1. TCP/IP网络模型层数和功能

1.1应用层：专注为用户提供应用功能如HTTP、FTP、Telnet、DNS等，不关心数据如何传输，在用户态工作，其它层在内核态。

1.2传输层：为应用层提供网络支持（服务应用，实际的传输功能交给网络层）

两个协议：TCP和UDP

TCP（Transmission Control Protocol 传输控制协议），大部分应用使用TCP协协议如HTTP，相较UDP多了流量控制、超时重传、拥塞控制等，保证可靠性。

UDP只负责发送数据包，不保证能否抵达，实时性和效率好。（在应用层实现TCP特性可以使UDP实现可靠传输）

传输数据包大于MSS（TCP最大报文段长度）需要将数据包分成TCP段（TCP Segment），一个段丢失只需重新发送这个TCP段。

一台设备上有多个应用接收或传输数据，需要一个编号（端口）将应用区分开，传输层报文中会携带端口号

1.3网络层：负责将数据从一个设备传到另一个设备

IP协议（Internet Protocol）：将传输层的报文作为数据部分，加上IP头部组成IP报文，IP报文大小超过MTU（以太网中一般1500字节）会再次分片。寻址和路由

IP地址给设备编号，IPv4协议：IP地址32位，分四段。一个IP地址被分成（配合子网掩码按位与）：

网路号：标识该IP地址属于哪个子网

主机号：标识同一子网下的不同主机

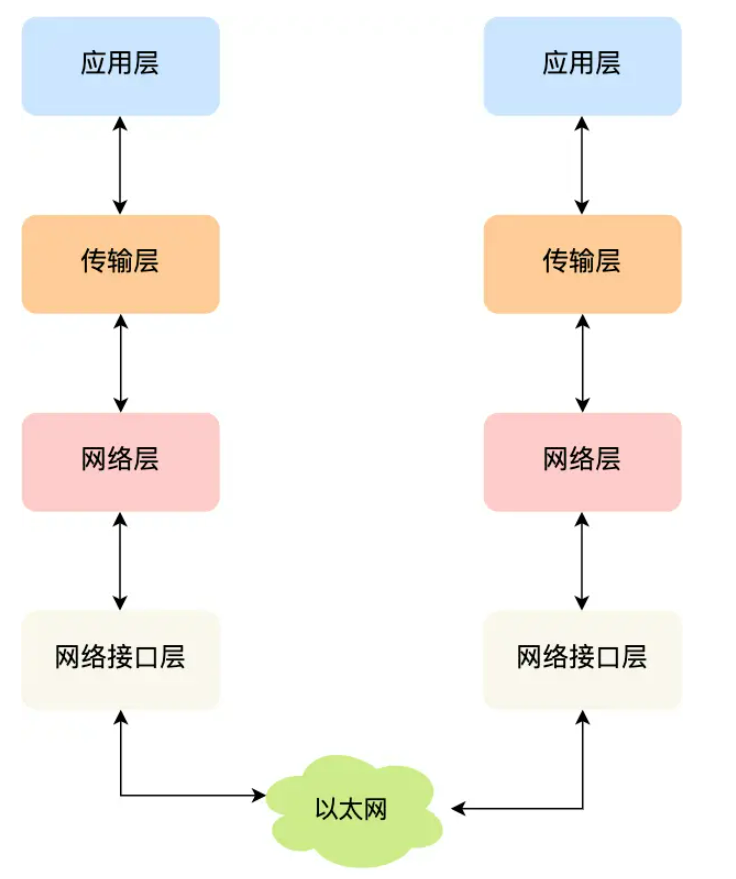
寻址过程中先匹配网络号再寻找对应主机

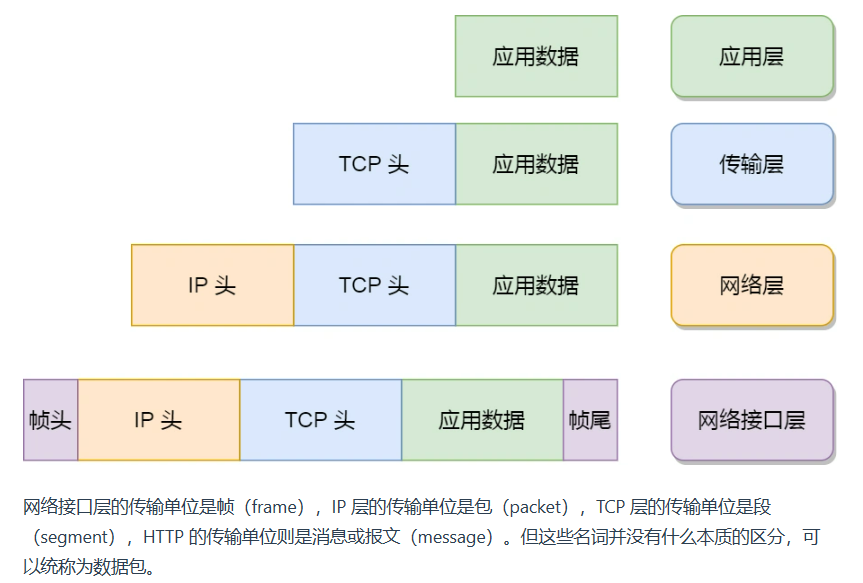
路由：通过路由算法决定下一步的路径

1.4网络接口层：为网络层提供链路级别传输服务，在以太网、WIFI这样的底层网络发送原始数据包，在这个层次上使用MAC地址来标识网络设备

在IP头部前面加上MAC头，封装成数据帧发送到网络上

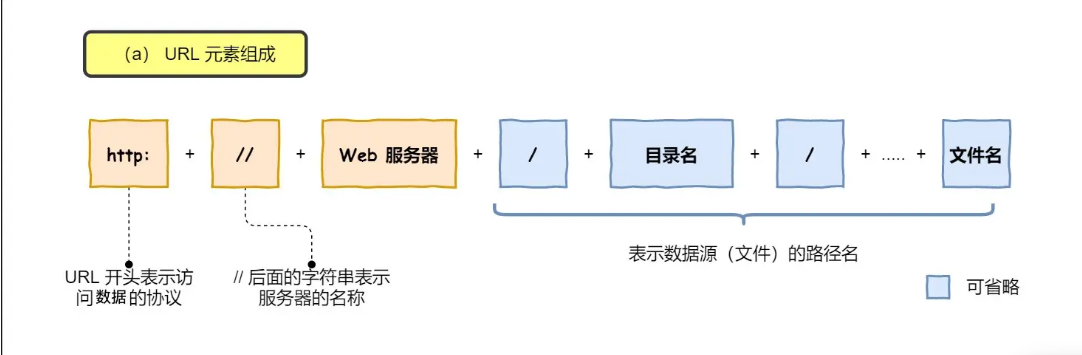
MAC头包含接收方和发送方的MAC地址，可以通过ARP协议获取对方的MAC地址





1. 键入网址到网页显示期间发生了什么？
2. HTTP

浏览器解析URL



生成HTTP请求信息

请求报文：

请求行：方法 空格 URL 空格 协议版本 回车换行

消息头：首部字段名：值 回车换行

回车换行

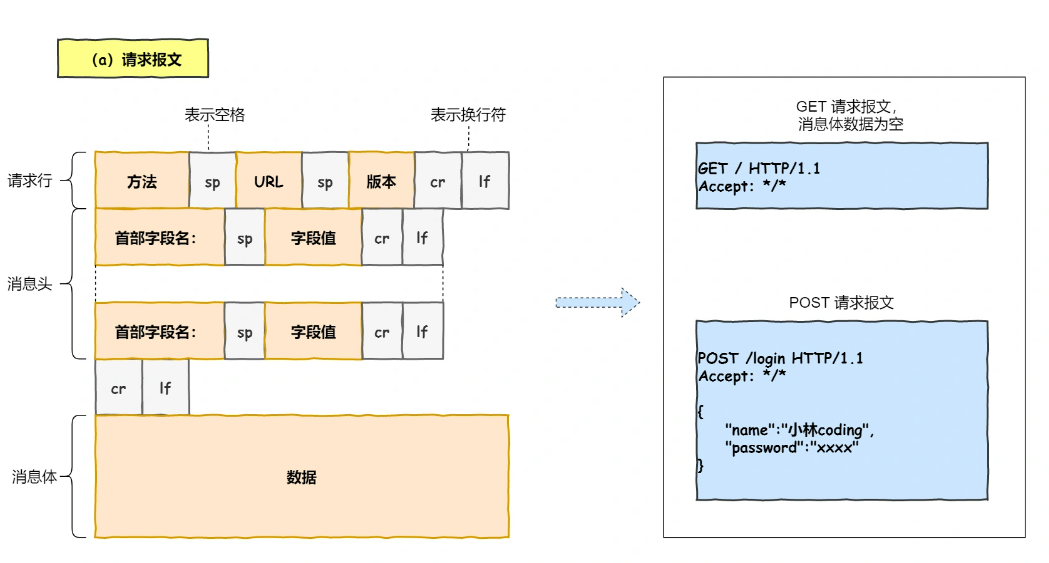
消息体：数据

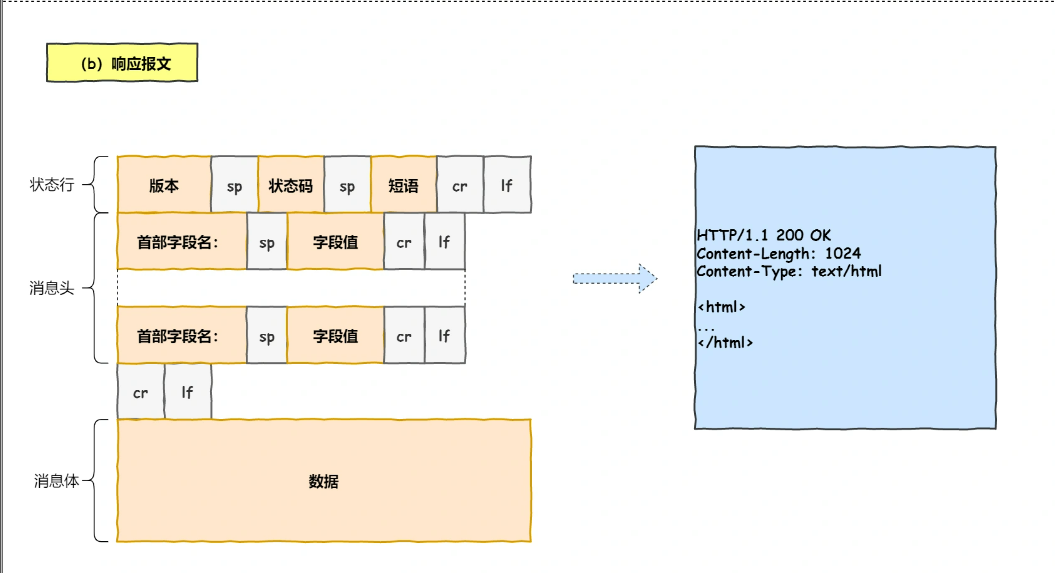
响应报文：

状态行：协议版本 空格 状态码 空格 短语 回车换行

消息头：首部字段名：值 回车换行

回车换行

消息体：数据 



怎么保护请求报文？发到哪？

1. 真实地址查询：DNS（domain name system，域名系统）

生成HTTP请求报文后要委托操作系统发送给web服务器，需要查询服务器域名对应的IP地址（委托操作系统发送消息时需要提供通信对象的地址）

DNS服务器专门保存了web服务器域名与IP的对应关系

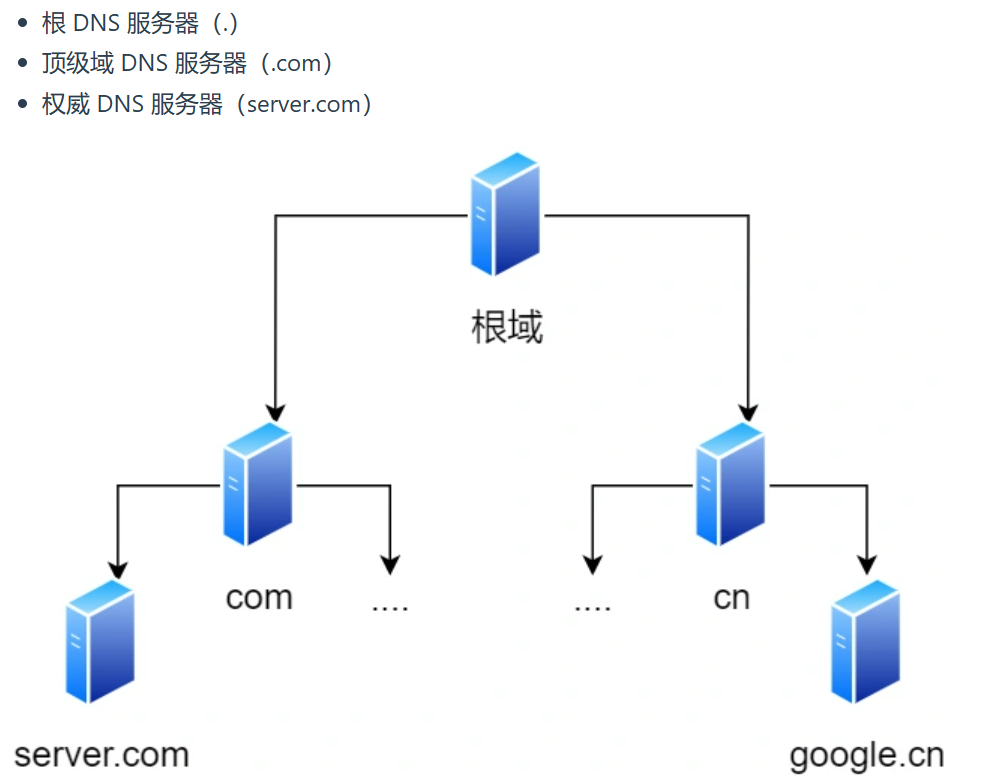
**域名的层级关系**：DNS中的域名用句点分割，越靠右层级越高，实际的域名最后还有一个点代表根域名

eg：[www.baidu.com](http://www.baidu.com) 实际上为[www.baidu.com](http://www.baidu.com).

根DNS服务器（.）

顶级域名服务器(.com)

权威DNS服务器（baidu.com）



根域的DNS服务器信息保存在互联网中所有的DNS服务器，客户端找到任何一台DNS服务器，然后找到根域DNS服务器，再随之找到根域下的某台目标DNS服务器

域名解析工作流程：

客户端发出一个DNS请求，寻找[www.baidu.com](http://www.baidu.com)的IP，并发给本地DNS服务器（客户端TCP/IP设置中填写的DNS服务器）

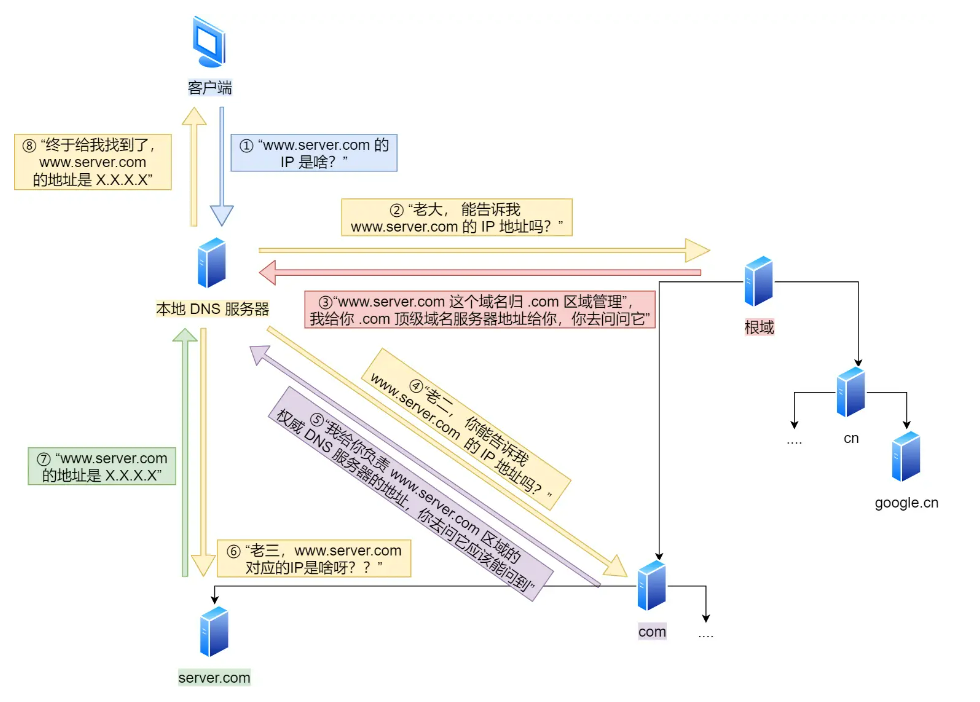
本地DNS服务器收到请求后，如果缓存中能找到对应域名的IP就直接返回，否则访问根域DNS服务器，并获得顶级DNS服务器地址

访问顶级DNS服务器并获得权威DNS服务器地址

权威DNS服务器将对应的IP地址发给本地DNS服务器

本地DNS服务器将IP返回给客户端

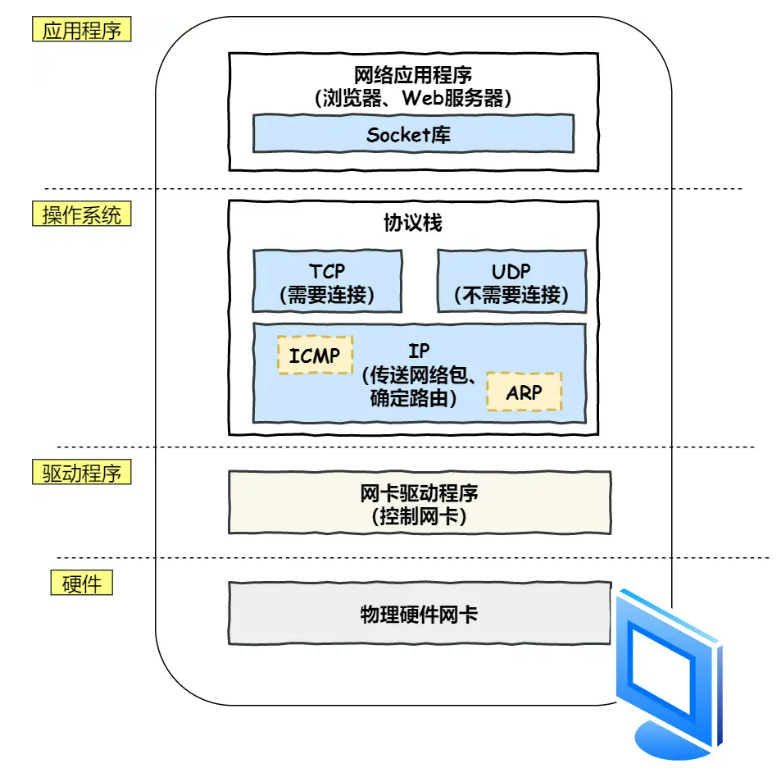
客户端和目标建立连接



不是每次域名解析都要经过这么多步骤：浏览器缓存没有-> 操作系统缓存->hosts文件->本地DNS服务器

获取到IP，如何发送？

1. 协议栈



协议栈上半部分是负责收发数据的TCP和UDP协议，接收应用层的委托并执行收发数据的操作

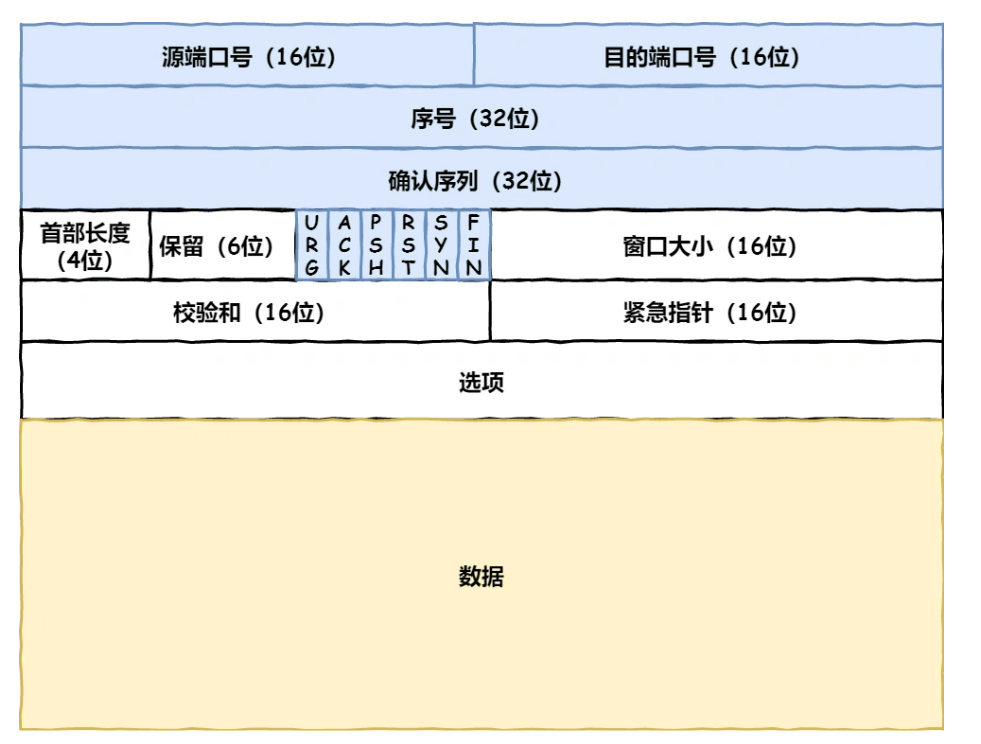
下半部分是用IP协议控制网络包收发操作，IP中还包括ICMP和ARP协议

ICMP用于告知网络包传送过程中产生的错误及控制信息

ARP用于根据IP地址查询相应的以太网MAC地址

1. 可靠传输：TCP

TCP报文头部：



源端口号和目的端口号：必不可少，指定发送的应用

序号：解决乱序问题

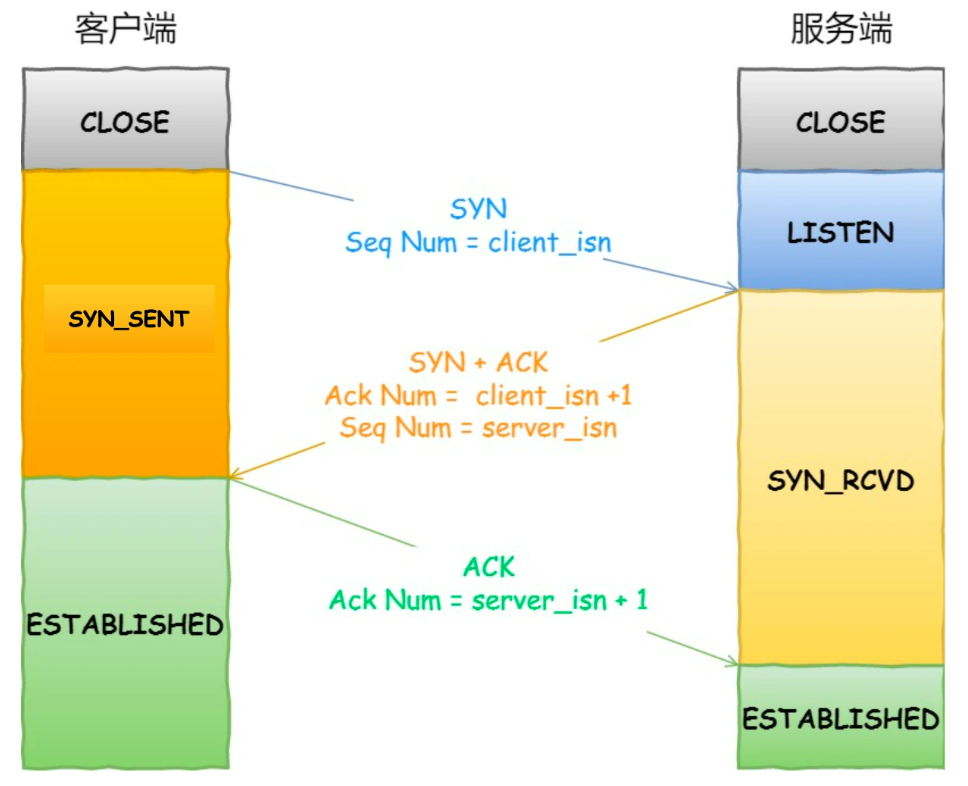
确认号：确认对方是否收到，没收到就重新发送直到送达，为了解决丢包问题

状态位：SYN发起连接，ACK回复，RST重新连接，FIN结束连接

窗口大小：用于TCP流量控制，双方各声明一个窗口标识自己当前的处理能力

HTTP传输数据前需要建立TCP连接

TCP传输数据前要三次握手建立连接：



开始服务端和客户端都处于CLOSED状态，服务端主动监听某个端口，处于LISTEN

客户端主动发起连接状态位SYN，处于SYN-SENT状态

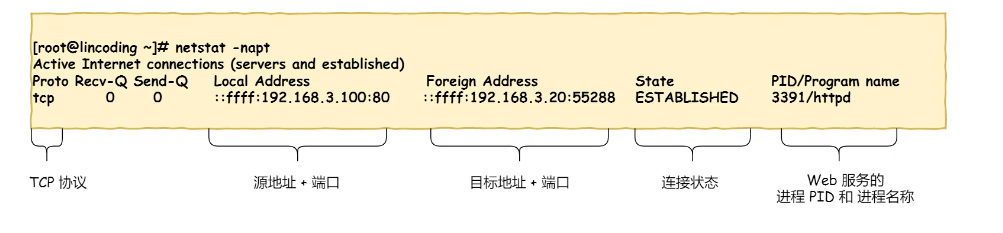
服务端收到连接，发送SYN并ACK客户端的SYN，处于SYN-RCVD状态

客户端收到服务端发送的SYN和ACK，发送SYN的ACK，处于ESTABLISHED状态

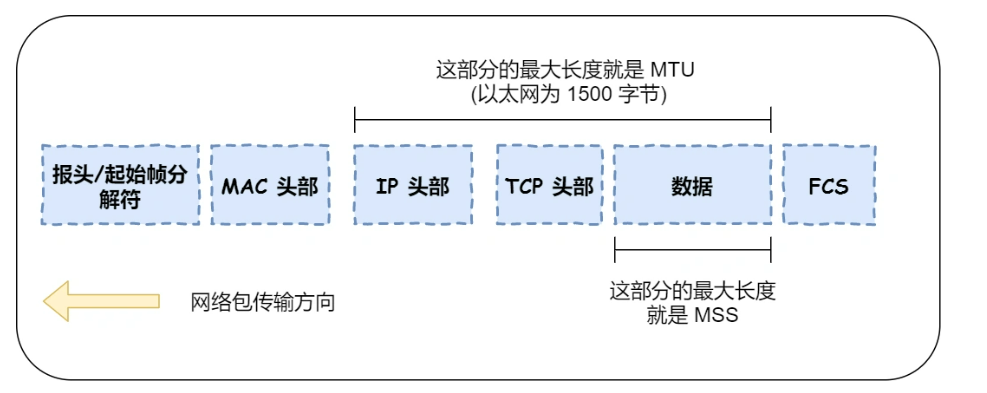
服务端收到ACK后处于ESTABLISHED状态

三次握手保证双方都有发送和接收的能力

Linux查看TCP连接状态：netstat -napt



TCP分割数据：发送的HTTP请求信息超过MSS长度，TCP需要把HTTP数据拆成一块块的数据



MTU（Maximum Transmission Unit 最大传输单元）一个网络包的最大长度，以太网中一般为1500字节

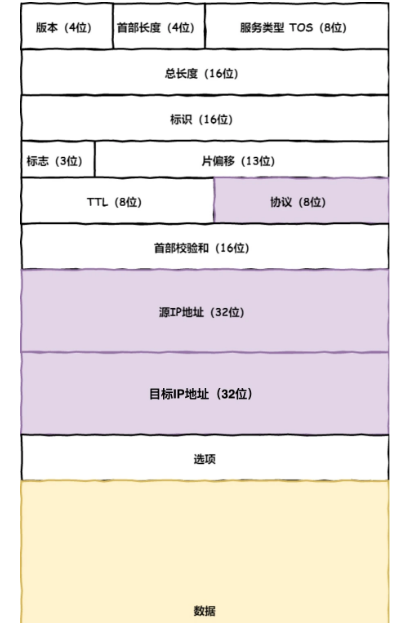
MSS（Maximum Segment Size最大报文段长度）除去IP和TCP头部一个网络包能容纳的TCP数据的最大长度

TCP报文生成后该发往哪？

1. 远程定位：IP

TCP模块在连接、收发、断开等阶段都需要委托IP将数据封装成网络包并发送给通信对象。

IP包头格式：



IP协议里要有源IP地址和目的IP地址

源IP地址：客户端IP地址

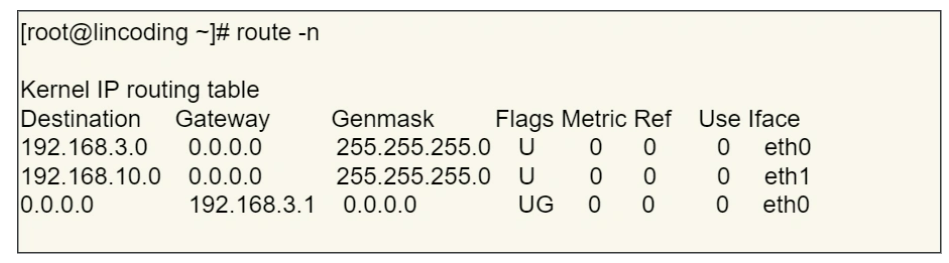
目标IP地址：通过DNS域名解析得到的web服务器IP

因为HTTP经过TCP传输，在IP包头的协议号为06（十六进制），表示协议为TCP

客户端有多个网卡，就会有多个IP地址，源地址应该选择哪个？

需要根据路由表规则判断哪个网卡作为源地址IP（使用哪个网卡来发送包）

Linux下 route -n命令查看路由表：



假设web服务器的目标地址为192.168.10.200

首先与192.168.3.0的子网掩码（genmask）进行与运算得到192.168.10.0与第一条不匹配

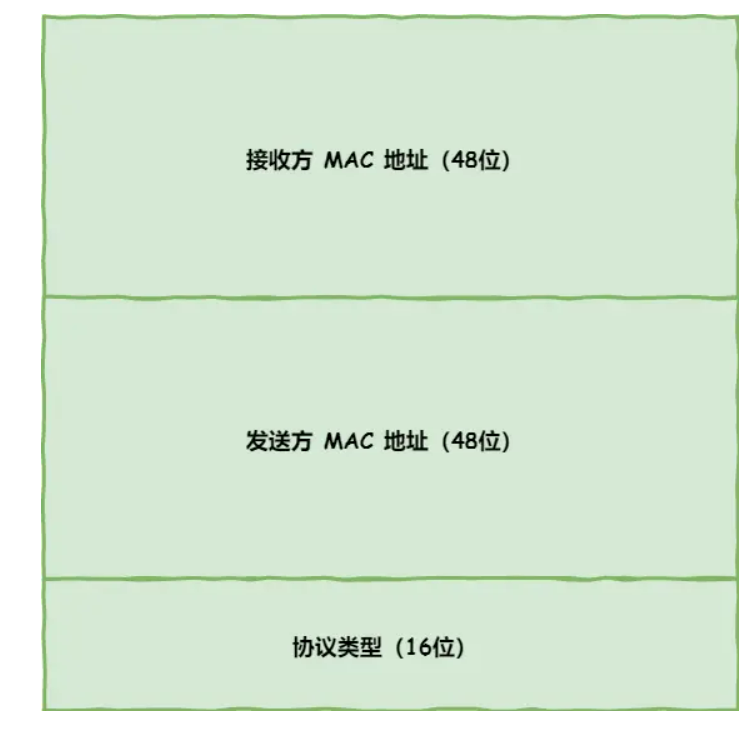
再与192.168.10.0的子网掩码与得到192.168.10.0，匹配即选择第二条为IP包头的源地址

假设web服务器的目标地址为10.100.20.100，与一二条都不匹配：第三条目标地址和子网掩码都是0.0.0.0，表示默认网关，如果所有条目都无法匹配会自动匹配这一行，后续把包发给路由器，Gateway是路由器的IP地址

生成IP报文后知道要去哪，下一站该怎么走？

1. 两点传输：MAC

MAC包头格式：



MAC包头里需要发送方和接收方的MAC地址用于两点间传输

一般在TCP/IP通信里，MAC包头协议只有：

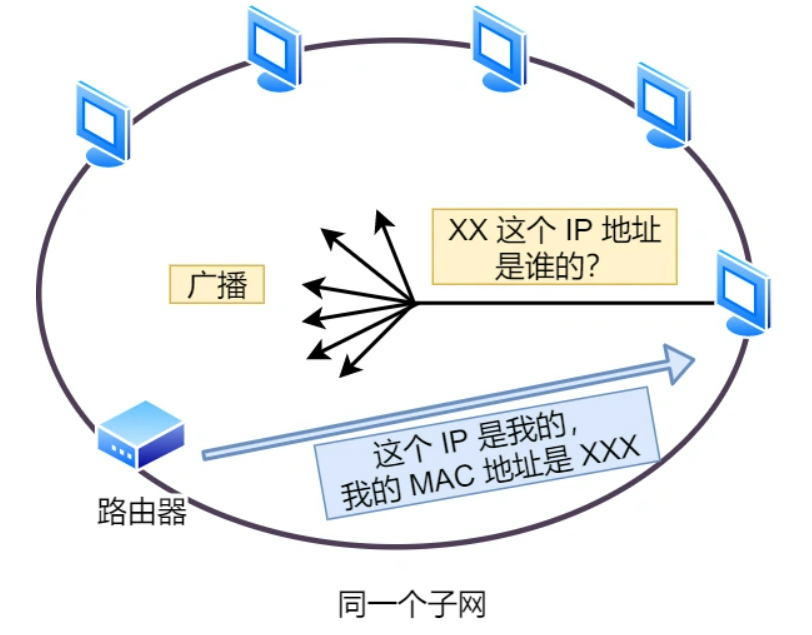
0800：IP协议

0806：ARP协议

MAC发送方和接收方MAC地址如何获得？

发送方MAC地址在网卡生产时写入ROM里

接收方MAC地址需要通过ARP协议获得：

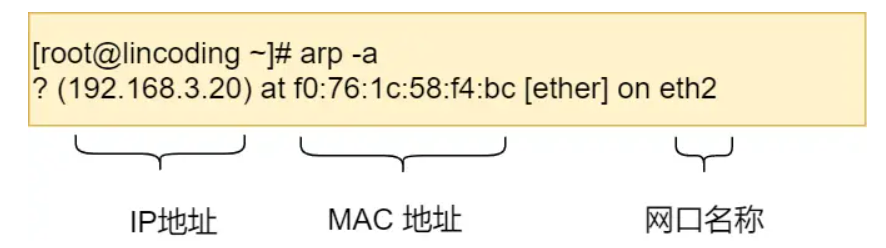


ARP协议在以太网中广播，并得到同一子网内目标的MAC地址

为了不每次都要广播，这个查询结果会被放到ARP缓存中，不过缓存时间就几分钟

发包时：先查ARP缓存，没有就发送ARP广播查询

Linux中通过arp -a查询ARP缓存



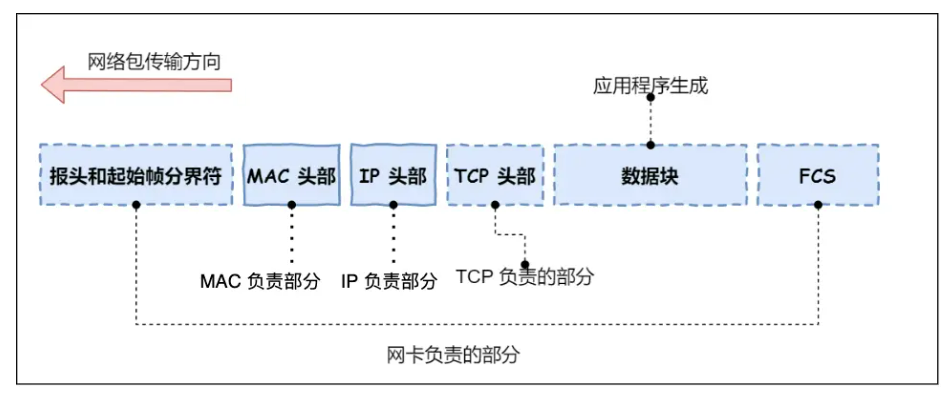
产生MAC报文后可以出门了

1. 出口：网卡

网络包是二进制数字信息，需要转化为电信号才能在网线上传输。

网卡负责这一操作，控制网卡需要网卡驱动程序

网卡驱动获取网络包后将其复制到网卡内的缓存中，在开头加上报头和起始帧分界符，在末尾加上用于检测错误的帧校验序列：



起始帧分界符表示包的起始位置

末尾FCS（帧校验序列）用来检测包传输过程中是否损坏

1. 交换机

交换机的设计是将网络包原样转发到目的地

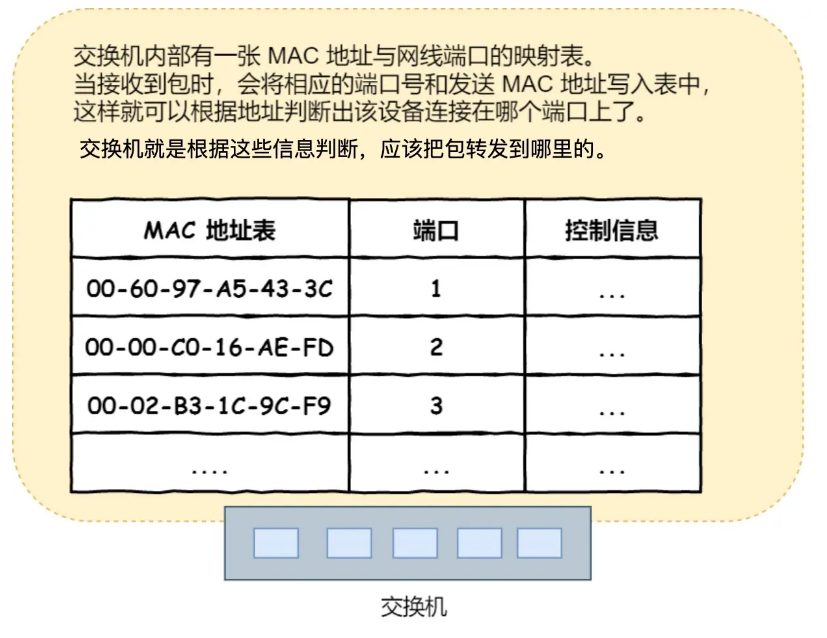
交换机的包接收操作：

电信号到达网线接口，交换机里的模块将其转换为数字信号并通过FCS校验，如果没问题放到缓冲区。

这一步和网卡相同但网卡有MAC地址，能判断包的接收方的MAC地址是不是自己，不是就丢弃，交换机端口没有MAC地址，将所有包存到缓冲区。

包存入缓冲区后查询这个包的接收方MAC地址是否在MAC地址表中。

交换机MAC地址表中包含：设备的MAC地址（发不发），该设备在交换机哪个端口上（发给谁）



交换机根据MAC地址将信号发送到相应端口

交换机MAC地址表中找不到指定MAC地址（可能是具有该地址的设备还没有向交换机发送过包，或这个设备一段时间没有工作被从地址列表中删除了）：这时交换机会将包转发到除了源端口的所有端口上，只有相应的接收者才会接收包，接收设备会做出响应，返回了响应包交换机就可以将它的地址写入MAC地址表。

如果接收方是个广播地址交换机也会将包发送到除了源端口外的所有端口：

广播地址：MAC地址FF:FF:FF:FF:FF:FF

IP地址：255.255.255.255

数据包通过交换机抵达路由器，即将离开子网

1. 路由器

网络包在路由器被转发到下一个路由器或目标设备，和交换机类似也通过查表判断转发目标。

路由器是基于IP设计的，各个端口都具有MAC地址和IP地址

交换机是基于以太网设计的，交换机的端口不具有MAC地址

路由器的基本原理：

路由器的端口有MAC地址，可以成为以太网的发送和接收方；具有IP地址，和计算机的网卡一样。转发包时，路由器端口接收发给自己的以太网包，路由表查询转发目标，再由相应端口将以太网包发送。

路由器的包接收操作：

电信号到达网线接口部分，路由器中的模块将电信号转换为数字信号，通过FCS校验。检测MAC头部的MAC地址，匹配就放到缓冲区，否则丢弃

查路由表确定输出端口：

完成接收操作后去掉包开头的MAC头。MAC头的作用是将包送达路由器，其中接收方的地址是路由器的端口的MAC地址，到达后就完成了任务。

接下来路由器会根据MAC头部后的IP头部的内容进行包的转发操作。查表过程同5

路由器的发送操作：

根据路由表的网关列判断对方地址：

如果网关是一个IP地址，这个IP地址就是要转发到的目标地址，还未抵达终点，需要路由器转发。

如果网关为空，IP头部中的接收方的IP地址就是要转发到的目标地址，抵达终点。

知道对方的IP地址后通过ARP协议根据IP地址查询MAC地址，将结果作为接收方的MAC地址。（路由器也有ARP缓存，在缓存中找不到才发送ARP查询请求）。

发送方的MAC地址填输出端口的MAC地址，网络包完成后将其转换成电信号通过端口发送。

发送的网络包通过交换机到达下一个路由器，层层转发。

网络包的传输过程中源IP和目标IP始终不变，一直变化的MAC地址，因为需要MAC地址在以太网内进行两个设备之间的传输。

1. 抵达后：服务器与客户端

数据包抵达后，服务端先解析并去掉MAC头部，查看是否匹配自己的MAC地址

解析IP头部，查看是否匹配自己的IP地址，根据IP头中的协议项确定自己的上层是TCP协议

解析TCP头，判断当前序列号是不是想要的，以及端口号

HTTP进程正在监听这个端口号，将包发给HTTP进程

HTTP进程解析请求报文，并将页面封装在HTTP响应报文中

一次加上TCP、IP、MAC头，源地址是服务器IP，目标地址是客户端IP

从网卡出去，由交换机转发到路由器，层层转发

最后到达客户端路由器，经过交换机转发到客户端

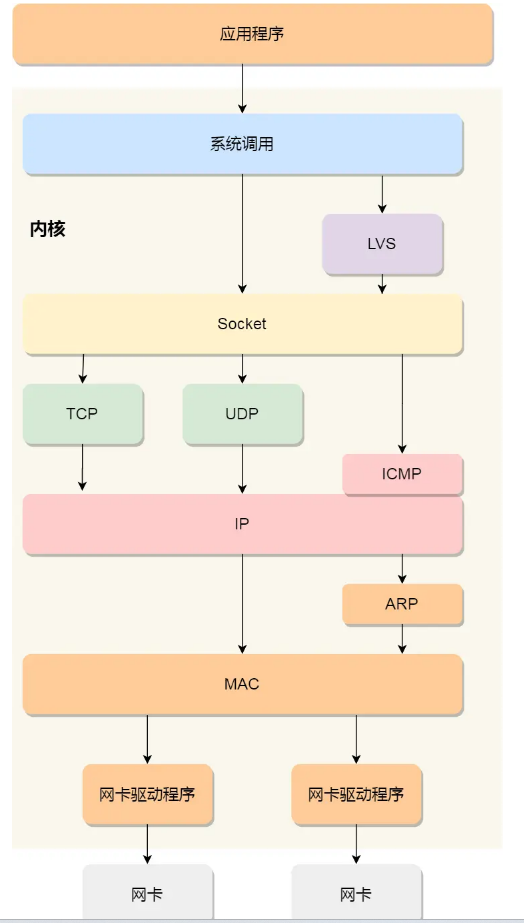
客户端解析

客户端要断开连接，经过TCP四次挥手断开连接。

1. Linux系统如何收发网络包

Linux网络模型：TCP/IP网络模型

Linux网络协议栈：



应用程序通过系统调用跟Socket层进行数据交互

Socket层下面就是传输层、网络层和网络接口层

1. Linux接收网络包的过程

网卡接收到一个网络包后，会通过DMA技术将网络包写入Ring Buffer中，接着网卡向CPU发起硬件中断，当CPU收到硬件中断请求后，根据中断表调用已经注册的中断处理函数。

（引入NAPI机制，使用混合中断和轮询的方式来接收网络包，不采用中断的方式读取数据（高性能网络下网络包多，会频繁触发中断，影响效率 ），而是采用中断唤醒数据接收的服务程序，然后使用poll的方法来轮询数据。）

硬件中断处理函数会：

先暂时屏蔽中断，表示知道Ring Buffer中有数据了，网卡下次收到数据包直接写入内存，不通知CPU，避免CPU不停中断。

发起软中断，回复屏蔽的中断

至此硬件中断处理函数的工作完成，主要耗时的工作交给软中断处理函数

软中断的处理：

内核中ksoftirqd线程负责中断处理，收到软中断后就轮询处理数据，会从Ring Buffer中取出一个数据帧sk\_buff，作为一个网络包交给网络

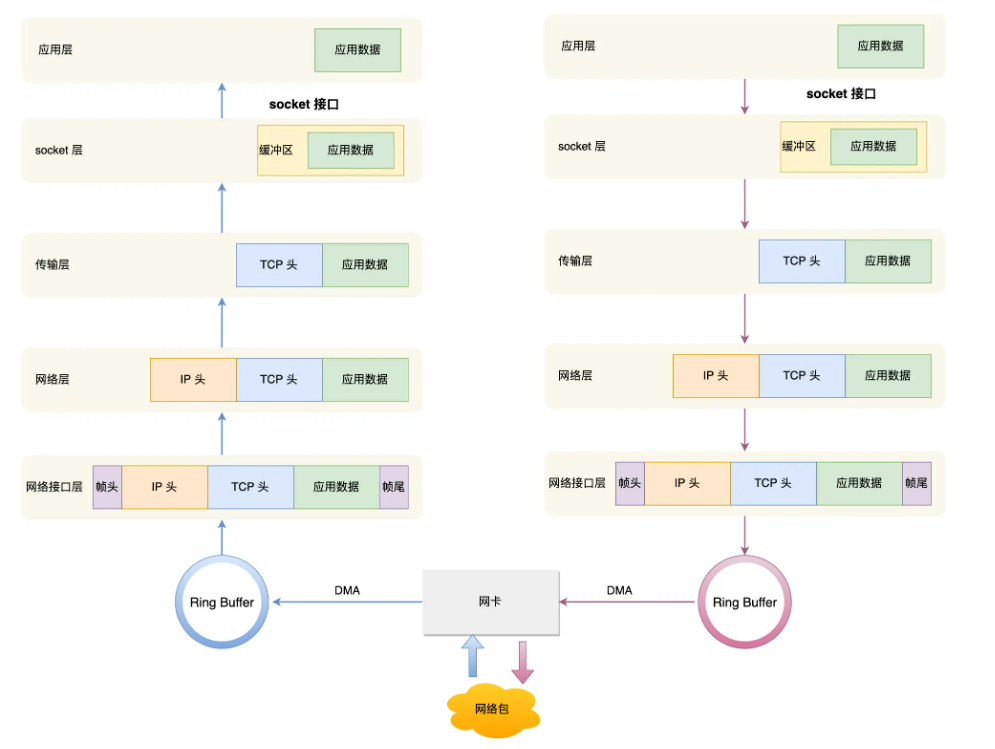
网络协议栈：

网络接口层：检测报文合法性，找出上层协议的类型，去掉帧头帧尾交给网络层

网络层：取出IP包，判断网络包下一步是交给上层还是转发，确定交给上层后确认上层协议类型，去掉IP头交给传输层

传输层：取出TCP头或UDP头，根据源IP、源端口、目的IP、目的端口作为标识，找到对应的Socket并把数据存入Socket接收缓冲区。

应用层程序调用Socket接口，将内核Socket接收缓冲区的数据拷贝到应用层的缓冲区，唤醒用户进程



1. Linux发送网络包流程

应用程序调用Socket发送数据包接口，这是系统调用，由用户态陷入内核态的Socket层，内核申请一个sk\_buff内存，将用户待发送的数据拷贝到sk\_buff内存并将其加入发送缓冲区。

网络协议栈从Socket发送缓冲区中取出sk\_buff按照TCP/IP协议栈从上到下处理

如果是TCP协议，先拷贝一个新的sk\_buff副本防止丢失重传（sk\_buff到达网卡发送完成时会被释放，TCP在每次发送时传递的是sk\_buff的拷贝，等收到ACK再真正删除）

对sk\_buff填充TCP头，在应用层叫data，在TCP层叫segment，在IP层叫packet，在数据链路层叫frame，但只用sk\_buff一个结构体通过调整data指针（接收报文时增加skb->data来剥离协议首部，发送时在创建时预留足够空间，减少来增加协议首部）来描述所有的网络包（如果用多个结构体在层之间的传递会发生多次拷贝降低效率）

交给网络层：选取路由（确认下一跳IP）、填充IP头、对超过MTU的数据包分片

网络接口层通过ARP协议获得下一跳MAC地址，填充头尾

触发软中断，网卡驱动程序将sk\_buff挂到Ring Buffer