定义一个表单。点击其中的 submit 元素会发出一个 POST 请求让服务器做一件事。这件事往往是有副作用的，不幂等的。
不幂等也就意味着不能随意多次执行。因此也就不能缓存。比如通过 POST 下一个单，服务器创建了新的订单，然后返回订单成功的界面。这个页面不能被缓存。试想一下，如果 POST 请求被浏览器缓存了，那么下单请求就可以不向服务器发请求，而直接
如果尝试重新执行 POST 请求，浏览器也会弹一个框提示下这个刷新可能会有副作用，询问要不要继续。

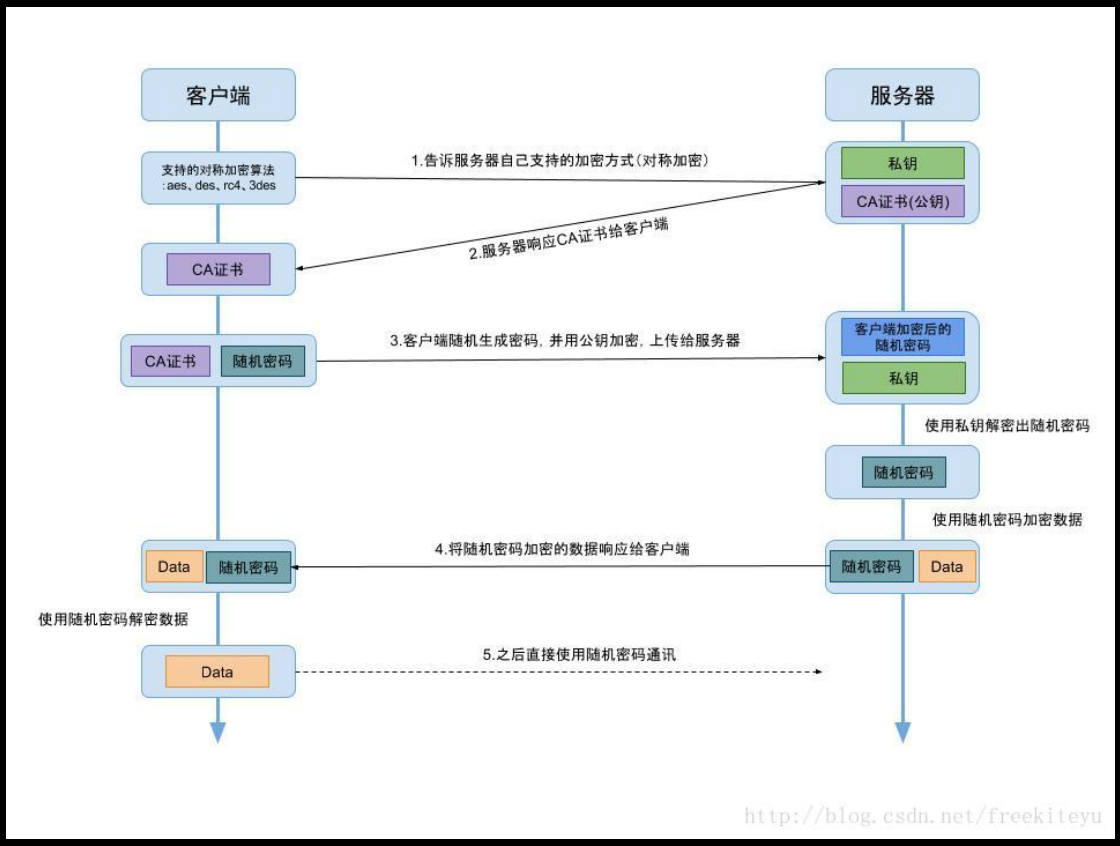
接口中的 GET 和 POST

这里是指通过浏览器的 Ajax api，或者 iOS/Android 的 App 的 http client，java 的 commons-httpclient/okhttp 或者是 curl,postman 之类的工具发出来的 GET 和 POST 请求。此时 GET/POST 不光能用在前端和后端的交互中，还能用在后端各个子服务的调用中（即当一种 RPC 协议使用）。尽管 RPC 有很多协议，比如 thrift，grpc，但是 http 本身已经有大量的现成的支持工具可以使用，并且对人类很友好，容易 debug。HTTP 协议在微服务中的使用是相当普遍的。

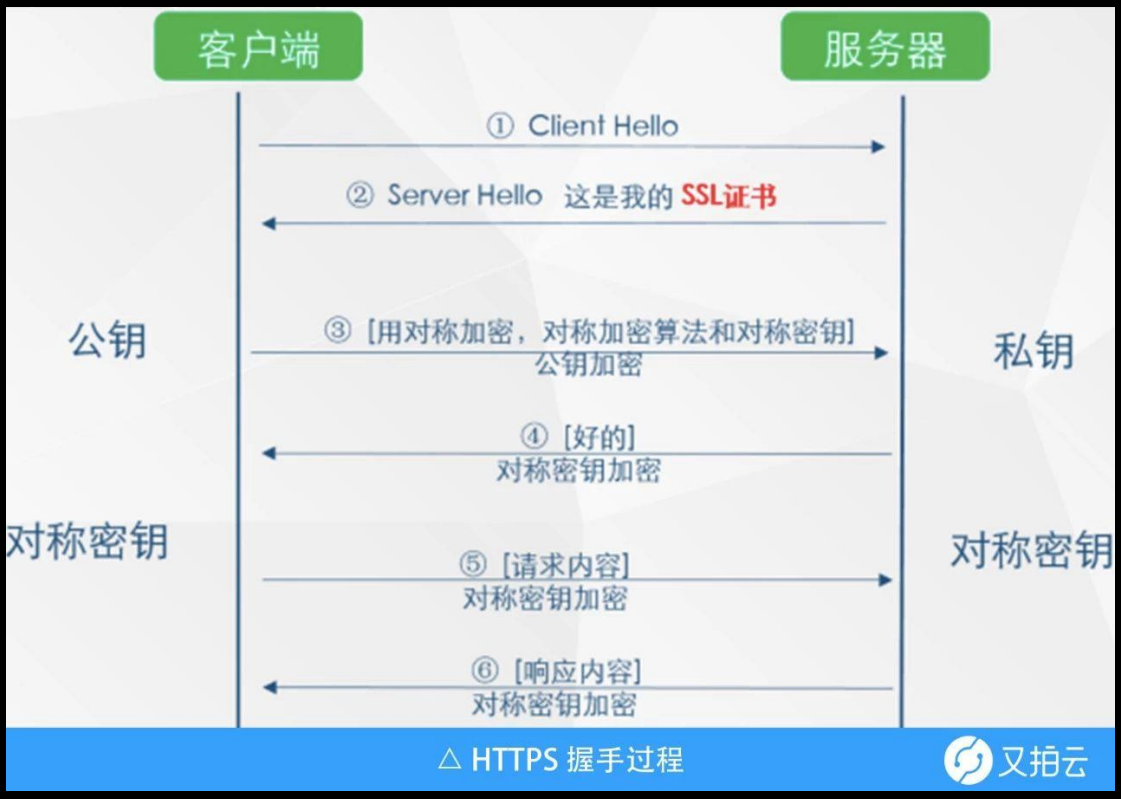
当用 HTTP 实现接口发送请求时，就没有浏览器中那么多限制了，只要是符合 HTTP 格式的就可以发

Http vs.https

超文本传输协议 HTTP 协议以明文方式发送内容，不提供任何方式的数据加密
安全套接字层超文本传输协议 HTTPS，为了数据传输的安全，HTTPS 在 HTTP 的基础上加入了 SSL 协议
　HTTPS 协议的主要作用可以分为两种：一种是建立一个信息安全通道，来保证数据传输的安全；另一种就是确认网站的真实性。



上图的随机密码应该是指的是对称加密密码



TLS/SSL 的核心三大步骤是：认证、秘钥协商、数据加密。

基于“对称密钥”的加密算法主要有 DES、TripleDES、RC2、RC4、RC5 和 Blowfish 等。
非对称加密算法：
RSA、Elgamal、背包算法、Rabin、D-H、ECC（椭圆曲线加密算法）。
使用最广泛的是 RSA 算法，Elgamal 是另一种常用的非对称加密算法。

HTTP 操作

创建(Create)、更新(Update)、读取(Retrieve)和删除(Delete)

1. GET：用来查询一下数据，不会修改、增加数据，不会影响资源的内容，即该请求不会产生副作用。无论进行多少次操作，结果都是一样的(幂等)。(R)
2. PUT：向服务器端发送数据的，从而改变信息，该请求就像数据库的 update 操作一样，用来修改数据的内容，但是不会增加数据的种类等。(U)
3. POST：同 PUT 请求类似，但是该请求会改变数据的种类等资源，就像数据库的 insert 操作一样，会创建新的内容。几乎目前所有的提交操作都是用 POST 请求的。(C)
4. DELETE：用来删除某一个资源的.(D)

POST 主要作用在一个集合资源之上的（url），而 PUT 主要作用在一个具体资源之上的（url/xxx），通俗一下讲就是，如 URL 可以在客户端确定，那么可使用 PUT，否则用 POST。

CURD:
1、POST /url 创建
2、DELETE /url/xxx 删除
3、PUT /url/xxx 更新
4、GET /url/xxx 查看

注意:
1. 对资源的增，删，改，查操作，其实都可以通过 GET/POST 完成，不需要用到 PUT 和 DELETE

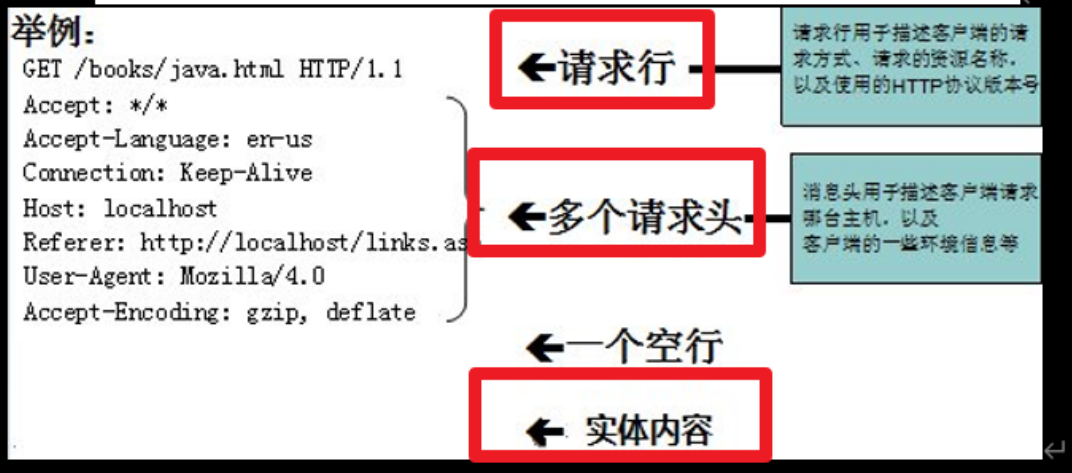
从表象上:
1. GET 请求的数据会附在 URL 之后(就是把数据放置在 HTTP 协议头中),以?分割 URL 和传输数据，参数之间以&相连。POST 把提交的数据则放置在是 HTTP 包的包体中。
2. POST 的安全性要比 GET 的安全性高。比如：通过 GET 提交数据，用户名和密码将明文出现在 URL 上，因为(1)登录页面有可能被浏览器缓存，(2)其他人查看浏览器的历史纪录，那么别人就可以拿到你的账号和密码了，除此之外，使用 GET 提交数据还可能会造成 **Cross-site request forgery** 攻击。

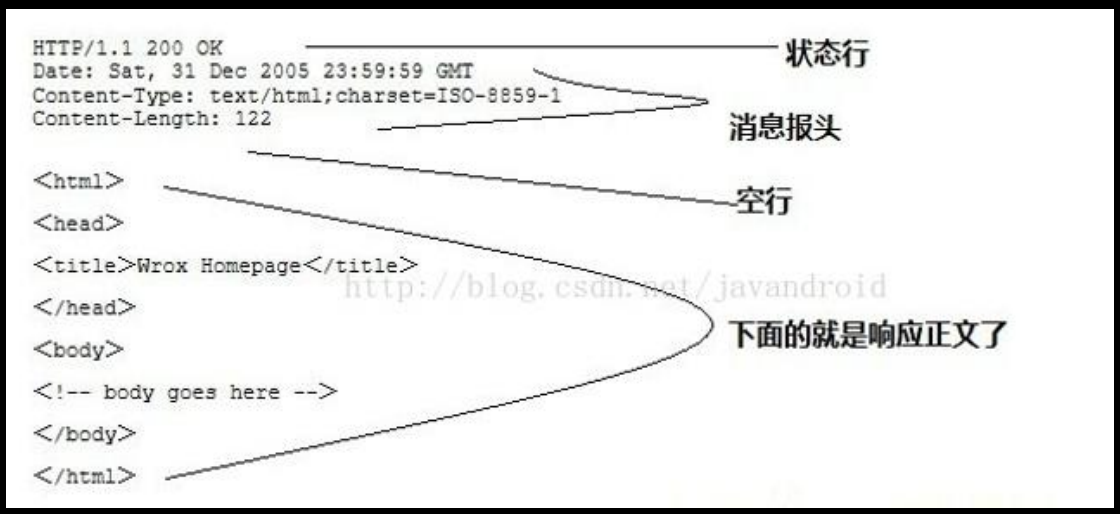
http 0.9\1.0\1.1\2.0\3.0

1. http/0.9:
   a) 仅 get
   b) 仅 HTML
2. http/1.0:
   a) POST、HEAD
   b) 多种数据格式 Content-Type
   c) Cache
   d) 头部变化
   e) 不支持持续性连接，性能差
3. http/1.1:
   a) 持续连接：Connection: close
   b) 管道机制、流水线处理：多个请求并发
   c) 增加请求方式：PUT、DELETE 等
4. http/2.0
   a) 双工模式、多路复用技术
   b) 服务器推送
   c) HTTP 报头压缩
   d) 请求优先级
5. http/3.0
   a) •••

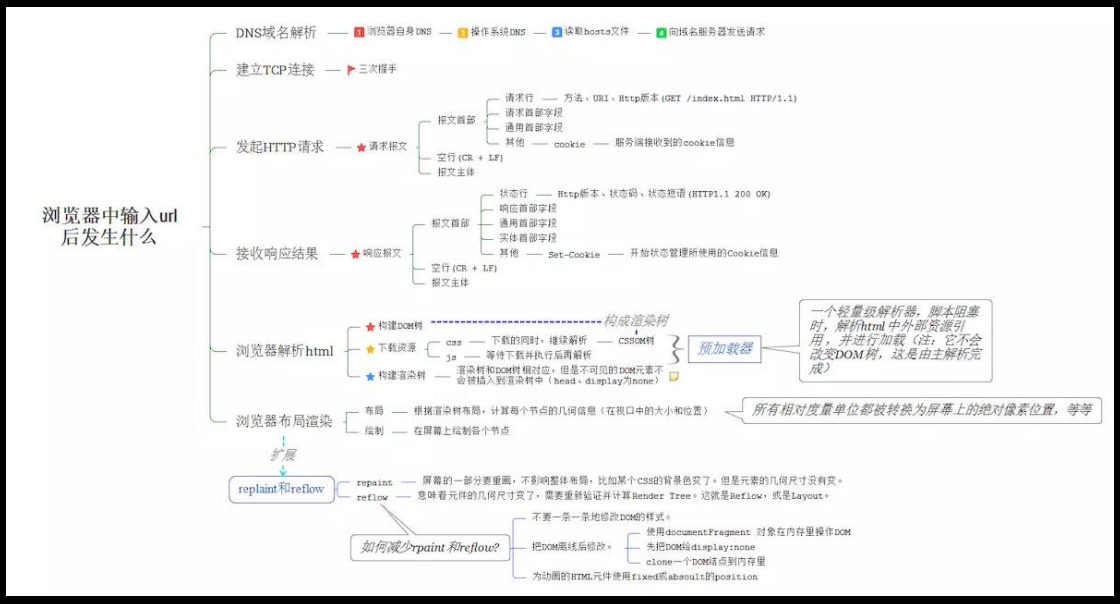
HTTP 报头

请求：





浏览器输入 URL 发生了什么



代理

正向代理与反向代理

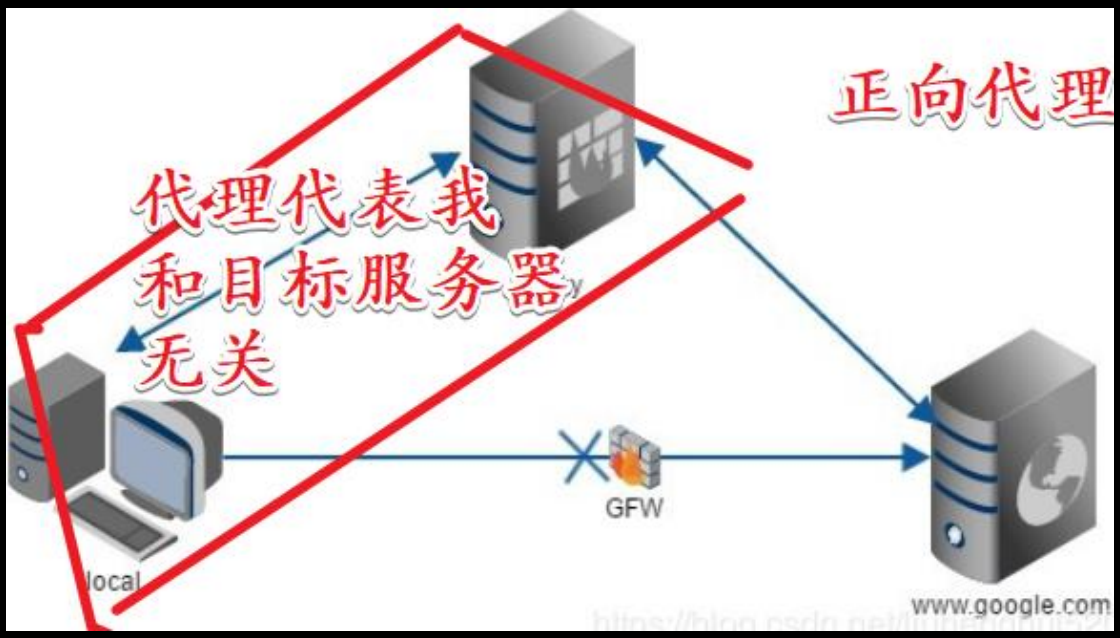
正向代理：

所谓正向代理就是顺着请求的方向进行的代理，即**代理服务器是由你配置为你服务，去请求目标服务器地址**。

比如我们要去访问谷歌网站，我们直接访问不通，那么我们就可以找一个代理服务器为我们服务，我们通过代理服务器请求到谷歌网站。对于谷歌而言他只知道有一个服务器访问了自己

举一个通俗的例子。你需要钱，C 正好有钱，但是你 C 不直接借给你。你和 B 关系比较好，B 可以找 C 借到钱。你和 B 沟通后，由 B 来找 C 借到钱后在给你。

两个例子中的共同特点是 **代理服务器和 B 都是你找到的，为你而服务的，代表你的利益。**我们还可以让代理服务器给你代理到推特、Facebook 等，他是代理的你。



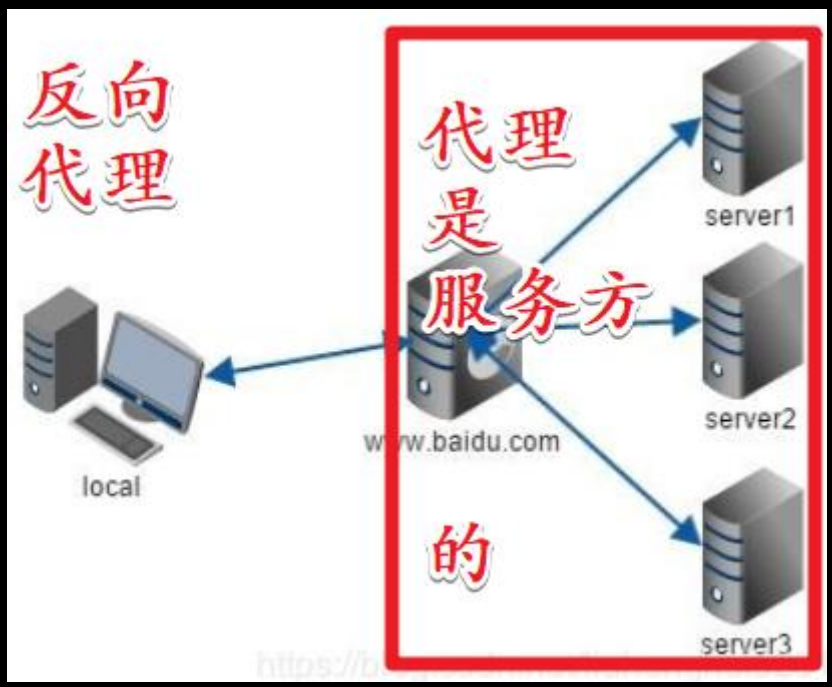
反向代理

所谓反向代理正好与正向代理相反，**代理服务器是为目标服务器服务的**，虽然整体的请求返回路线都是一样的都是 Client 到 Proxy 到 Server。

我们访问百度网站，百度的代理服务器对外的域名为 https://www.baidu.com 。具体内部的服务器节点我们不知道。现实中我们通过访问百度的代理服务器后，代理服务器给我们转发请求到他们 N 多的服务器节点中的一个给我们进行搜索后将结果返回。

我们同样需要钱，但是我们又不知道谁有钱，所以我们找了一家网贷平台，你提交资料后，网贷平台直接将钱打给你。但是你不知道，也不用关注网贷平台的钱从哪里来。网贷平台内部他们可能从哪一个财主哪里融的钱。对你而言网贷平台和他们的金主是一起的。

代理服务器和后面的目标主机是一个系统的。他们是对外提供服务的，所以称为反向代理



DNS

概述

Domain Name System

因特网的域名结构

由于因特网的用户数量较多，所以因特网在命名时采用的是层次树状结构的命名方法。任何一个连接在因特网上的主机或路由器，都有一个唯一的层次结构的名字，即域名(domain name)。这里，“域”(domain)是名字空间中一个可被管理的划分。

级别最低的域名写在最左边，而级别最高的字符写在最右边
域名由英文和数字、连字符(-)组成
每一级域名不超过 63 个字符，
完整域名总共不超过 255 个字符

各级域名由其上一级的域名管理机构管理，而最高的顶级域名则由 ICANN 进行管理。

顶级域名 **TLD(Top Level Domain)**已有 265 个，分为三大类：

1. 国家顶级域名 nTLD：采用 ISO3166 的规定。如：cn 代表中国，us 代表美国，uk 代表英国，等等。国家域名又常记为 ccTLD(cc 表示国家代码 contry-code)。
2. 通用顶级域名 gTLD：最常见的通用顶级域名有 7 个，即：com(公司企业)，net(网络服务机构)，org(非营利组织)，int(国际组织)，gov(美国的政府部门)，mil(美国的军事部门)。
3. 基础结构域名(infrastructure domain)：这种顶级域名只有一个，即 arpa，用于反向域名解析，因此称为反向域名。

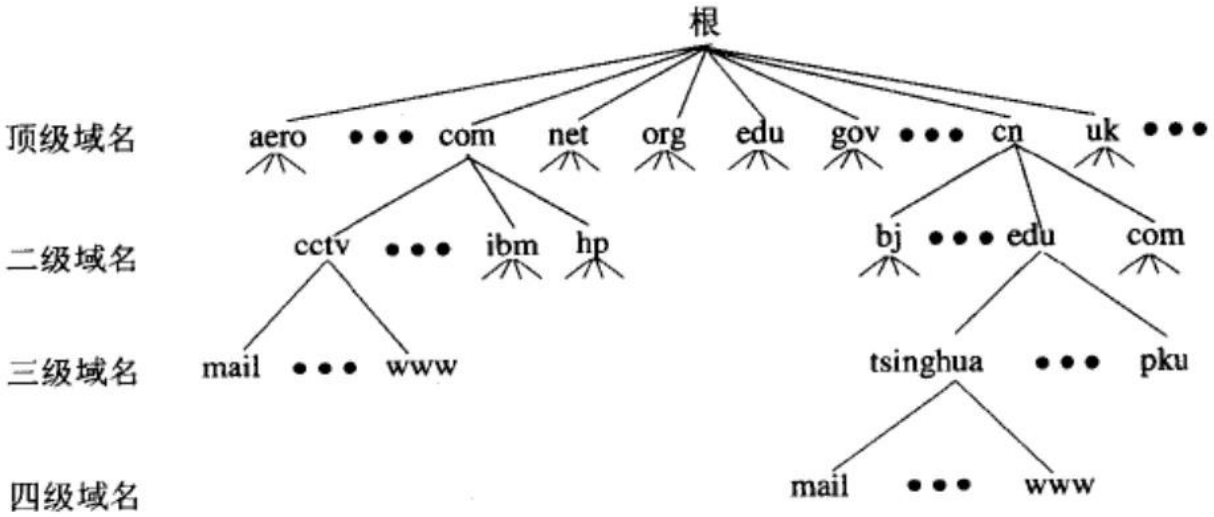


图 6-1　因特网的域名空间

域名服务器

如果采用上述的树状结构，每一个节点都采用一个域名服务器，这样会使得域名服务器的数量太多，使域名服务器系统的运行效率降低。所以**在 DNS 中，采用划分区的方法来解决**。

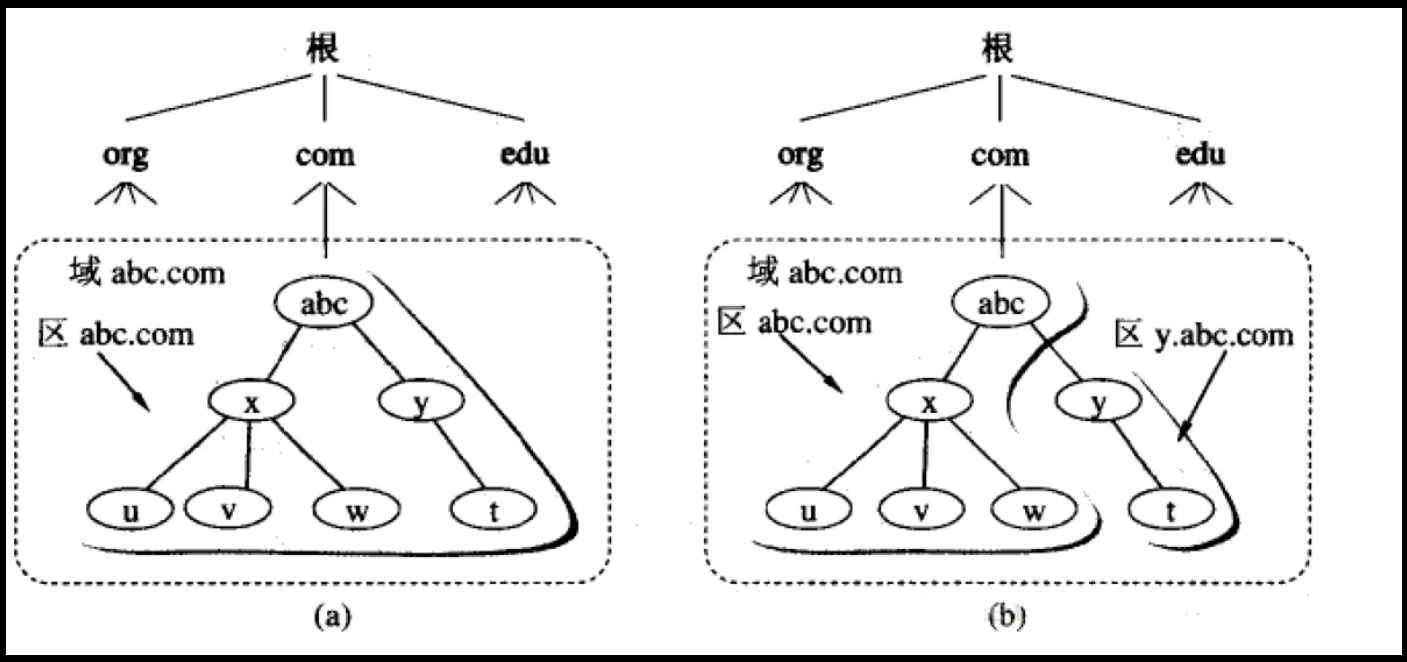
一个服务器所负责管辖(或有权限)的范围叫做区(zone)。各单位根据具体情况来划分自己管辖范围的区。但在一个区中的所有节点必须是能够连通的。

每一个区设置相应的权限域名服务器，用来保存该区中的所有主机到域名 IP 地址的映射。

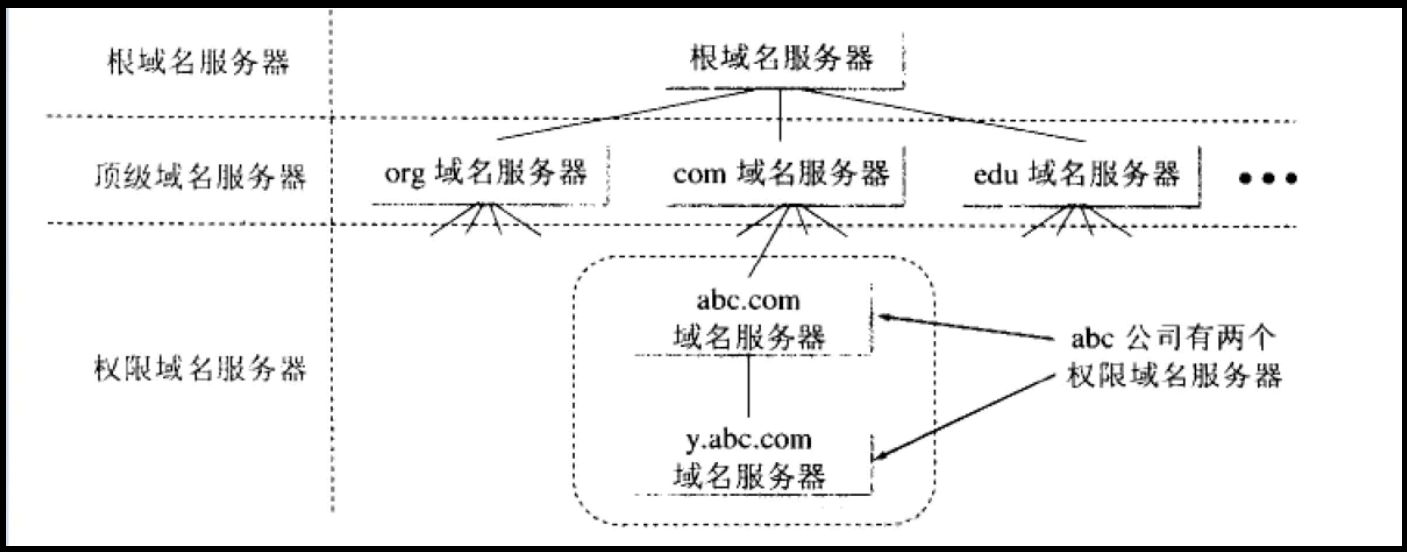
区←→权限域名服务器

总之，DNS 服务器的管辖范围不是以“域”为单位，而是以“区”为单位。区是 DNS 服务器实际管辖的范围。区 <= 域

下图是区的不同划分方法的举例。假定 abc 公司有下属部门 x 和 y，部门 x 下面有分三个分布们 u,v,w，而 y 下面还有下属部门 t。图 a 表示 abc 公司只设一个区 abc.com。这是，区 abc.com 和域 abc.com 指的是同一件事。但图 b 表示 abc 公司划分为两个区：abc.com 和 y.abc.com。这两个区都隶属于域 abc.com，都各设置了相应的权限域名服务器。不难看出，区是域的子集。



下图是以上图 b 中 abc 公司划分的两个区为例，给出了 DNS 域名服务器树状结构图。这种 DNS 域名服务器树状结构图可以更准确地反映出 DNS 的分布式结构。图中的每一个域名服务器都能够部分域名到 IP 地址的解析。当某个 DNS 服务器不能进行域名到 IP 地址的转换时，它就会设法找因特网上别的域名服务器进行解析。



因特网上的 DNS 服务器也是按照层次安排的。每一个域名服务器只对域名体系中的一部分进行管辖。根据域名服务器所起的作用，可以把域名服务器划分为下面四种不同的类型。

根域名服务器：

最高层次的域名服务器，也是最重要的域名服务器。所有的**根域名服务器都知道所有的顶级域名服务器的域名和 IP 地址**。不管是哪一个**本地域名服务器，若要对因特网上任何一个域名进行解析，只要自己无法解析，就首先求助根域名服务器**。所以根域名服务器是最重要的域名服务器。假定**所有的根域名服务器都瘫痪了，那么整个 DNS 系统就无法工作**。需要注意的是，在很多情况下，**根域名服务器并不直接把待查询的域名直接解析出 IP 地址，而是告诉本地域名服务器下一步应当找哪一个顶级域名服务器进行查询**。

顶级域名服务器：负责管理在该顶级域名服务器注册的二级域名。

本地域名服务器：本地服务器不属于上图的域名服务器的层次结构，但是它对域名系统非常重要。当一个主机发出 DNS 查询请求时，这个查询请求报文就发送给本地域名服务器。

本地域名是任何人都可以访问的，权限域名可以在服务器里面设置特定人群访问

域名的解析过程

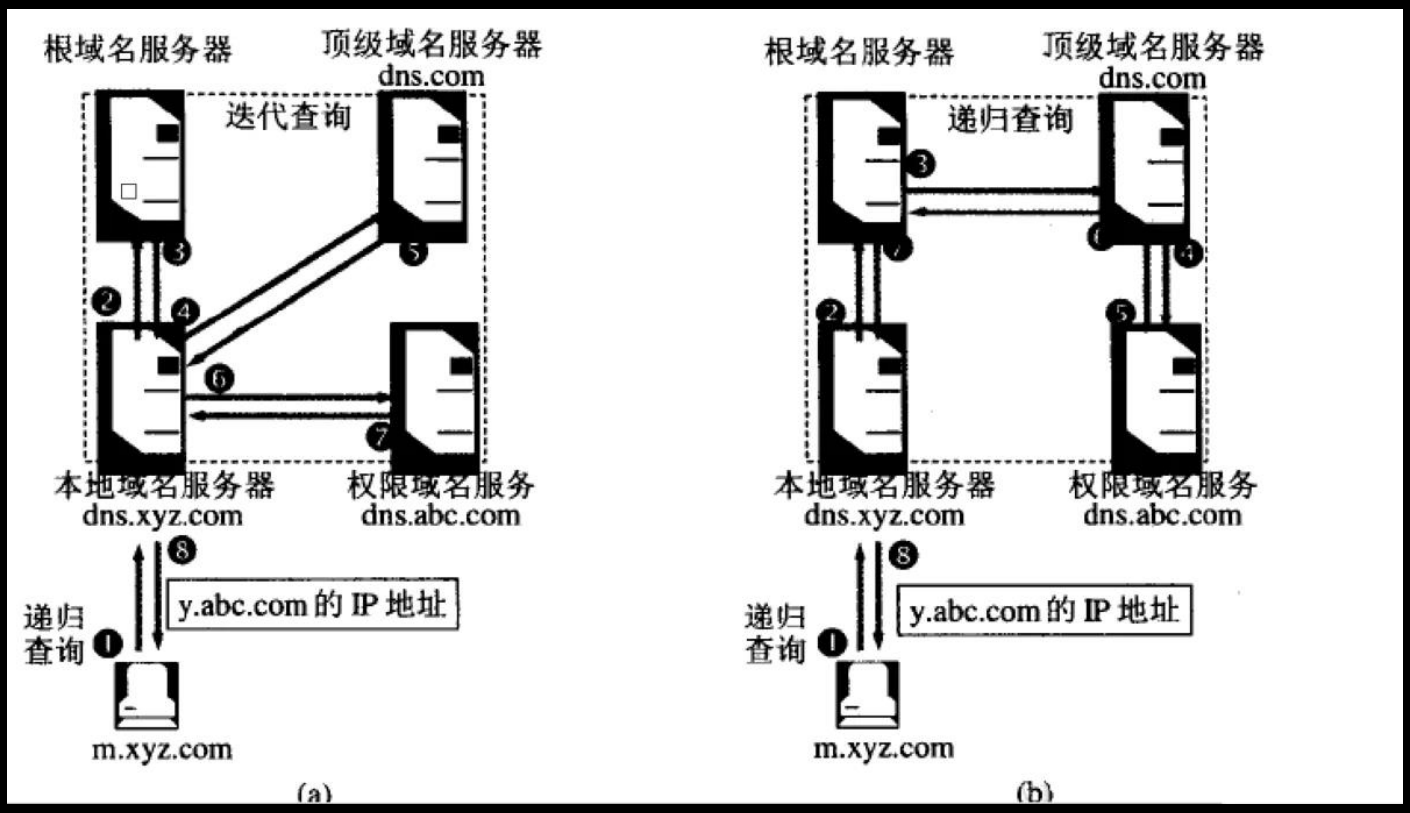
1. **主机向本地域名服务器**的查询一般都是采用**递归查询**

   递归查询：

   如果主机所询问的本地域名服务器不知道被查询的域名的 IP 地址，那么本地域名服务器就以 DNS 客户的身份，向其它根域名服务器继续发出查询请求报文(即替主机继续查询)，而不是让主机自己进行下一步查询。因此，递归查询返回的查询结果或者是所要查询的 IP 地址，或者是报错，表示无法查询到所需的 IP 地址。

2. **本地域名服务器向根域名服务器的查询的迭代查询**

   迭代查询的特点：当根域名服务器收到本地域名服务器发出的迭代查询请求报文时，要么给出所要查询的 IP 地址，要么告诉本地服务器：“你下一步应当向哪一个域名服务器进行查询”。然后让本地服务器进行后续的查询。根域名服务器通常是把自己知道的顶级域名服务器的 IP 地址告诉本地域名服务器，让本地域名服务器再向顶级域名服务器查询。顶级域名服务器在收到本地域名服务器的查询请求后，要么给出所要查询的 IP 地址，要么告诉本地服务器下一步应当向哪一个权限域名服务器进行查询。最后，知道了所要解析的 IP 地址或报错，然后把这个结果返回给发起查询的主机。



域名为 m.xyz.com 的主机想知道另一个主机 y.abc.com 的 IP 地址

上图 a 的几个查询步骤：

1、主机 m.abc.com 先向本地服务器 dns.xyz.com 进行递归查询。

2、本地服务器采用迭代查询。它先向一个根域名服务器查询。

3、根域名服务器告诉本地服务器，下一次应查询的顶级域名服务器 dns.com 的 IP 地址。

4、本地域名服务器向顶级域名服务器 dns.com 进行查询。
5、顶级域名服务器 dns.com 告诉本地域名服务器，下一步应查询的权限服务器 dns.abc.com 的 IP 地址。
6、本地域名服务器向权限域名服务器 dns.abc.com 进行查询。
7、权限域名服务器 dns.abc.com 告诉本地域名服务器，所查询的主机的 IP 地址。
8、本地域名服务器最后把查询结果告诉 m.xyz.com。

DNS 为什么使用 UDP

以浏览器响应时间= DNS 域名解析时间 + TCP 连接建立时间 + HTTP 交易时间
采用 TCP 传输，则域名解析时间为：
DNS 域名解析时间 = TCP 连接时间 + DNS 交易时间
采用 UDP 传输，则域名解析时间为：
DNS 域名解析时间 = DNS 交易时间
DNS 是面向无连接的！
不就多一次 TCP 连接时间吗？No
在很多时候，用户在访问一些冷门网站时，由于 DNS 服务器没有冷门网站的解析缓存，需要到域名根服务器、一级域名服务器、二级域名服务器迭代查询，直到查询到冷门网站的权威服务器，这中间可能涉及到多次的查询。

DNS 解析过程

1. 查询浏览器自身的 DNS 缓存。
   如 Chrome 中：chrome://net-internals/#dns
2. 查询操作系统的 DNS 缓存：
   如 Windows 下：ipconfig /displaydns

```
time.windows.com
----------------------------------------
Record Name . . . . . : time.windows.com
Record Type . . . . . : 5
Time To Live  . . . . : 1
Data Length . . . . . : 8
Section . . . . . : Answer
CNAME Record  . . . . : time.microsoft.akadns.net


Record Name . . . . . : time.microsoft.akadns.net
Record Type . . . . . : 1
Time To Live  . . . . : 1
Data Length . . . . . : 4
Section . . . . . : Answer
A (Host) Record . . . : 52.231.114.183
```

3. 访问 Hosts 文件
   Win：C:\Windows\System32\drivers\etc\hosts
   Linux：/etc/hosts

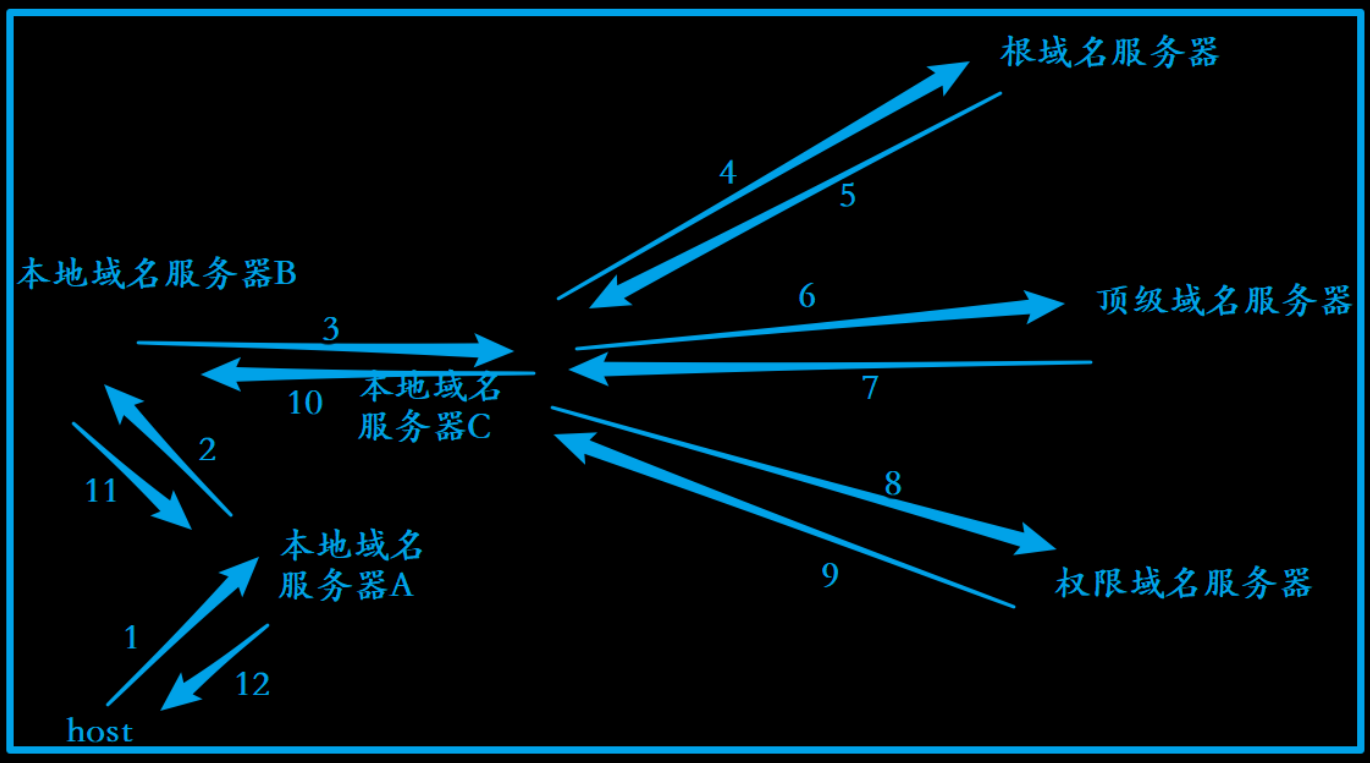
```
140.82.114.4 github.com
140.82.113.3 github.global.SSL.fastly.net

192.168.43.141 host.docker.internal
192.168.43.141 gateway.docker.internal
192.168.100.2 host.docker.internal
192.168.100.2 gateway.docker.internal
192.168.124.5 host.docker.internal
192.168.124.5 gateway.docker.internal
# Added by Docker Desktop
192.168.100.5 host.docker.internal
192.168.100.5 gateway.docker.internal
# To allow the same kube context to work on the host and the
container:
127.0.0.1 kubernetes.docker.internal
# End of section
```

4. 访问域名服务器

递归 vs. 迭代

1. 递归：主机向本地域名服务器查询一般使用递归(主机请求本地服务器 A，A 不知道，A 以 DNS 客户身份访问 B，B 不知道，再访问本地域名服务器 C，C 知道，原路返回)
2. 迭代：本地域名服务器向根域名服务器一般以迭代方式进行(本地域名服务器询问根域名服务器，A 说你去访问顶级域名服务器 B，访问 B 后 B 说你去访问权限域名服务器 C，C 知道，返回给本地域名服务器)
   a) 递归是服务服务器帮你访问
   b) 迭代是自己去访问
   c) 迭代和递归都可以，不一定是绝对的。



默认配置

默认端口：53
默认刷新周期：1 天

Socket

套接字

网络编程就是编写程序使两台联网的计算机相互交换数据。这就是全部内容了吗？是的！网络编程要比想象中的简单许多。
那么，这两台计算机之间用什么传输数据呢？首先需要物理连接。如今大部分计算机都已经连接到互联网，因此不用担心这一点。
在此基础上，只需要考虑如何编写数据传输程序。但实际上这点也不用愁，因为**操作系统已经提供了 socket**。即使对网络数据传输的原理不太熟悉，我们也能通过 socket 来编程。

socket 的原意是“插座”，在计算机通信领域，socket 被翻译为“**套接字**”，它**是计算机之间进行通信的一种约定或一种方式**。通过 socket 这种约定，一台计算机可以接收其他计算机的数据，也可以向其他计算机发送数据。

Where some see coincidence, I see consequence. Where others see chance, I see cost!

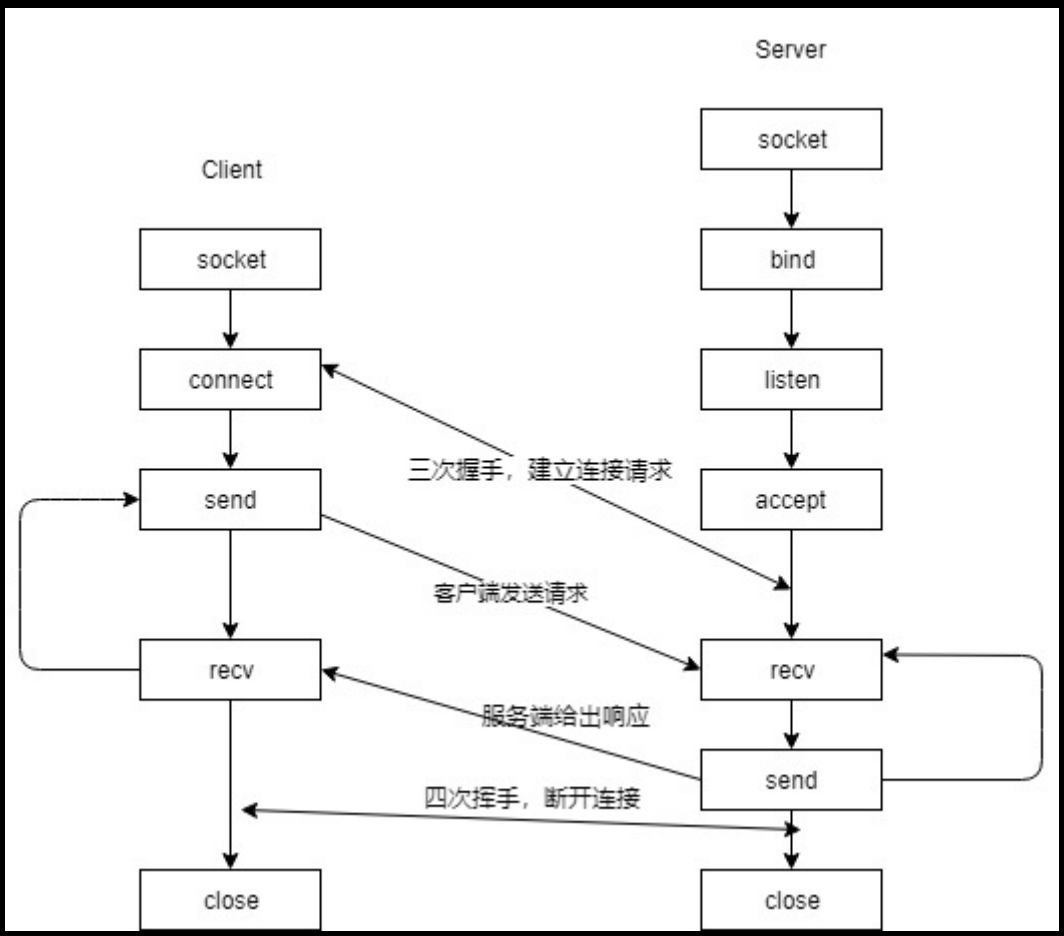
socket 的典型应用就是 Web 服务器和浏览器：浏览器获取用户输入的 URL，向服务器发起请求，服务器分析接收到的 URL，将对应的网页内容返回给浏览器，浏览器再经过解析和渲染，就将文字、图片、视频等元素呈现给用户。

UNIX/Linux 程序在执行任何形式的 I/O 操作时，都是在读取或者写入一个文件描述符。一个文件描述符只是一个和打开的文件相关联的整数，它的背后可能是一个硬盘上的普通文件、FIFO、管道、终端、键盘、显示器，甚至是一个网络连接。
(Linux:普通木块接管了字节

- 普通
- 目录
- 块文件
- 连接
- 管道
- 字符

套接字)

Windows 也有类似“文件描述符”的概念，但通常被称为“文件句柄”。



粘包问题

Where some see coincidence, I see consequence. Where others see chance, I see cost!

假设我们希望客户端每次发送一位学生的学号，让服务器端返回该学生的姓名、住址、成绩等信息，这时候可能就会出现问题，服务器端不能区分学生的学号。例如第一次发送 1，第二次发送 3，服务器可能当成 13 来处理，返回的信息显然是错误的。

这就是数据的“粘包”问题，**客户端发送的多个数据包被当做一个数据包接收**。也称数据的**无边界性**，read()/recv() 函数不知道数据包的开始或结束标志（实际上也没有任何开始或结束标志），只把它们当做连续的数据流来处理。

通信

RPC vs. Restful

RESTful 是一种**软件架构风格**，用于约束客户端和服务器交互，满足这些约束条件和原则的应用程序或设计就是 RESTful。比如 HTTP 协议使用同一个 URL 地址，通过 GET，POST，PUT，DELETE 等方式实现查询、提交、删除数据。RPC 是远程过程调用，是用于解决分布式系统服务间调用的一种方式。RPC 采用客户端与服务端模式，双方通过约定的接口（常见为通过 IDL 定义或者是代码定义）以类似本地方法调用的方式来进行交互，客户端根据约定传输调用函数+参数给服务端（一般是网络传输 TCP/UDP），服务端处理完按照约定将返回值返回给客户端。

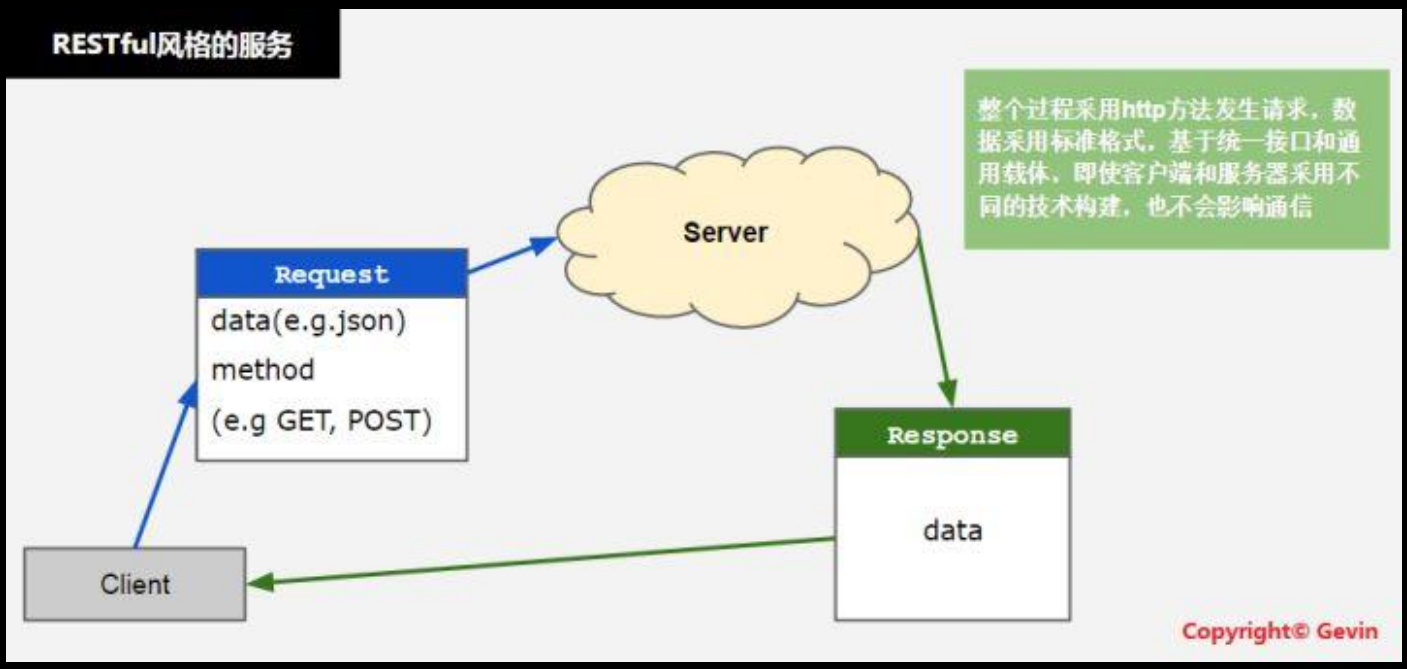
从使用方面看，Http 接口只关注服务提供方，对于客户端怎么调用，调用方式怎样并不关心

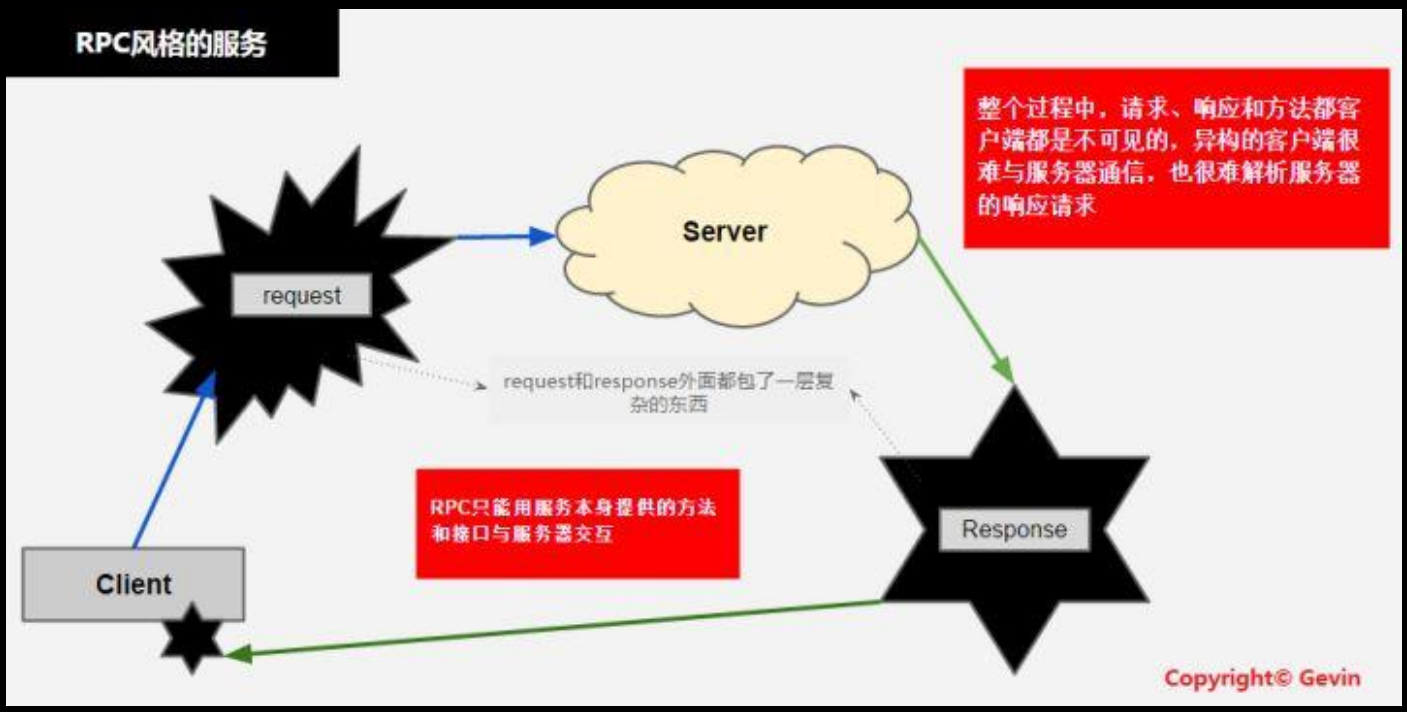
从性能角度看，使用 Http 时，Http 本身提供了丰富的状态功能与扩展功能，但也正由于 Http 提供的功能过多，导致在网络传输时，需要携带的信息更多，从性能角度上讲，较为低效。而 RPC 服务网络传输上仅传输与业务内容相关的数据，传输数据更小，性能更高。

REST 是面向资源的

左边是错误的设计，而右边是正确的：
GET /rest/api/getDogs --> GET /rest/api/dogs 获取所有小狗狗
GET /rest/api/addDogs --> POST /rest/api/dogs 添加一个小狗狗
GET /rest/api/editDogs/:dog_id --> PUT /rest/api/dogs/:dog_id 修改一个小狗狗
GET /rest/api/deleteDogs/:dog_id --> DELETE /rest/api/dogs/:dog_id 删除一个小狗狗

REST 很好地利用了 HTTP 本身就有的一些特征，如 HTTP 动词、HTTP 状态码、HTTP 报头等等





Where some see coincidence, I see consequence. Where others see chance, I see cost!

Where some see coincidence, I see consequence. Where others see chance, I see cost!