网络编程中的两个主要问题

- 如何准确定位网络上一台或多台主机，定位主机上特定的应用；
- 如何可靠高效的传输信息

通信的两个要素

- IP和端口号
- 通信协议（TCP/IP参考模型）
- IP：主机地址
- 端口号：标识正在计算机上运行的进程。
- 网络套接字：端口号和IP地址组合得到网络套接字。
- **IP地址：**

  IP地址是计算机之间进行网络通信所必须的，它标识了不同计算机的身份，IP地址=网络地址+主机地址(又称网络号+主机号)；我们去访问另一台计算机时，需要先找到另一台计算机所在的网络号，再获取该计算机在此网络中的主机号，才能成功通信。在Java中使用InetAddress类代表IP。IP分类：IPv4和IPv6；万维网和局域网。

- **域名：**

  类似于www.baidu(域名).com。如果我们通过IP地址去访问服务器，非常难以记忆，所以引入了域名的概念。

- **DNS**：

  域名解析服务器，存储了网络中域名和对应IP地址的服务器。他会通过域名找到对应的IP地址去访问对应的网络服务器。

- **网络掩码：**

  在一台计算机去访问另一台计算机的时候，我们需要查看它们是否在同一网络地址下，这个时候就需要网络掩码，与IP地址相与得到网络地址，网络地址相同的两台计算机可以直接访问。

- **URL**（统一资源定位符）

  对应着互联网的某一资源地址。网站地址。浏览器会通过超文本传输协议（http），把万维网（www）的服务器上的站点网页代码找出来，并翻译成网页。

讲一下浏览器输入URL返回页面过程

- 首先是解析域名，找到主机IP
- 浏览器利用IP直接与网站主机通信，三次握手，建立TCP连接。浏览器会以一个随机端口向服务器的web程序80端口发起TCP的连接
- 建立TCP连接后，浏览器向主机发起一个HTTP请求
- 服务器响应请求，返回响应数据
- 浏览器解析响应内容，进行渲染，呈现给用户

HTTP和HTTPS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你的理解是怎么保证他的安全性的

HTTPS使用**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。

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。

混合加密的方式实现信息的**机密性**，交互信息无法被窃取，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，无法篡改通信内容，解决了篡改的风险。
将服务器公钥放入到**数字证书**中，证明淘宝是真的淘宝网，解决了冒充的风险。

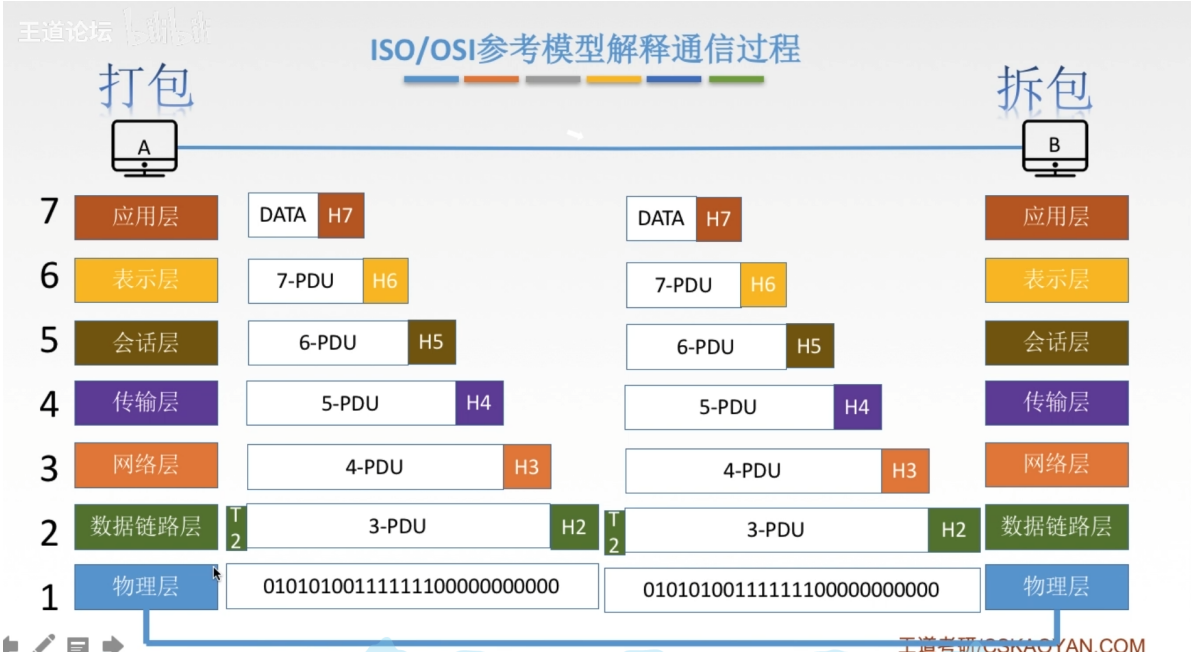
OSI分层

开放系统互连的参考模型（OSI）：物理层、数据链路层、网络层、传输层、会话层、表示层、应用层

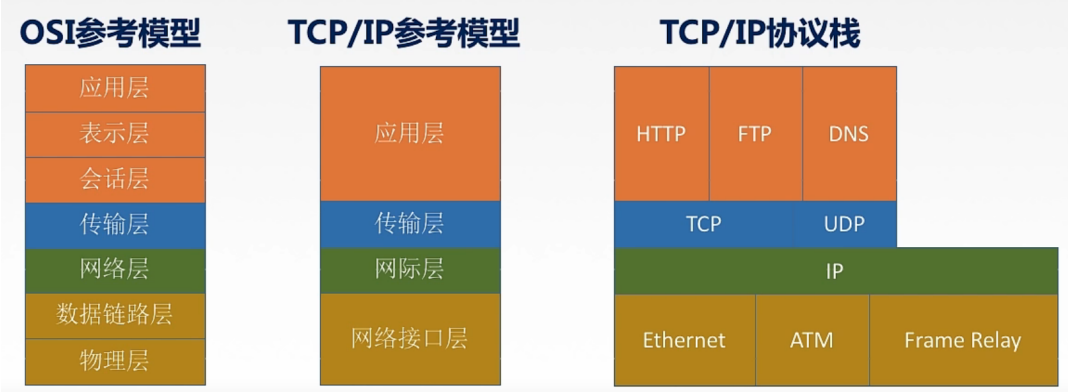
1.都分层

2.基于独立的协议栈的概念

3.可以实现异构网络互联



TCP/IP参考模型



	ISO/OSI参考模型	TCP/IP模型
网络层	无连接+面向连接	无连接
传输层	面向连接	无连接+面向连接

物理层、数据链路层、网络层、传输层、应用层

- 应用层：为应用程序提供交互服务。在互联网中的应用层协议很多，如域名系统DNS、HTTP协议、SMTP协议等.
- 传输层：负责向两台主机进程之间的通信提供数据传输服务。传输层的协议主要有传输控制协议TCP和用户数据协议UDP。
  网络层：选择合适的路由和交换结点，确保数据及时传送。主要包括IP协议。
- 数据链路层：在两个相邻节点之间传送数据时，数据链路层将网络层交下来的IP数据报组装成帧，在两个相邻节点间的链路上传送帧。
- 物理层:实现相邻节点间比特流的透明传输，尽可能屏蔽传输介质和物理设备的差异。

为什么需要ARP 协议

arp协议的主要功能是将IP地址解析为物理地址
当主机A想要向本局域网中的主机B发送数据报时，就会先在它的ARP高速缓存中查看有没有主机B的IP地址。如果有，就可以查到他对应的硬件地址，再将他的硬件地址写入MAC帧，然后通过局域网将这个MAC帧发往硬件地址。如果ARP上没有主机B的Ip地址，ARP进程就会在本局域网上广播发送一个ARP请求分组，收到ARP响应分组后，将得到的IP地址到硬件地址的映射写入ARP高速缓存。

什么是SYN洪范攻击？如何解决？

在TCP连接的半连接状态中，服务器处于等待客户端返回确认号的状态，此时如果收到客户端的确认号则连接建立成功。如果没有收到则会一直重发请求直到成功。SYN攻击者会在短时间内伪造大量不存在的IP地址，向服务器不断地发送syn包，服务器回复确认包，并等待客户的确认，由于源地址是不存在的，服务器需要不断的重发直至超时，这些伪造的SYN包将长时间占用未连接队列，正常的SYN请求被丢弃，目标系统运行缓慢，严重者引起网络堵塞甚至系统瘫痪。

解决：

- 通过防火墙、路由器等过滤网关防护；
- 通过加固TCP/IP协议栈防范，比如增加最大半连接数，缩短超时时间。
- 使用SYN cookies技术，这个技术是对服务器端的三次握手做一些修改，是用来专门SYN防范洪范攻击的一种手段。

TCP协议

TCP叫传输控制协议（打电话）

- TCP是面向连接的传输层协议
- 每条连接都是点对点的
- TCP提供可靠交付的服务，无差错、不丢失、按序到达。可靠有序，不丢不重复
- TCP提供全双工通信（发送缓冲和接受缓存）
- TCP面向字节流 把应用程序交下来的数据仅仅看成一连串无结构的字节流

TCP协议如何保证可靠性？

一 检验和：

- 在报文的发送端，会根据报文的首部或数据来计算一个检验和；
- 然后接收端接受到相应报文，也会对报文的首部或数据进行一次检验和计算；
- 如果接收端算出来的检验和和发送端发送的不一样，那么接收端认为报文在传输过程中出了错，就会丢掉该报文。

二 序列号和确认应答：

- 发送端的数据到达主机时，接收端会返回一个收到数据的的消息。这个消息就叫做确认应答（ACK）。
- 当发送端发送数据后，会等待接收端的确认应答。如果收到确认应答，说明数据已经成功到达。反之，则数据丢失的可能性很大。
- 发送端一定时间内没有等到确认应答，发送端就认为数据已经丢失了，就可以重发数据。因此，即使产生了丢包，也能保证数据到达对端。实现可靠传输。
- ACK报文中带有对应的确认序列号，接收方能够将收到的数据根据序列号排序，并且去掉重复序列号的数据。序列号能够告诉发送方，收到了哪些数据，下一次的数据从哪里开始发。

三 超时重传

是指发送出去的数据包到接受确认应答包之间的时间，如果超过了这个时间会被认为是丢包了，需要重传。最大超时时间是动态的。

四 滑动窗口

在进行数据传输的时候，如果要传送的数据比较大，那么这时候需要把数据拆分成多个数据包进行发送。TCP协议需要对数据进行确认后才可以发送下一个数据包，这样会在等待确认应答的时候浪费时间。为了避免这种情况引入了滑动窗口的概念。窗口大小指的是不需要等待而可以继续发送数据包的最大值。

滑动窗口分为两个部分，一部分是已经发送但未被确认的分组，一部分是等待发送的分组。随着已发送的分组不断被确认，等待发送的分组也不断地被发送。整个窗口就会向后移动，让没有轮到的分组进入窗口。

五 拥塞控制

对网络中某个资源的需求超过了这个资源能够提供的可用部分，比如点缓存的容量太小，链路的容量不足等，会导致网络的性能变差，这就是拥塞现象。

网络中发生拥塞时，发送的数据包会因为延迟无法到达接收端，基于超时重传机制，发送端会重新发送数据包，那么本来拥塞的网络会更加拥塞，由此导致恶性循环。

TCP使用了四个算法来实现拥塞控制：

在拥塞控制中，发送方维持了一个叫做拥塞窗口的门限值的状态变量，他会根据网络的拥塞程度动态变化（比如说到达了慢启动门限值就会开始拥塞避免算法，发生丢包情况就会触发乘法减小算法。）

1. 慢开始：

一开始发送数据时，从小到大的逐渐增加拥塞窗口的大小。刚开始收到ACK时将拥塞窗口大小加一，之后每轮次发送窗口都增加一倍，呈指数增长。慢开始有一个慢启动门限值，达到这个门限值时就会触发拥塞避免算法。

2. 拥塞避免：

拥塞避免算法中每收到一个接收方的ACK时窗口大小都是只加一，呈线性上升。

3. 快速重传：

如果接受方发现丢包时，会发送三次前一个包的确认应答，发送端就会快速重传数据，不用等待超时。

4. 快速恢复：

快速恢复就是为了配合快速重传，当发送方收到三个重复确认时，就会执行“乘法减小”算法，把慢启动门限值减少到当前拥塞窗口大小的一半，接着执行拥塞避免算法。（不执行慢开始的原因是因为发送方能收到三个重复确认说明网络状况还可以）

六 流量控制

抑制发送方发送数据的速率，使得接收方来得及接受。利用滑动窗口可以很好的实现流量控制。

拥塞控制和流量控制的区别?

流量控制是为了抑制发送端发送数据的熟虑使得接收端来得及接受，是**点对点**通信流量的控制；

而拥塞控制它是防止过多的数据注入到网络中而导致网络中的路由器或者链路过载。拥塞控制是一个**全局性**的过程，涉及到降低网络传输质量的所有因素。

举例说明：

比如说有一个网络的链路传输速率为每秒1000G比特，那么一个巨型计算机要给一个主机以每秒1G比特的速率传送一个文件，那么先让网络本身的带宽是足够大的，我们不用担心网络拥塞的问题。但是流量控制是必须的，巨型计算机必须经常停下来以便于主机来得及接收数据。

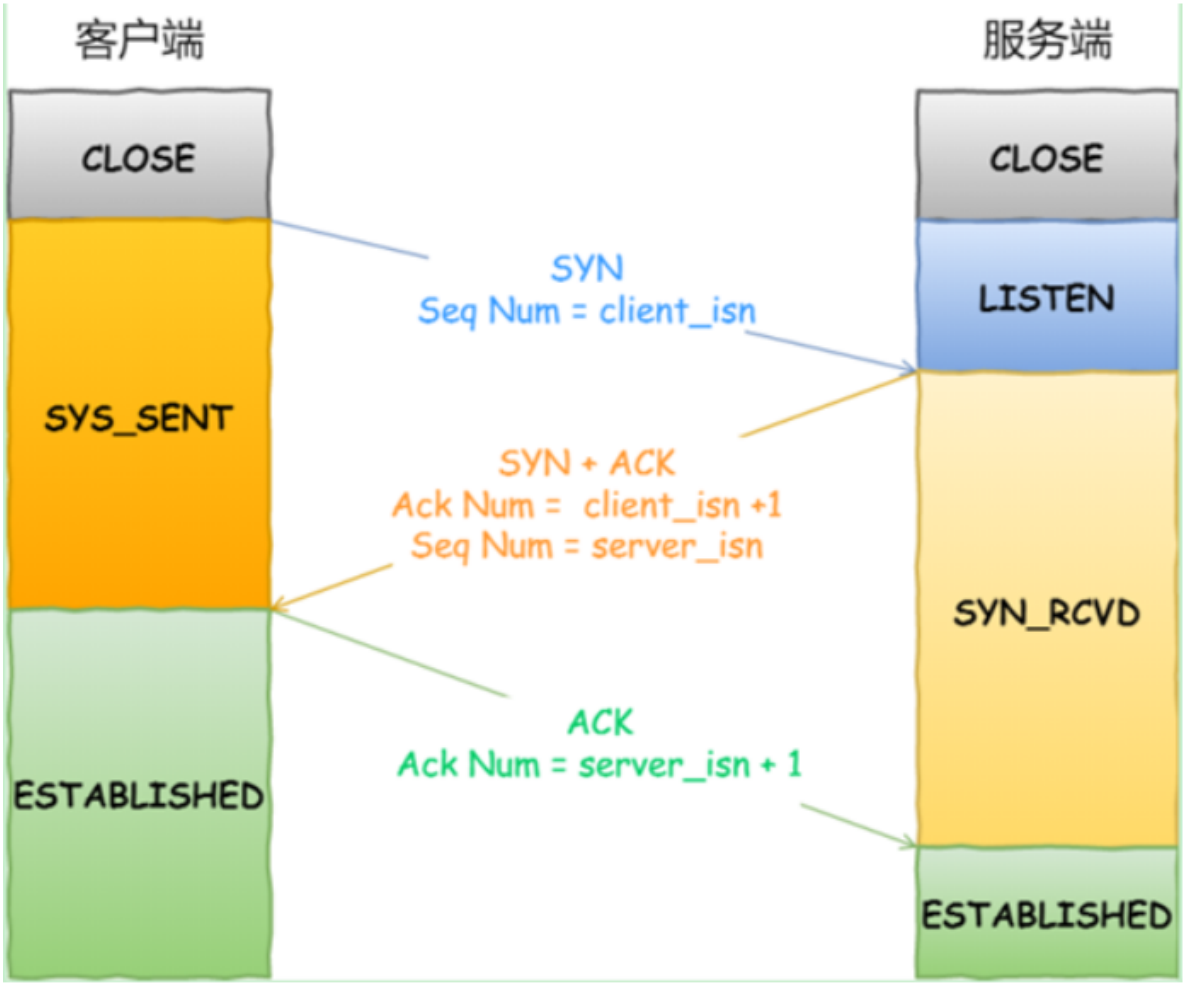
那么我们假设另一个网络的链路传输速率为每秒1M比特，有1000台计算机连接在这个网络上，假设其中500台计算机以每秒100k比特的速率分别向另外500台计算机传送数据。那么现在的问题就不是接收端的计算机能否来得及接受，而是整个网络的输入负载是否超过这个网络能够承受的限度了，这个就是拥塞控制的问题。

三次握手

客户端向服务端发起连接时，会先发一包连接请求数据过去询问一下，能否与你建立连接，这包数据我们称为SYN（森）包，如果对端同意连接，则回复一个SYN+ACK包，客户端收到回复一个ACK包，连接建立，因为这个过程中发送了三次数据，所以称之为三次握手。

为什么三次握手不是两次握手

服务端发完SYN+ACK完就建立连接，为了防止已失效的请求报文，突然又传到服务器引起错误。假设两次握手建立连接，客户端向服务端发送了一个SYN包来请求建立连接，因为某些未知的原因并未到达服务器，在中间某个网络节点产生了滞留，为了建立连接客户端会重发SYN包，这次数据包正常送达，服务端回答SYN+ACK，建立连接。但是第一包数据阻塞的网络节点突然恢复，第一包SYN包又送到服务端，这时服务端会误认为客户端又发起了一个新的连接，从而再两次握手之后进入等待数据状态。服务端认为两个连接，客户端认为是一个连接，造成了状态不一致。三次握手的话接收不到最后的ACK包，就认为连接失败，所以三次握手就是为了解决网络信道不可靠的问题。



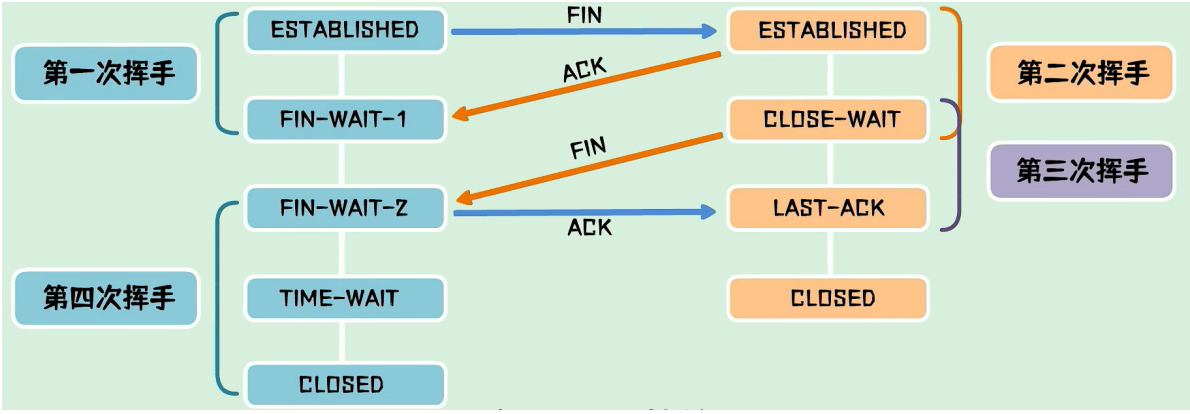
如何处理丢包、乱序问题

为了解决上述问题tcp协议为每一个连接建立了发送缓冲区，从建立连接后的第一个字节的序列号为0，后面每个字节号都会增加1，发送数据时，从发送缓冲区取一部分数据组成发送报文，在其tcp协议头中会附带序列号和长度，接收端在收到数据后，需要回复确认报文，确认报文中的ACK等于接收序列号加长度也就是下一包数据发送的起始序列号，这样一问一答的方式能够使得发送端确认发送的数据已经被对方收到，发送端也可以一次性发送连续的多包数据，接收端只需要回复一次ACK就可以了，这样发送

端可以把待发送的数据，分割成一系列的碎片发送到对端，对端根据序列和长度在接收后重构出来完整的数据，假设其中丢失了某些数据包，在接收端可以要求发送端重传。比如丢失100-199这100个字节，接收端向发送端发送ACK=100的报文，发送端收到后重传这包数据，接收端进行补齐。

四次挥手

假设客户端主动发起连接关闭请求，需要向服务端发起一包FIN包，表示关闭连接，自己进入终止等待1状态，这是第一次挥手服务端随后发送一包ACK包，表示自己进入了关闭等待状态，客户端进入终止等待2状态，这是第二次挥手，服务端此时还可以发送未发送的数据，客户端还可以接收数据，待服务端发送完数据后，发送一个FIN包进入最后确认状态，这是第三次挥手，客户端收到回复ACK包，进入超时等待状态，经过超时时间后关闭连接，服务端收到ack包后立即关闭连接，这是第四次挥手。客户端等待超时时间的原因是保证对方已经收到ACK包，假设客服端发送完最后一包ACK就释放了连接，一旦ACK包在网络中丢失，服务端将一直停留在最后确认状态，如果客户端发送最后一包ACK包后，进入停留等待，这时候服务端会因为没收到ACK包会重发FIN包，客户端会响应这个FIN包，重发ACK包并刷新超时时间。



总结：服务端需要等待完成数据的发送和处理，所以服务端的ACK和FIN会分开发送，所以导致比三次握手多了一次。

TCP和UDP的区别

1. TCP是面向连接的; 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; UDP不保证可靠交付。
3. TCP面向字节流，把数据看成一连串无结构的字节流; UDP是面向报文的。
4. TCP有拥塞控制; 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，如实时视频会议等)
5. 每一条TCP连接只能是点到点的; UDP支持一 对一、一 对多、多对一和多对多的通信方式。
6. UDP首部开销小，8B，TCP20B

Http和TCP的区别

http协议是应用层协议,主要是解决如何包装数据。而tcp协议是传输层协议,主要解决数据如何在网络中传输

SSL/TLS的区别

这两位都是安全加密协议，前后者关系，大多人已经用TLS了
TLS有更严密的警报。除了SSL的报警代码，TLS还补充了很多报警代码，如解密失败，记录溢出，拒绝访问等

说三个http的请求

GET方法 检索和获取
POST 创建和更新
PUT 向服务器提交数据，以修改数据
DELETE 删除服务器上的某些资源

Socket说一下

套接字(Socket)，就是对网络中不同主机上的应用进程之间进行双向通信的端点的抽象。

TCP粘包拆包

TCP是面向流，没有界限的一串数据。TCP底层并不了解上层业务数据的具体含义，它会根据TCP缓冲区的实际情况进行包的划分，所以在业务上认为，**一个完整的包可能会被TCP拆分成多个包进行发送，也有可能把多个小的包封装成一个大的数据包发送**，这就是所谓的TCP粘包和拆包问题。

为什么会产生粘包和拆包呢?

- 要发送的数据小于TCP发送缓冲区的大小，TCP将多次写入缓冲区的数据一次发送出去，将会发生粘包；
- 接收数据端的应用层没有及时读取接收缓冲区中的数据，将发生粘包；
- 要发送的数据大于TCP发送缓冲区剩余空间大小，将会发生拆包；
- 待发送数据大于MSS（最大报文长度），TCP在传输前将进行拆包。即TCP报文长度-TCP头部长度>MSS。

解决方案:

- 发送端将每个数据包封装为固定长度
- 在数据尾部增加特殊字符进行分割
- 将数据分为两部分，一部分是头部，一部分是内容体；其中头部结构大小固定，且有一个字段声明内容体的大小。

总结：TCP是面向流，没有界限的一串数据，TCP底层不了解上层业务数据的具体含义，会根据TCP缓存区情况进行包的划分，TCP将多次写入缓冲区数据一次发送出去，以及接收端应用层没有及时读取接收缓冲区数据，将发生粘包。发送数据大于缓冲区剩余空间大小，待发送数据大于MSS（最大报文长度），将拆包。解决办法主要有将每个包封装为固定长度、数据尾部增加特殊字符进行分割、将数据分为头部和内容体，头部结构大小固定，且有一个字段声明内容体大小。

传输层和网络层的作用

传输层为相互通信的应用进程提供了逻辑通信。主要包括两个协议：TCP协议和UDP协议。

网络层负责为**分组交换网上的不同主机提供通信服务**

Cookie和Session的区别

总结：Cookie 一般用来保存用户信息，只能存储 ASCII 码字符串，存储在客户浏览器中，容易被恶意查看。Session 的主要作用就是通过服务端记录用户的状态，可以存储任何类型的数据，在考虑数据复杂性的时候首选session，存储在服务器。