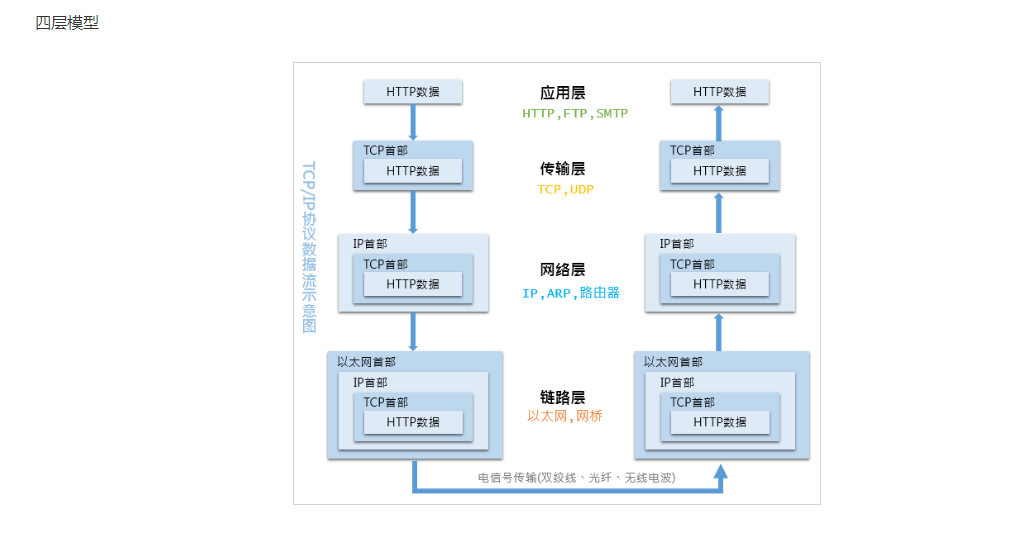
# TCP/IP协议

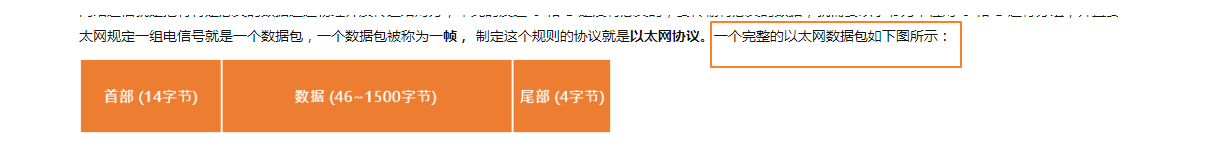


## 链路层

作用:将电信号0和1进行分组,并形成成有特定意义的数据帧,然后以广播的形式通过物流介质传送给接收方.

协议:以太网协议

一个完整的以太网数据包如下图,mac地址在首部存储



如何表示接收方呢?

通mac地址.mac的网卡的唯一标识.mac地址不会重复.以太网就会以广播 形式,将广播给子网内的所有主机.主机会匹配数据包中的mac地址,如果和自己的mac地址相同就接收并处理.不同就丢弃.

## 网络层

网络层的主要工作是**定义网络地址，区分网段，子网内MAC寻址，对于不同子网的数据包进行路由。**

协议:ip协议,arp协议,路由协议

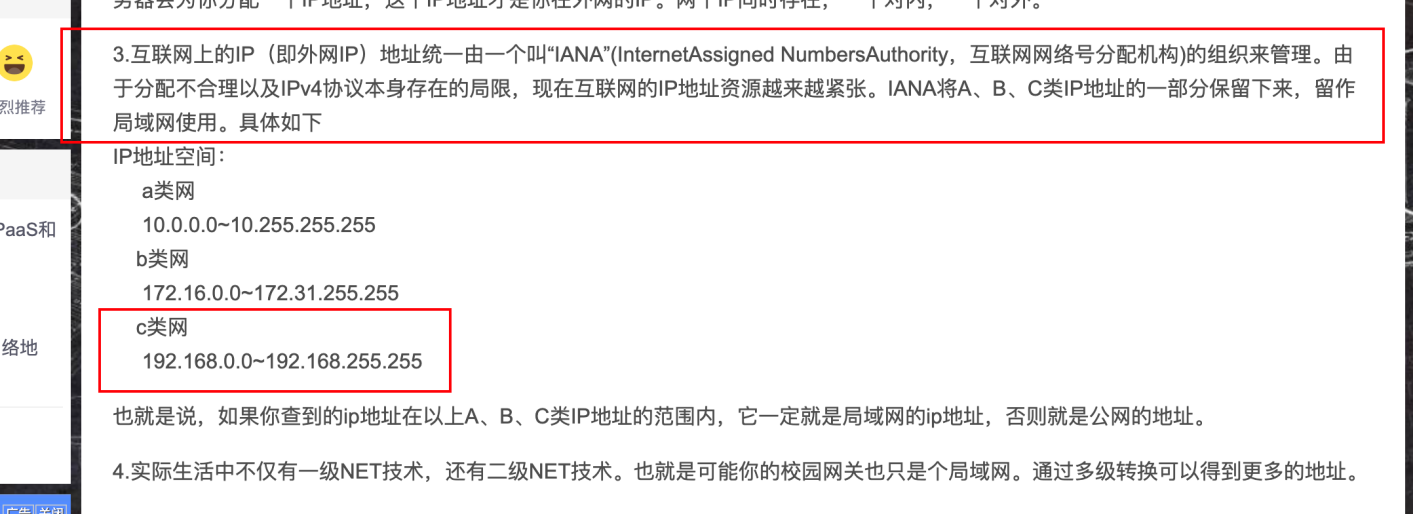
ip协议: IP地址分为五类，A类保留给政府机构，B类分配给中等规模的公司，C类分配给任何需要的人，D类用于组播，E类用于实验

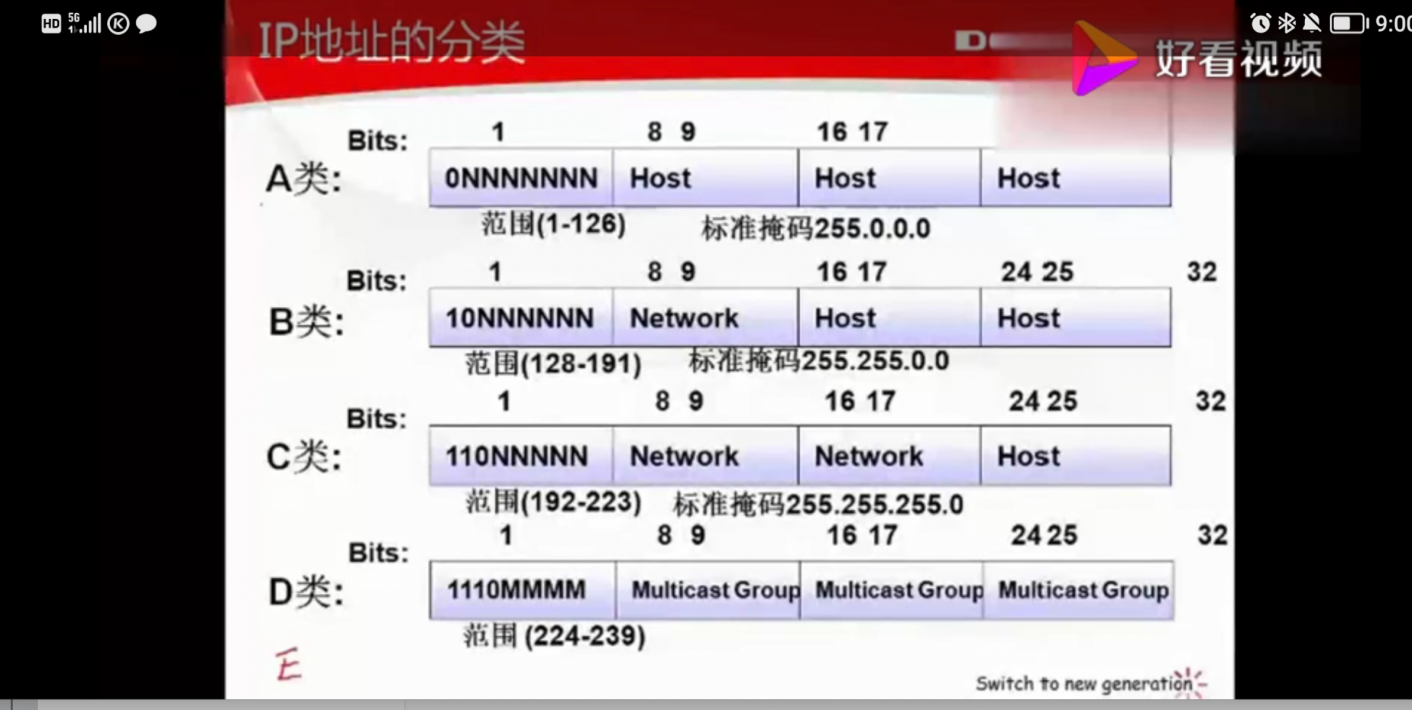
子网掩码的作用:因此我们只要通过子网掩码对两个IP地址进行AND运算后就能够判断双方是否在同一个子网了。如果与运算后的网络地址(主机位全0的，就是网络地址)是一样的，那么这两个ip就是在同一个网络里，可以直接通信。网络地址一样的话，就在同一个网络里。

主机位全1的话，就是广播地址。往广播地址发送数据包，这个网络的所有主机都能收到数据包。（就像往在村里的广播台说一句话，全村人都能听到这句话）。

子网掩码作用就是判断两个ip是否在同一个网络里。

内网ip和外网ip的区分。一般在192.168.0.0～192.168.255.255之间的是内网ip





ARP协议: 是根据**IP地址**获取**MAC地址**的一个网络层协议 .

mac地址是网卡制造商给网卡设定的唯一标识。mac地址处于网络的第一层，数据链路层。数据链路层协议可以使数据从一个节点传递到相同链路的另一个节点上（通过MAC地址），而网络层协议使数据可以从一个网络传递到另一个网络上（ARP根据目的IP地址，找到中间节点的MAC地址，通过中间节点传送，从而最终到达目的网络）

路由协议: 如果发送方和接收方不在同一个子网内,就会用路由协议,将数据传输到接收方所在的子网内,然后通过ARP协议,将数据的ip地址转成mac地址,已广播的形式传给所以的主机.

通过ARP协议的工作原理可以发现，ARP的MAC寻址还是局限在同一个子网中，因此网络层引入了路由协议，首先通过IP协议来判断两台主机是否在同一个子网中，如果在同一个子网，就通过ARP协议查询对应的MAC地址，然后以广播的形式向该子网内的主机发送数据包；如果不在同一个子网，以太网会将该数据包转发给本子网的网关进行路由。网关是互联网上子网与子网之间的桥梁，所以网关会进行多次转发，最终将该数据包转发到目标IP所在的子网中，然后再通过ARP获取目标机MAC，最终也是通过广播形式将数据包发送给接收方。

## 传输层:

传输层的主要工作是**定义端口，标识应用程序身份，实现端口到端口的通信，TCP协议可以保证数据传输的可靠性**。

协议:tcp协议,udp协议.tcp协议简单来说就是有确认机制的udp协议.

tcp协议三次握手,确认可靠之后才开始传输数据.

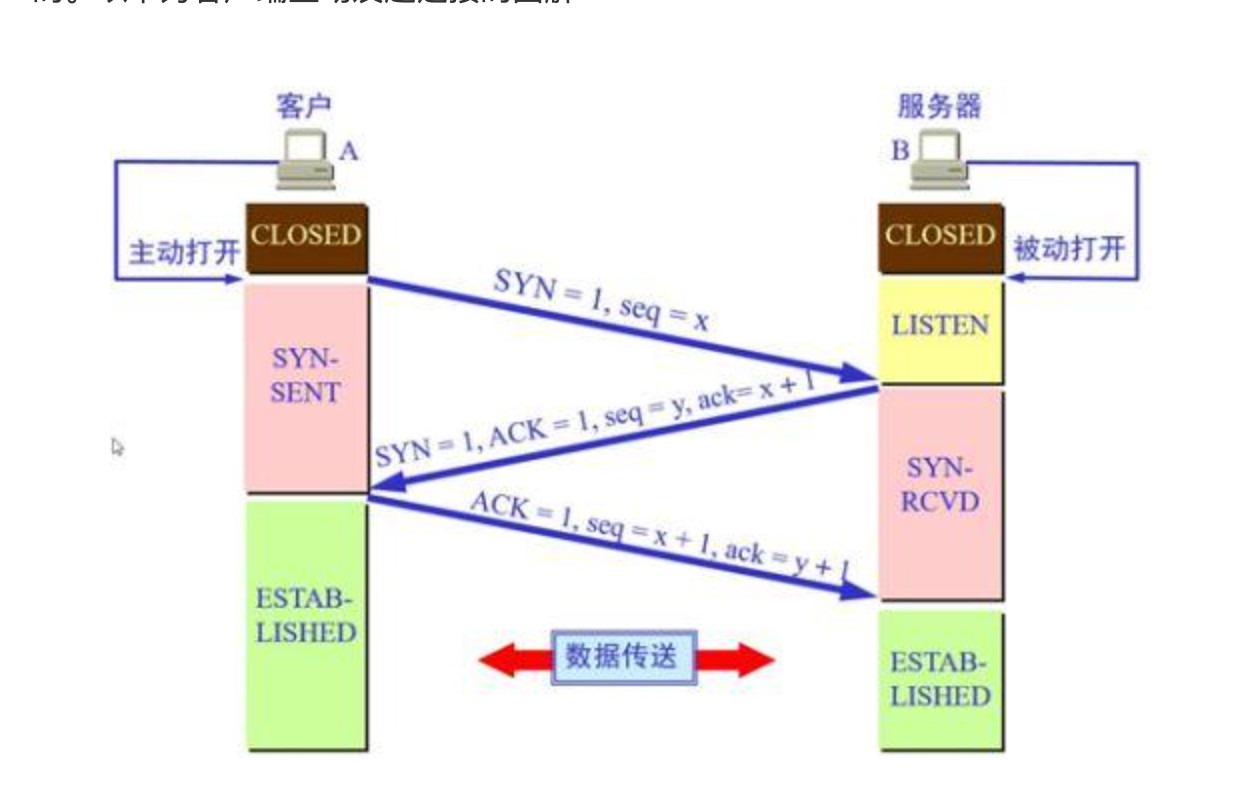
tcp3次握手和4次挥手详解

syn报文:代表申请建立链接

ack报文，代表确认收到消息

fin报文，代表断开连接。

3次握手

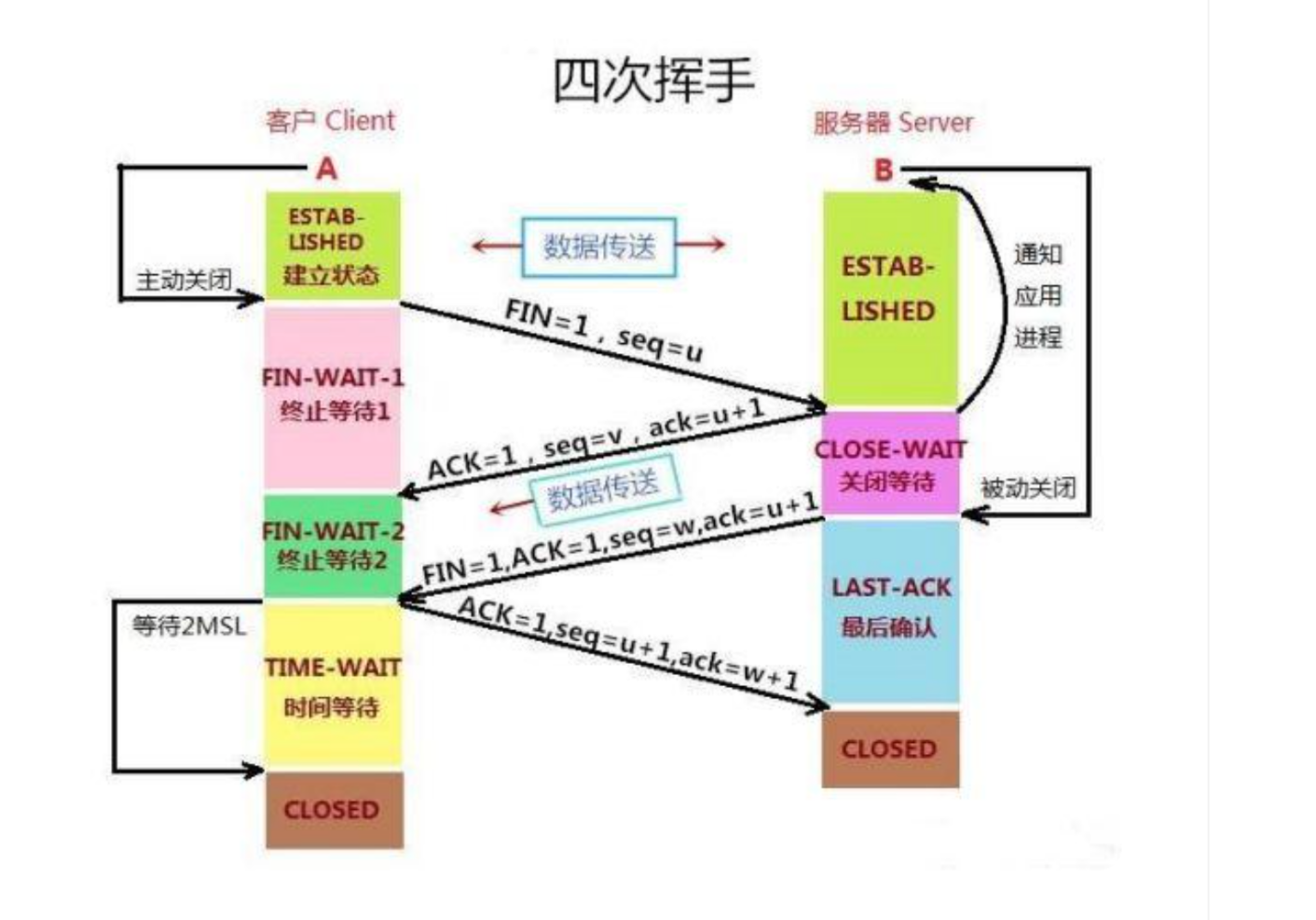


第一次握手，客户端往服务端发送一个申请建立链接的syn报文，代表客户端申请建立链接。

第二次莴笋，服务器收到后报文后，会返回ack报文和syn报文给客户端。ack报文代表服务器确认和客户端的链接。syn报文代表服务端申请和客户端建立链接

第三次，客户端也返回一个确认ack报文给服务段，代表客户端确认了服务器的链接请求。到此3此握手完成，tcp链接建立成功。

4次挥手



第一次:客户端发送fin报文，代表客户端申请断开连接。

第二次:服务器返回一个ack报文，代表服务端确认了这次断开连接的请求。并且此时服务器开始进入了断开链接的状态。

第三次:服务器准备好断开后，会发起一个fin报文给客户端，代表服务端申请断开连接。此时服务器进去最后的关闭状态。

第四次，客户端返回一个ack报文，代表客户端确认了这次断开连接的请求。

此时客户端断开了链接。服务端收到确认报文后，也断开了链接

TCP是一种面向广域网的通信协议，目的是在跨越多个网络通信时，为两个通信端点之间提供一条具有下列特点的通信方式： [1]

（1）基于流的方式；

（2）面向连接；

（3）可靠通信方式；

（4）在网络状况不佳的时候尽量降低系统由于重传带来的带宽开销；

（5）通信连接维护是面向通信的两个端点的，而不考虑中间网段和节点。

为满足TCP协议的这些特点，TCP协议做了如下的规定： [10]

①数据分片：在发送端对用户数据进行分片，在接收端进行重组，由TCP确定分片的大小并控制分片和重组；

②到达确认：接收端接收到分片数据时，根据分片数据序号向发送端发送一个确认；

③超时重发：发送方在发送分片时启动超时定时器，如果在定时器超时之后没有收到相应的确认，重发分片；

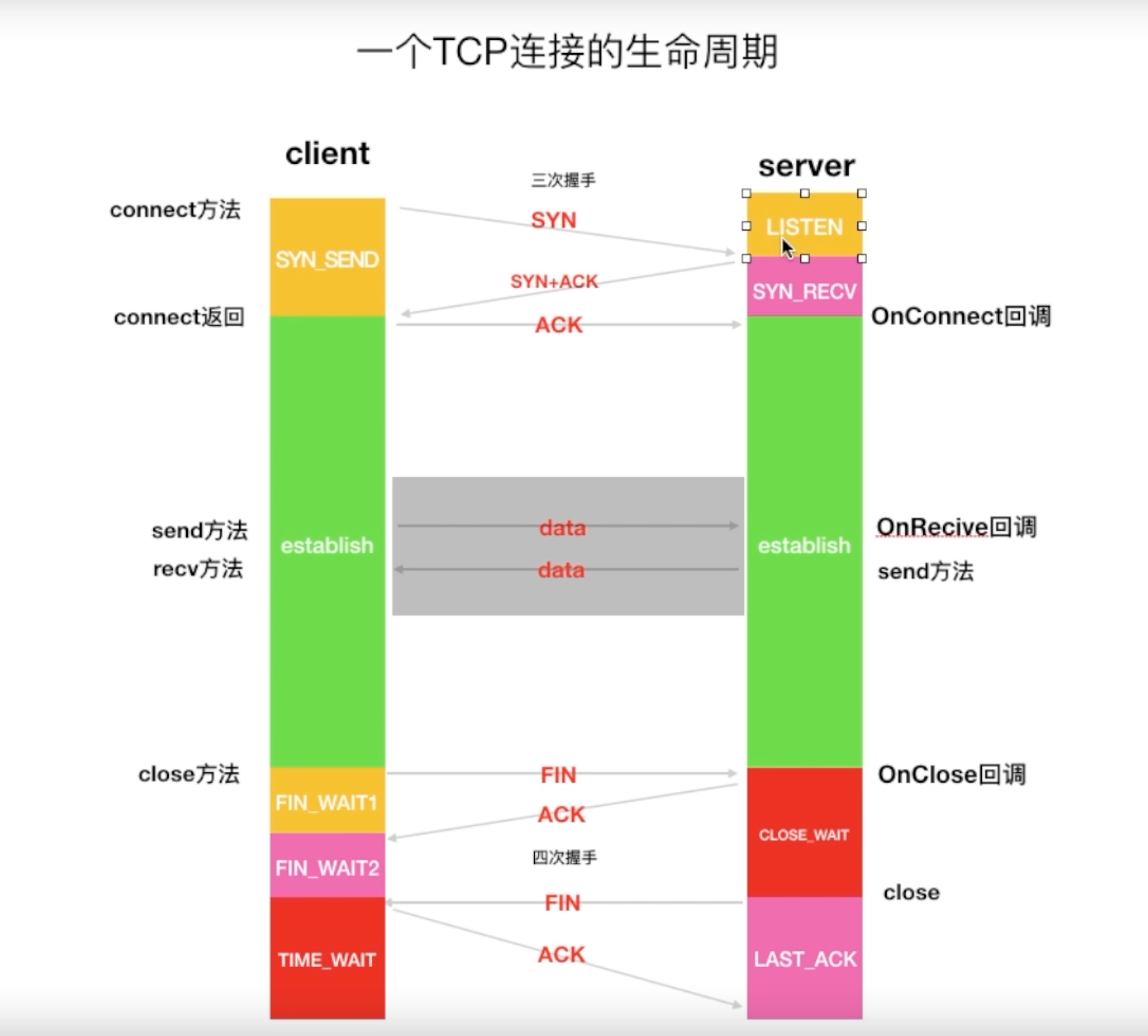
④滑动窗口：TCP连接每一方的接收缓冲空间大小都固定，接收端只允许另一端发送接收端缓冲区所能接纳的数据，TCP在滑动窗口的基础上提供流量控制，防止较快主机致使较慢主机的缓冲区溢出；

⑤失序处理：作为IP数据报来传输的TCP分片到达时可能会失序，TCP将对收到的数据进行重新排序，将收到的数据以正确的顺序交给应用层；

⑥重复处理：作为IP数据报来传输的TCP分片会发生重复，TCP的接收端必须丢弃重复的数据；

⑦数据校验：TCP将保持它首部和数据的检验和，这是一个端到端的检验和，目的是检测数据在传输过程中的任何变化。如果收到分片的检验和有差错，TCP将丢弃这个分片，并不确认收到此报文段导致对端超时并重发

整体的生命流程



在TCP层，有个FLAGS字段，这个字段有以下几个标识：SYN, FIN, ACK, PSH, RST, URG.

其中，对于我们日常的分析有用的就是前面的五个字段。

 它们的含义是：

SYN表示建立连接，

FIN表示关闭连接，

ACK表示响应，

PSH表示有 DATA数据传输，

RST表示连接重置。

### **RST**

　　TCP的异常终止是相对于正常释放TCP连接的过程而言的，我们都知道，TCP连接的建立是通过三次握手完成的，而TCP正常释放连接是通过四次挥手来完成，但是有些情况下，TCP在交互的过程中会出现一些意想不到的情况，导致TCP无法按照正常的四次挥手来释放连接。如果此时不通过其他的方式来释放TCP连接的话，这个TCP连接将会一直存在，占用系统的部分资源。在这种情况下，我们就需要有一种能够释放TCP连接的机制，这种机制就是TCP的reset报文。

场景：

* 客户端尝试与服务器未对外提供服务的端口建立TCP连接，服务器将会直接向客户端发送reset报文。
* 客户端建立连接时，接收到服务器的syn超时，客户端发送reset报文。
* 客户端和服务器的某一方在交互的过程中发生异常（如程序崩溃等），该方系统将向对端发送reset报文，告之对方释放相关的TCP连接。
* 接收端收到TCP报文，但是发现该TCP的报文，并不在其已建立的TCP连接列表内，则其直接向对端发送reset报文。
* 在交互的双方中的某一方长期未收到来自对方的确认报文，则其在超出一定的重传次数或时间后，会主动向对端发送reset报文释放该TCP连接。
* 有些应用开发者在设计应用系统时，会利用reset报文快速释放已经完成数据交互的TCP连接，以提高业务交互的效率。

利用：

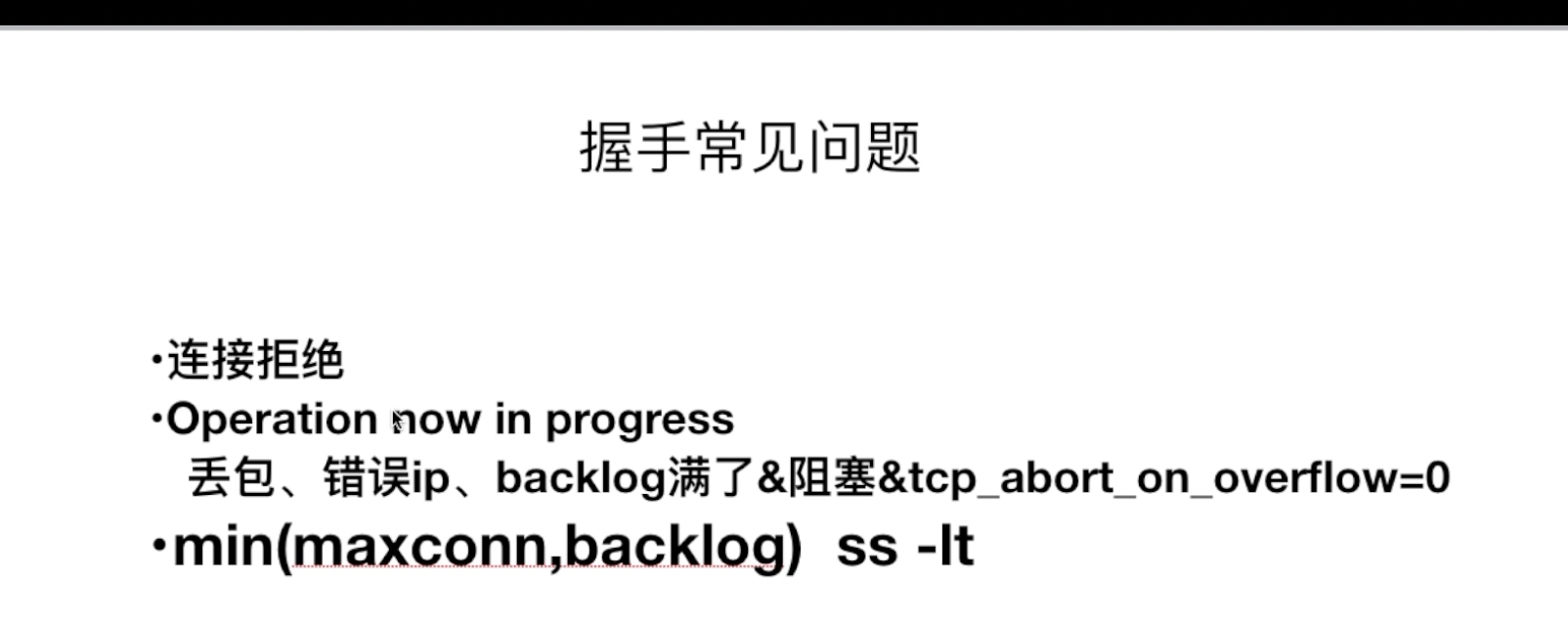
* 安全设备利用reset报文阻断异常连接：安全设备（如防火墙、入侵检测系统等）在发现某些可疑的TCP连接时，会构造交互双方的reset报文发给对端，让对端释放该TCP连接。比如入侵检测检测到黑客攻击的TCP连接，其构造成被攻击端给黑客主机发送reset报文，让黑客主机释放攻击连接。
* 利用reset报文实施攻击：安全设备可以利用reset报文达到安全防护的效果，黑客和攻击者也可以利用reset报文实现对某些主机的入侵和攻击，最常见的就是TCP会话劫持攻击。

### **SYN Flood**

　　SYN Flood是指利用TCP协议缺陷，发送大量伪造的TCP连接请求（SYN包），从而使得被攻击方资源耗尽（CPU满负荷或内存不足）的攻击方式。

　　解决方法：

* 增加积压队列：在内核里有个队列用来存放还没有确认ACK的客户端请求，当等待的请求数大于tcp\_max\_syn\_backlog时，后面的会被丢弃。所以，适当增大这个值，可以在压力大的时候提高握手的成功率。推荐大于1024。
* 回收最早的半开TCP连接：一旦积压已被填补，另一个缓解策略就是覆盖最早的半开式连接。这种策略要求合法连接可以在比可以填充恶意SYN数据包的积压时间更短的时间内完全建立。当攻击量增加时，或者如果积压量太小而不实际，这种特定的防御就会失败。
* 减少ack重试：这个是三次握手中，服务器回应ACK给客户端里，重试的次数。默认是5。显然攻击者是不会完成整个三次握手的，因此服务器在发出的ACK包在没有回应的情况下，会重试发送。当发送者是伪造IP时，服务器的ACK回应自然是无效的。为了防止服务器做这种无用功，可以把tcp\_synack\_retries设置为0或者1。因为对于正常的客户端，如果它接收不到服务器回应的ACK包，它会再次发送SYN包，客户端还是能正常连接的，只是可能在某些情况下建立连接的速度变慢了一点。
* SYN cookie：当半连接的请求数量超过了 tcp\_max\_syn\_backlog 时，内核就会启用SYN cookie机制，不再把半连接请求放到队列里，而是用SYN cookie来检验。SYN cookie是非常巧妙地利用了TCP规范来绕过了TCP连接建立过程的验证过程，从而让服务器的负载可以大大降低。在三次握手中，当服务器回应（SYN + ACK）包后，客户端要回应一个n + 1的ACK到服务器，其中n是服务器自己指定的。当启用tcp\_syncookies时，backlog满了后，linux内核生成一个特定的n值，并不把客户端的连接放到半连接的队列里（即没有存储任何关于这个连接的信息，不浪费内存）。当客户端提交第三次握手的ACK包时，linux内核取出n值，进行校验，如果通过，则认为这个是一个合法的连接。（backlog表示全连接队列长度）



链路层定义了主机的身份，即MAC地址， 而网络层定义了IP地址，明确了主机所在的网段，有了这两个地址，数据包就从可以从一个主机发送到另一台主机。 但实际上数据包是从一个主机的某个应用程序发出，然后由对方主机的应用程序接收。而每台电脑都有可能同时运行着很多个应用程序，所以当数据包被发送到主机上以后，是无法确定哪个应用程序要接收这个包。

## 应用层

所以应用层的主要工作就是定义数据格式并按照对应的格式解读数据。

协议:http,ftp(文件传输协议),smtp(邮件协议)

理论上讲，有了以上三层协议的支持，数据已经可以从一个主机上的应用程序传输到另一台主机的应用程序了，但此时传过来的数据是字节流(二进制)，不能很好的被程序识别，操作性差。因此，应用层定义了各种各样的协议来规范数据格式，常见的有 HTTP、FTP、SMTP 等，HTTP 是一种比较常用的应用层协议，主要用于B/S架构之间的数据通信 .

在 Resquest Headers 中，Accept 表示客户端期望接收的数据格式，而 ContentType 则表示客户端发送的数据格式；在 Response Headers 中，ContentType 表示服务端响应的数据格式，这里定义的格式，一般是和 Resquest Headers 中 Accept 定义的格式是一致的。

有了这个规范以后，服务端收到请求以后，就能正确的解析客户端发来的数据，当请求处理完以后，再按照客户端要求的格式返回，客户端收到结果后，按照服务端返回的格式进行解析。

## 5、全流程

首先我们梳理一下每层模型的职责：

* **链路层**：对0和1进行分组，定义数据帧，确认主机的物理地址，传输数据；
* **网络层**：定义IP地址，确认主机所在的网络位置，并通过IP进行MAC寻址，对外网数据包进行路由转发；
* **传输层**：定义端口，确认主机上应用程序的身份，并将数据包交给对应的应用程序；
* **应用层**：定义数据格式，并按照对应的格式解读数据。

然后再把每层模型的职责串联起来，用一句通俗易懂的话讲就是：

当你输入一个网址并按下回车键的时候，首先，应用层协议对该请求包做了格式定义；紧接着传输层协议加上了双方的端口号，确认了双方通信的应用程序；然后网络协议加上了双方的IP地址，确认了双方的网络位置；最后链路层协议加上了双方的MAC地址，确认了双方的物理位置，同时将数据进行分组，形成数据帧，采用广播方式，通过传输介质发送给对方主机。而对于不同网段，该数据包首先会转发给网关路由器，经过多次转发后，最终被发送到目标主机。目标机接收到数据包后，采用对应的协议，对帧数据进行组装，然后再通过一层一层的协议进行解析，最终被应用层的协议解析并交给服务器处理。

## socket通信

什么是socket？

socket其实是tcp/ip协议中的接口，比如connect，send，read，write等接口的统一称呼。

socket通信其实就是tcp通信的原理。客户端需要和服务端通信时，客户端建立一个socket对象，通过socket对象的connect接口，和服务端发起一个链接请求。服务端一直在监听某个端口，当某个端口收到了客户端的请求时，服务器的socket对象会调用read接口，读取客户端发送的数据。如下所示

。

