# Java基础

**一：基础语法**

**1.JDK：Java开发包 = JRE+开发工具集**

JRE = JVM(Java虚拟机) + JavaSE标准类库

**2.变量、运算符等基础部分容易忘记的：**

Int I,j ：一行中可申明多个变量

用Final来指示常量：Final double pai = 3.14（一旦赋值不能更改）

除号运算符相关：15/2=7，15%2=1，15.0/2=7.5

instanceof：判断对象是属于哪个类的实例

输入Scanner类：

public class Scantry {

public static void main(String[] args){

Scanner scan = new Scanner(System.in);

System.out.println("请输入你的名字");

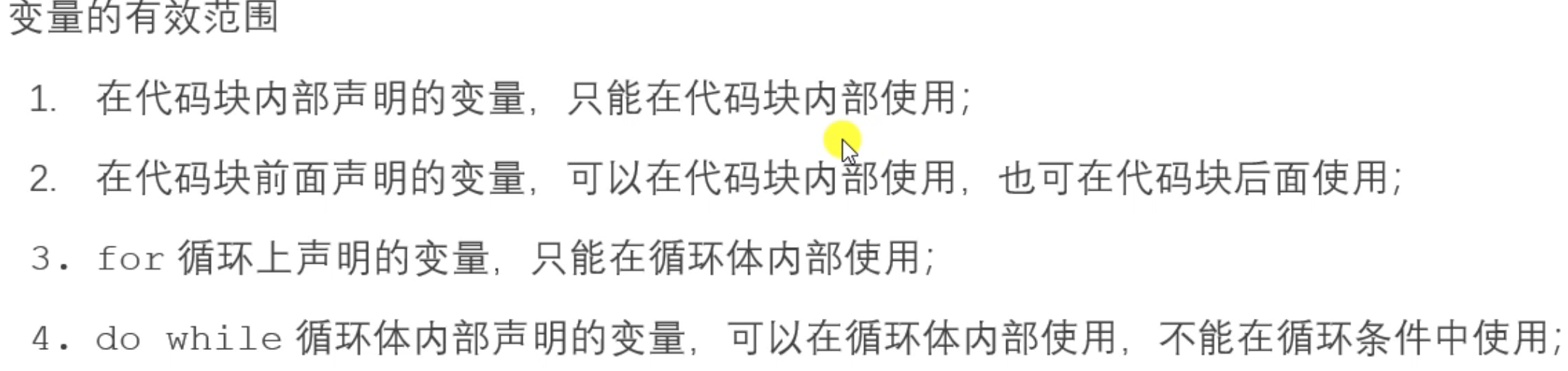
String name = scan.nextLine();

Scan.close();

}

}

for循环中：continue略过本次循环；break跳出循环



**3.数组**

1.保存同一类数据，长度不可改变

2.Int[ ] newarray = new int[7]

1. **Foreach循环**：专门遍历数组或集合：int[] array1 = new int[8];
2. *//foreach循环遍历*
3. int min = array1[0];
4. **for(int i:array1)**{
5. *//此处i不是序号而是数组中具体的值*
6. if(i<min){
7. min = 1;
8. }
9. }
10. System.out.println(min);

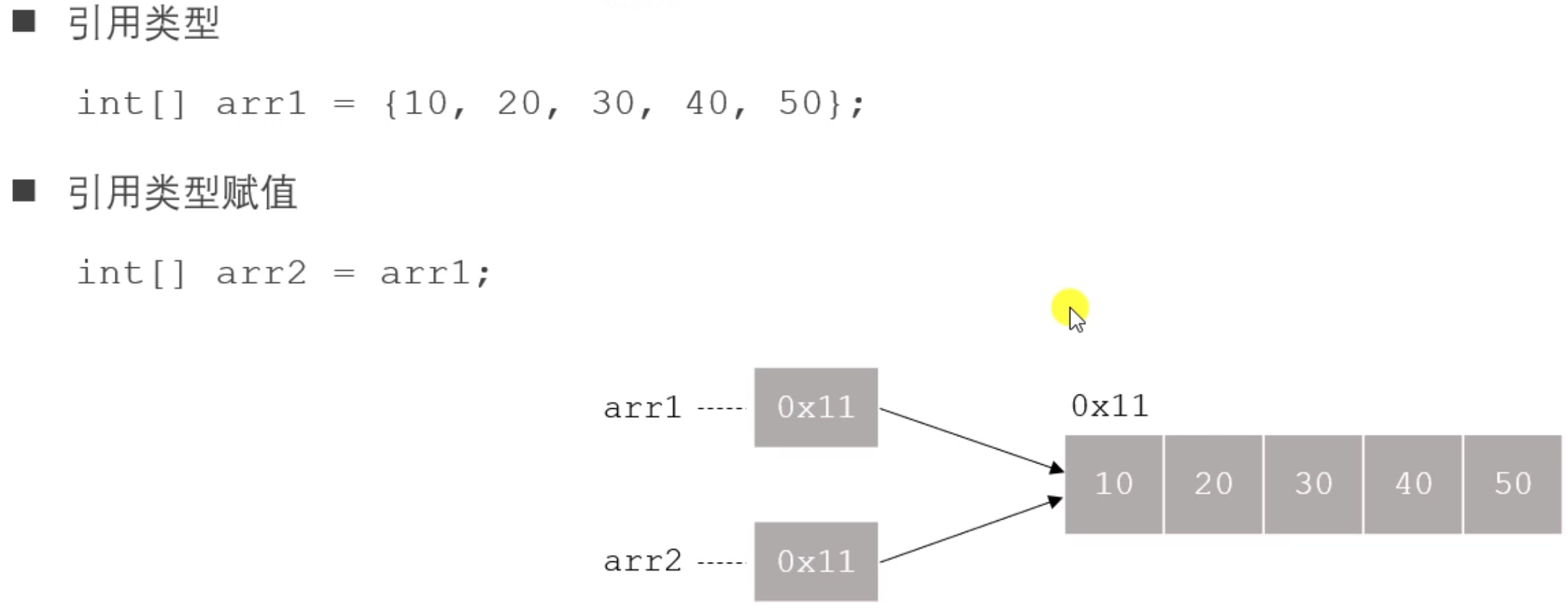
3. **Arrays**工具类：

数组转字符串显示：**Arrays**.toString(array1)

数组排序：Arrays.sort(array1)

数组比较是否相等：Arrays.equals(arr1,arr2)

**4.数组是引用类型**，int[ ] array1只是开辟数组地址0x11，指向真实数据所在的地址



**5.二维数组**：

声明二维数组：type[ ][ ] arrayName;

动态初始化：arrayName = new type[outLength][innerLength]

转化为字符串：Arrays.deeptoString()

**6.数组倒转**：

public class 数组倒转 {  
 public static void main(String[] args){  
 int[] muniArray = new int[6];  
 for(int i=0;i< muniArray.length;i++){  
 muniArray[i] = (int)(Math.*random*() \* 100);  
 }  
 System.*out*.println(Arrays.*toString*(muniArray));  
 for(int j=0;j<muniArray.length/2;j++){  
 int temp = muniArray[j];  
 muniArray[j] = muniArray[muniArray.length-1-j];  
 muniArray[muniArray.length-1-j] = temp;  
 }  
 System.*out*.println(Arrays.*toString*(muniArray));  
 }  
}

**7.冒泡排序：**

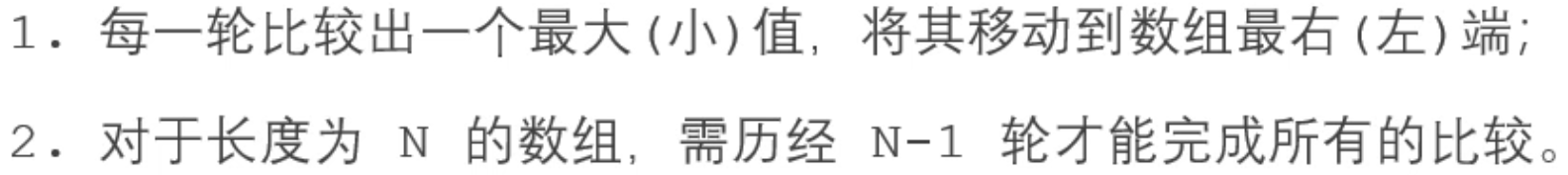
**for(int i = 0; i<Array.length-1; i++){**

**for(int j=0 ; j<Array.length-1-i ;j++){**

**//arr[j]和arr[j+1]比**

**}**

**}**



**8.方法**

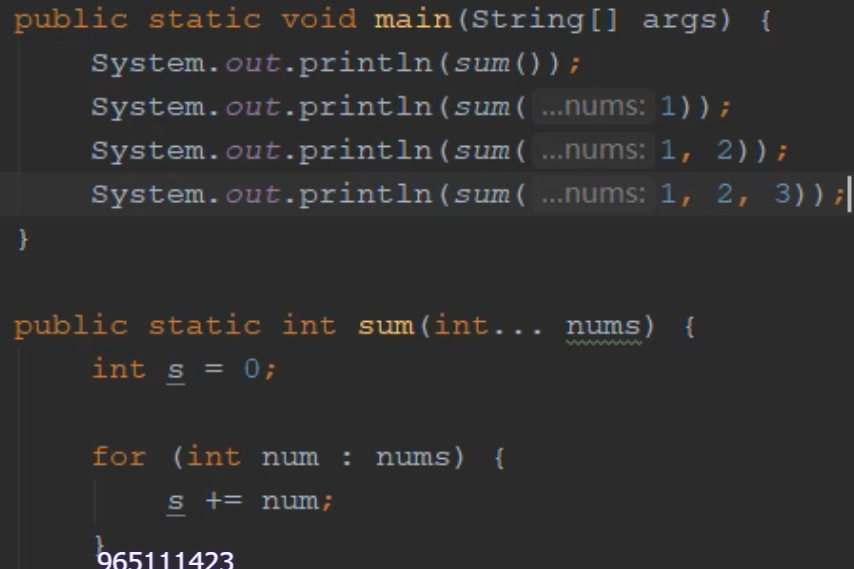
**1.**Public static int sum(int a,int b){

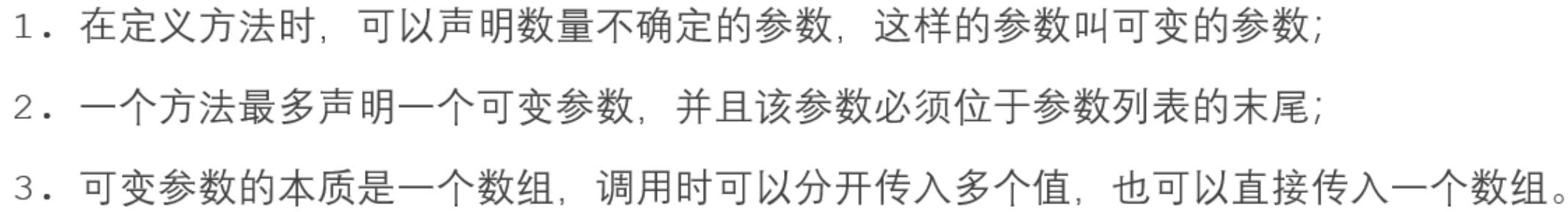
Return a+b;

}

若没有返回值：则public static **void**(…)

**2.可变参数：**





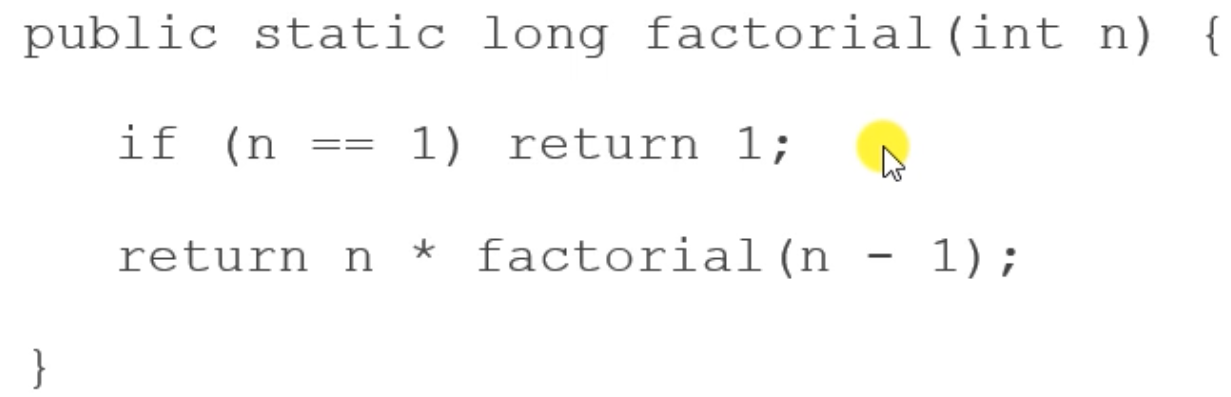
**3.方法重载**：方法名一样，参数不同，与其他无关

示例：对整数求和，浮点数求和等等，重载方法:

sum(int … nums)；sum(float … nums)；等等

**4.方法递归**：一个方法调用它自身，隐循环，重复执行某段代码无需循环控制

示例：求n的阶乘：



**9.调试**

1.debugger中按F8 向下执行一步

2.F9：执行到下一个断点位置

**二：面向对象**

1.

# 计算机网络

**1.** 通过发送请求获取服务器资源的Web浏览器等，都可称为客户端（client），Web是建立在HTTP协议上通信的

**2.**把与互联网相关联的协议集合起来总称为**TCP/IP**。也有说法认为TCP/IP是指TCP和IP这两种协议 or TCP/IP是在IP协议的通信过程中，使用到的协议族的统称

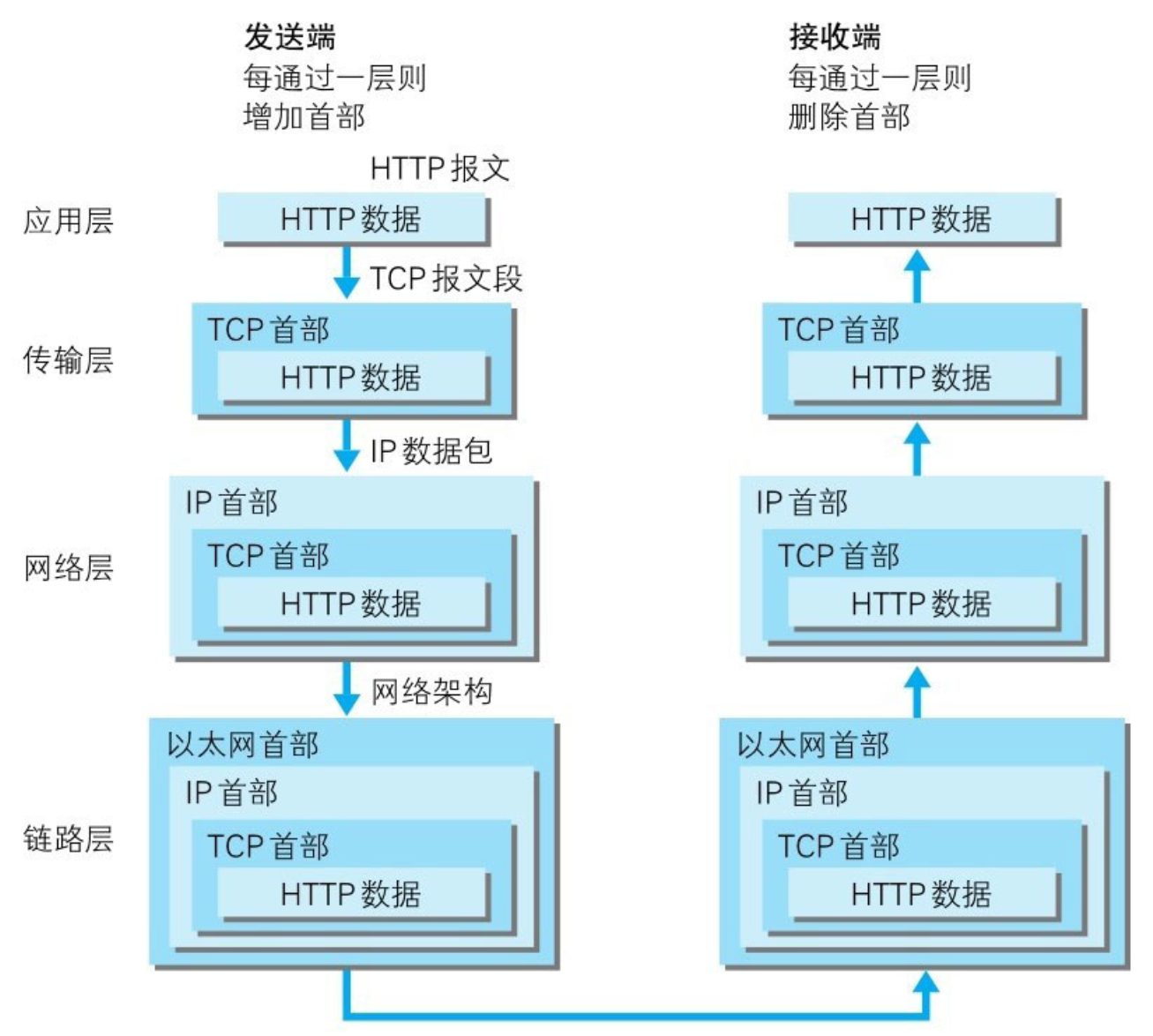
**TCP/IP四层网络协议模型：应用层、传输层、网络层和数据链路层**。分层好处：改变设计时较为自由

**应用层**：决定向用户提供应用服务时通信的活动，如**FTP**文件传输协议，**DNS**域名系统服务，**HTTP**也属于该层

**传输层：**对上层应用层，提供处于网络连接中的两台计算机之间的数据传输。有两个性质不同的协议：**TCP（传输控制协议）**和**UDP（用户数据报协议）**

**网络层：**用来处理在网络上流动的数据包。数据包是网络传输的最小数据单位。该层规定了通过怎样的路径（所谓的传输路线）到达对方计算机，并把数据包传送给对方。作用就是在众多的选项内选择一条传输路线。**（IP协议）**

**链路层：**用来处理连接网络的硬件部分。包括控制操作系统、硬件的设备驱动、网卡及光纤等物理可见部分。硬件上的范畴均在链路层的作用范围之内。



发送端**从**应用层往下走，接收端则**往**应用层往上走。

用HTTP举例，首先作为发送端的客户端在应用层（HTTP协议）发出一个想看某个Web页面的HTTP请求。接着为传输方便，在传输层（TCP协议）把从应用层处收到的数据（HTTP请求报文）进行分割，并在各个报文上打上标记序号及端口号(封装)后转发给网络层。在网络层（IP协议）增加作为通信目的地的MAC地址后转发给链路层。

接收端的服务器在链路层接收到数据，按序往上层发送一直到应用层。当传输到应用层，才能算真正接收到由客户端发送过来的HTTP请求。

**OSI七层网络协议模型：**

**7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，支持万维网应用的***HTTP协议***，支持电子邮件的SMTP协议等。

**6表示层：**主要负责数据格式的转换，如加密解密、转换翻译、压缩解压缩等。

**5会话层：**负责在网络中的两节点之间建立、维持和终止通信，如服务器验证用户登录便是由会话层完成的。

**4运输层：**有时也译为传输层，向主机进程提供通用的数据传输服务。该层主要有以下两种协议：

**TCP：**提供面向连接的、可靠的数据传输服务；

**UDP：**提供无连接的、尽最大努力的数据传输服务，但不保证数据传输的可靠性。

**3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。

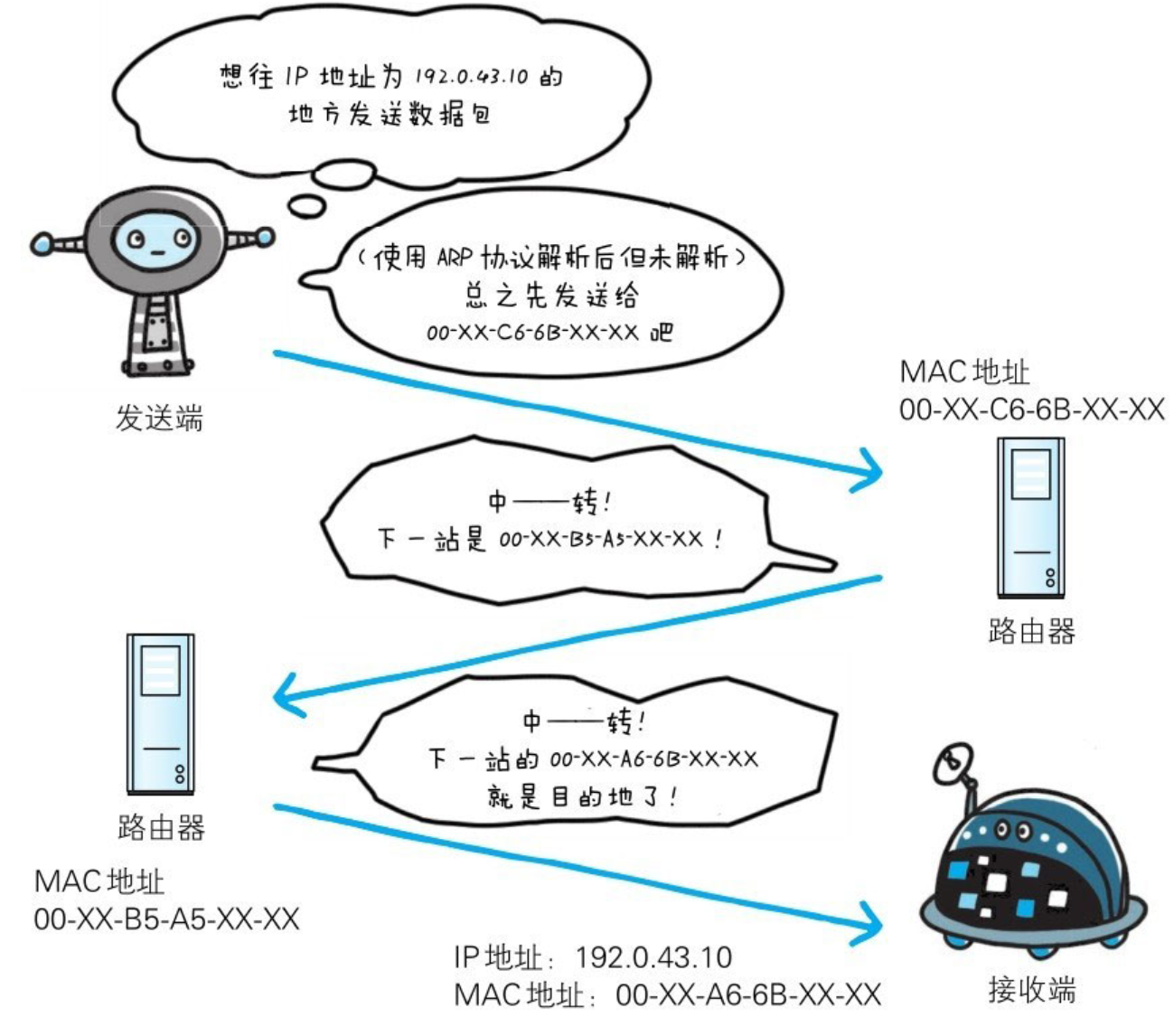
**2数据链路层：**数据链路层通常简称为链路层。将网络层传下来的IP数据包组装成帧，并再相邻节点的链路上传送帧。

**1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通信手段的差异。

**3. IP和TCP**

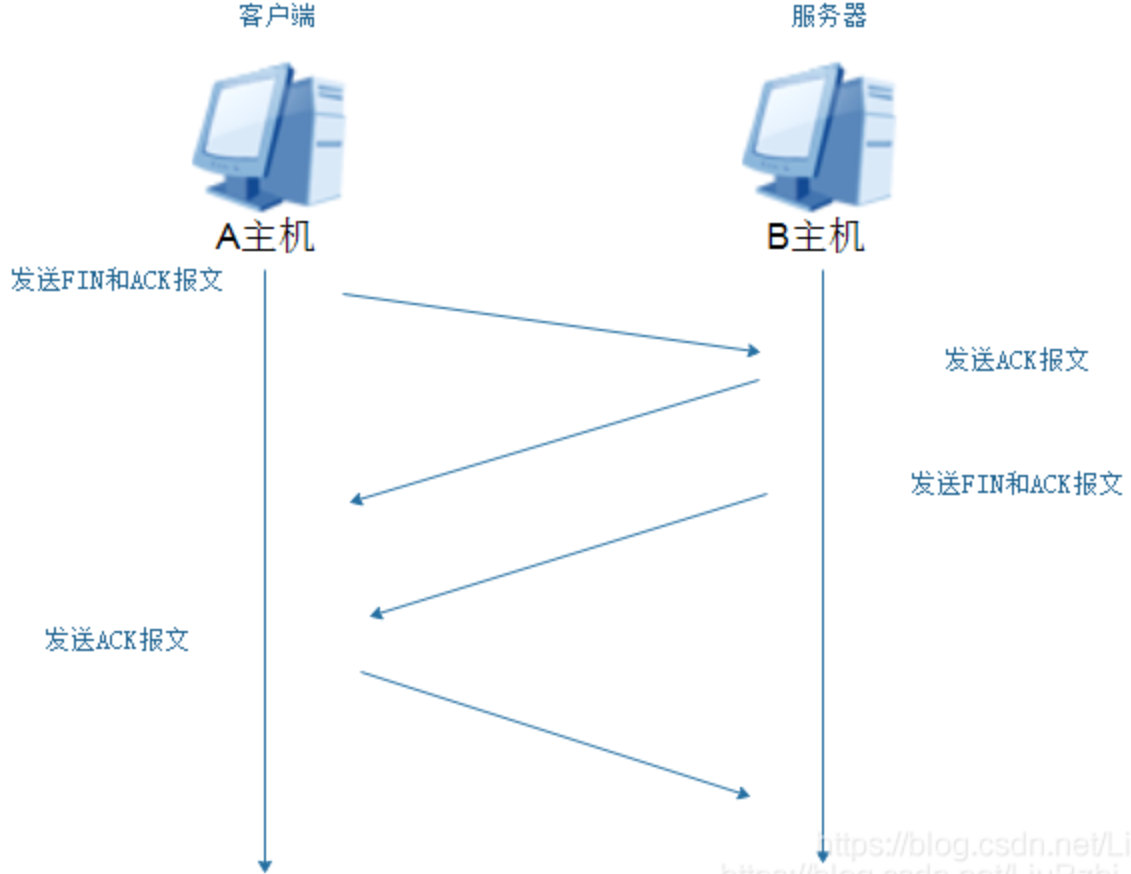
**IP(网际协议)：**把各种数据包传送给对方。要保证确实传送到对方那里**需满足IP地址和MAC地址**，IP地址指明了节点被分配到的地址，MAC地址是指网卡所属的固定地址。IP地址和MAC地址进行配对。IP地址可变换，但MAC地址基本上不会更改。

通信中通常经过多台计算机和网络设备中转才能连接到对方。中转时会利用下一站中转设备的MAC地址来搜索下一个中转目标。这时会采用**ARP协议**（Address Resolution Protocol）。**ARP是一种用以解析地址的协议，根据通信方的IP地址就可以反查出对应的MAC地址**。



**4.TCP协议**：位于传输层，提供**可靠的字节流服务**：为方便传输将大块数据分割成以报文段为单位的数据包进行管理，同时能确定数据最终是否送达到对方。

**TCP建立连接三次握手，TCP断开连接四次挥手**：



**三次握手：**

（1）第一次握手

PC1使用一个随机的端口号向PC2的80端口发送连接请求，此过程的典型标志为SYN控制位为1，其他五位为0。

（2）第二次握手

首先PC2收到PC1请求向PC1回复确认信息。并且PC2也向PC1发送建立连接请求。

（3）第三次握手

PC1收到PC2回复，也要向PC1回复一个确认信息。

**四次挥手：**

中断连接端可以是Client端，也可以是Server端。

假设Client端发起中断连接请求，也就是**发送FIN报文**。Server端接到FIN报文后，意思是说"我Client端没有数据要发给你了"，但是如果你还有数据没有发送完成，则不必急着关闭Socket，可以继续发送数据。所以你**先发送ACK**，"告诉Client端，你的请求我收到了，但是我还没准备好，请继续你等我的消息"。这个时候Client端就**进入FIN\_WAIT状态**，继续等待Server端的FIN报文。当Server端确定数据已发送完成，则**向Client端发送FIN报文**，"告诉Client端，好了，我这边数据发完了，准备好关闭连接了"。Client端收到FIN报文后，"就知道可以关闭连接了，但是他还是不相信网络，怕Server端不知道要关闭，所以**发送ACK后进入TIME\_WAIT状态**，如果Server端没有收到ACK则可以重传。“，Server端收到ACK后，"就知道可以断开连接了"。Client端等待了2MSL后依然没有收到回复，则证明Server端已正常关闭，那好，我Client端也可以关闭连接了。TCP连接就这样关闭了！

在TCP断开连接的过程中，有一个半关闭得概念。TCP一端可以中止发送数据，但是仍然可以接收数据，称之为半关闭：

（1）客户端发送FIN，半关闭了这个链接。服务器发送ACK接受半关闭。

（2）服务器继续发送数据，而客户端只发送ACK确认，不发送任何数据。

（3）当服务器所有数据传输完毕，就发送FIN报文段，客户再发送ACK报文段，这样就关闭了TCP连接。

**面试题延伸：**

1、三次握手和四次挥手的本质是什么？

三次握手的本质是确认通信双方收发数据的能力 。

四次挥手的目的是关闭一个连接 。

2、为什么TCP连接的时候是3次？2次不可以吗？

因为需要考虑连接时丢包的问题，如果只握手2次，第二次握手时如果服务端发给客户端的确认报文段丢失，此时服务端觉得自己已经连接成功，而客户端不知道服务端是否已经准备好，就不会给服务端发数据。

如果是三次握手即便发生丢包也不会有问题，如果第三次握手客户端发的确认ack报文丢失，服务端在一段时间内没有收到确认ack报文的话会重新进行第二次握手，也就是服务端会重发SYN报文段。

3、为什么TCP连接的时候是3次，关闭的时候却是4次？

因为只有在客户端和服务端都没有数据要发送的时候才能断开TCP。

客户端FIN时只能保证客户端没有数据发，服务端收FIN只能先回复一个确认报文来告诉客户端我服务端已经收到FIN，但要等我服务端数据发完了才能给客户端发FIN报文(不能一次性将确认报文和FIN报文发给客户端，就是这里多出来了一次)。

4、为什么客户端发出第四次挥手的确认报文后要等2MSL的时间才能释放TCP连接？即为什么客户端在TIME-WAIT阶段要等2MSL？

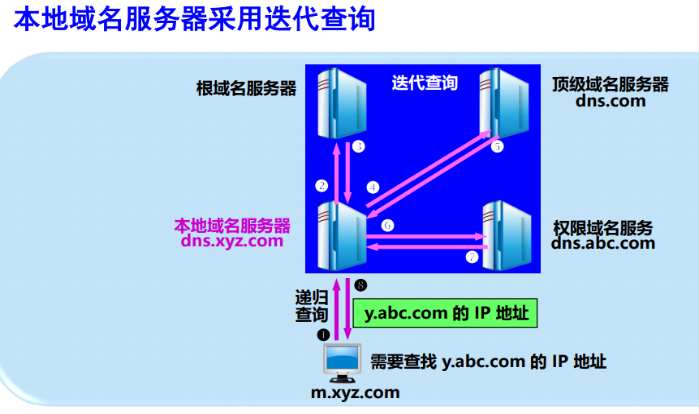
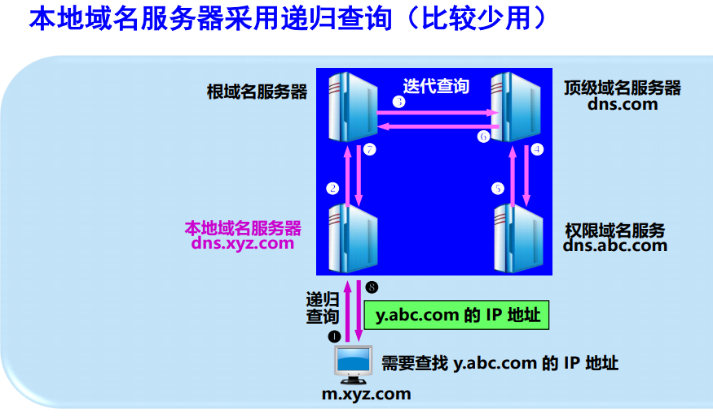
MSL 指的是一段 TCP 报文在传输过程中的最大生命周期。

2MSL 即是服务器端发出为 FIN 报文和客户端发出的 ACK 确认报文所能保持有效的最大时长。同样是考虑丢包的问题，如果第四次挥手的报文丢失，服务端没收到确认ack报文就会重发第三次挥手的报文，这样报文一去一回最长时间就是2MSL，所以需要等这么长时间来确认服务端确实已经收到了。

**5.DNS服务**

DNS服务是和HTTP协议一样位于应用层的协议。它***提供域名到IP地址之间的解析服务***。（通常是将域名解析为IP地址）

**域名解析：迭代和递归：**

主机向本地域名服务器的查询一般都是采用递归查询，本地域名服务器向根域名服务器的查询通常是采用迭代查询。

每个域名服务器都维护一个高速缓存，存放最近用过的名字以及从 何处获得名字映射信息的记录。可大大减轻根域名服务器的负荷，使互联网上的 DNS 查询请求和回答报文的数量大为减少。为保持高速缓存中的内容正确，域名服务器应为每项内容设置计时器，当权限域名服务器回答查询请求时，在响应中都指明绑定的有效时间值。**增加此时间值可减少网络开销，而减少此时间值可提高域名转换的准确性。**

**7. TCP和UDP的区别**

TCP 面向连接提供可靠的服务，UDP 是无连接的即发送数据之前不需要建立连接，UDP 尽最大努力交付，即不保证可靠交付。

UDP实时性好，工作效率比 TCP 高，适用于对高速传输和实时性有较高要求的

TCP 连接只能是一对一的，UDP 支持一对一，一对多，多对一和多对多、

UDP 分组首部开销小，TCP 首部开销 20 字节，UDP 的首部开销小，只有 8 个字节。

TCP 面向字节流，把数据看成一连串无结构的字节流；UDP 面向报文，一次交付一个完整的报文，不可分割，报文是 UDP 数据报处理的最小单位。

UDP 适合一次性传输较小数据的网络应用，如 DNS 等。

**8. TCP可靠传输的实现：**

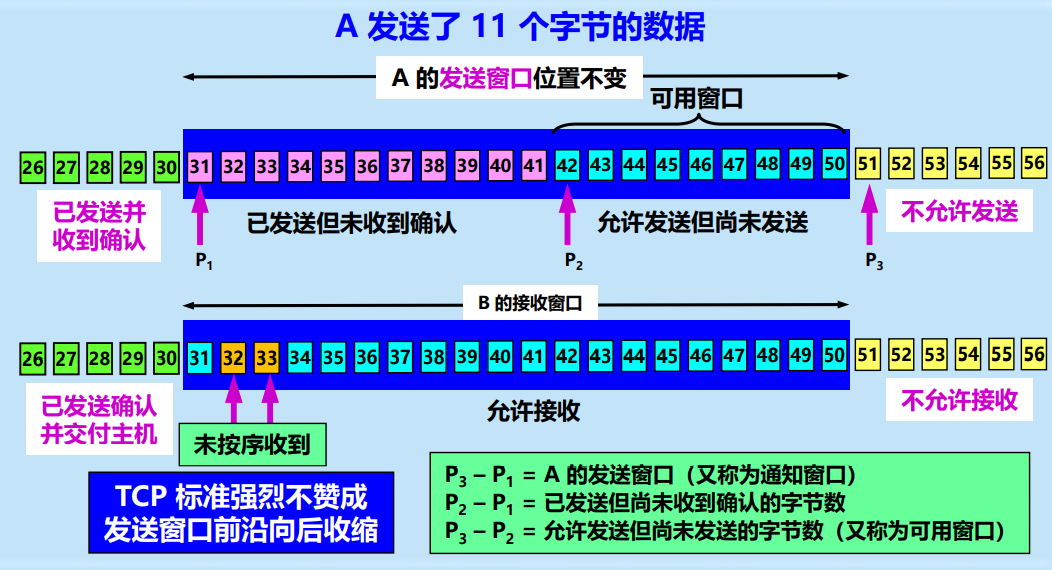
**使用流水线传输和滑动窗口协议，以字节为单位，若A 的发送窗口内的序号都已用完，但还没有再收到确认，必须停止发送。**

**发送缓存**用来暂时存放：1.发送应用程序传送给发送方TCP准备发送的数据；2.TCP已发送出但尚未收到确认的数据。

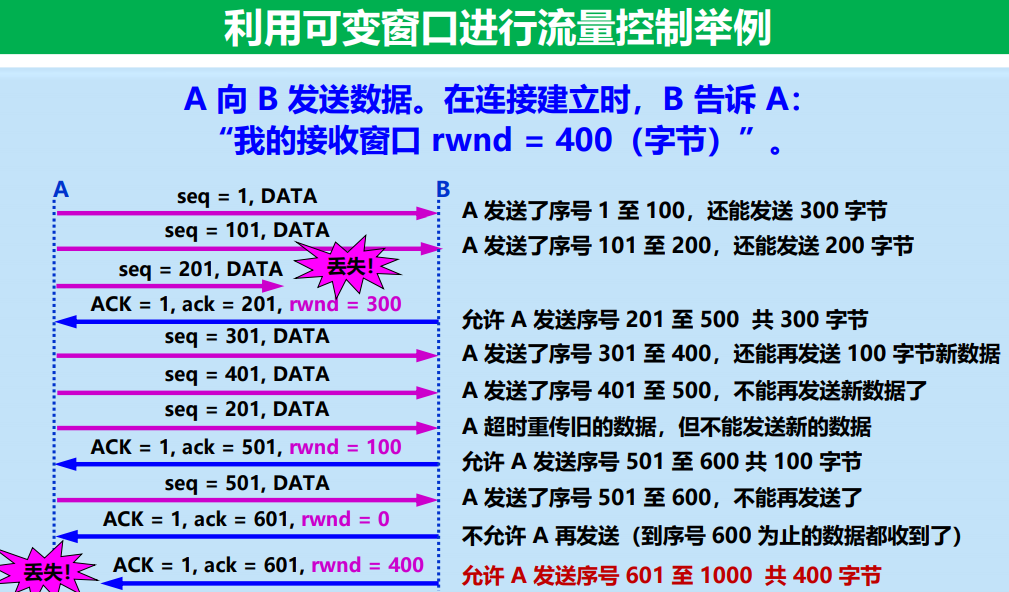
**接收缓存**用来暂时存放：1. 按序到达的、但尚未被接收应用程序读取的数据；2.不按序到达的数据

**超时重传时间设置：**TCP 采用了一种自适应算法，它记录一个报文段发出的时间，以及收到相应的确认的时间。这两个时间之差就是报文段的往返时间RTT

**选择确认SACK：**设法只传送缺少的数据



**TCP的流量控制**：让发送方的发送速率不要太快，既要让接收方来得及接收，也不要使网络发生拥塞。



最后B向A发出的丢失，会出现互相等待的**死锁**局面。为解决，TCP 为每一个连接设一个**持续计时器，**TCP 连接的一方收到对方的零窗口通知，就启动该持续计时器。若时间到期，就发送一个零窗口探测报文段（仅携带 1 字节数据），而对方就在确认这个探测报文段时给出了现在的窗口值。仍未0，重设，不为0，打破。

**TCP的拥塞控制**：对网络中某资源的需求超过了该资源所能提供的可用部分，网络的

性能就要变坏，这种现象称为拥塞，简单增加资源不能解决网络拥塞。

开环控制：在设计网络时事先考虑周全，力求工作时不发生拥塞。

闭环控制：在发生拥塞后，采取措施进行控制，消除拥塞，**措施：**监测网络系统、将拥塞发生的信息传送到可采取行动的地方、调整网络系统的运行。

**连接建立和释放：三次握手和四次握手**：1. 连接建立2. 数据传送3. 连接释放，采用客户服务器方式，主动发起连接建立的应用进程叫做客户，等待连接的叫做服务器。

9.以太网(局域网)：

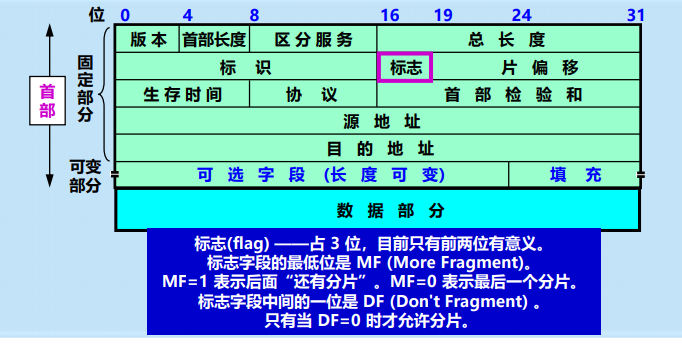
**特点：**1.为一个单位所拥有；2地理范围和站点数目均有限，拓补结构：星型，总线型，环型

以太网发送数据都采用**曼彻斯特编码**，用较为灵活的无连接的工作方式，不必先建立连接就可以直接发送数据；对发送的数据帧不进行编号，也不要求对方发回确认；以太网提供的服务是不可靠的交付，即尽最大努力的交付；如果高层发现丢失了一些数据而进行重传，但以太网并不知道这是一个重传的帧，而是当作一个新的数据帧来发送。

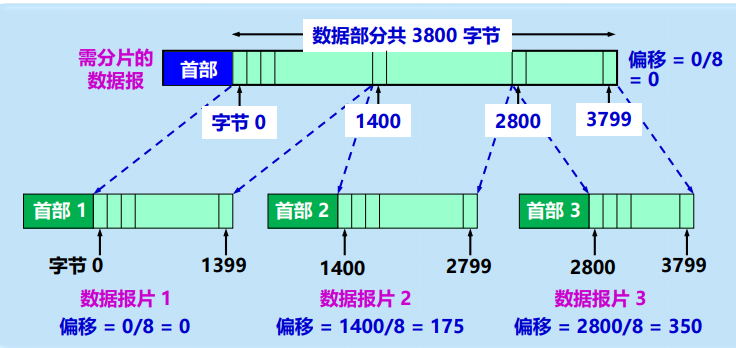
10. IP数据报格式

由首部和数据两部分组成，**首部的前一部分是固定长度，共 20 字节**，是所有 IP 数据报必须具有的。

**长度分片**



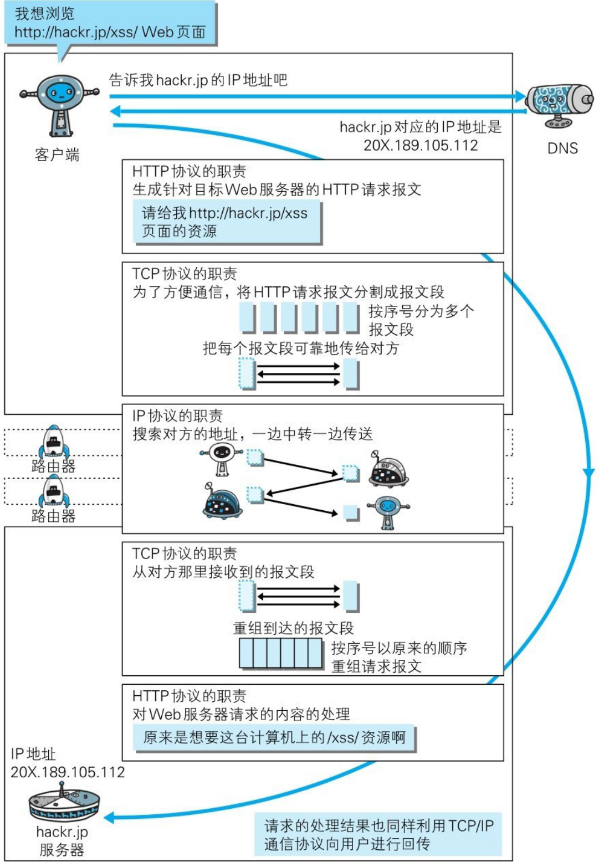
总长度3820，数据部分3800，分片为长度不超过1420，片偏移以8个字节为偏移单位。



**11.URI和URL**

**URL(统一资源定位符)**：使用Web浏览器等访问Web页面时需要输入的网页地址，如<http://hackr.jp/>

**URI(统一资源标识符)**：用字符串标识某一互联网资源，而URL表示资源的地点（互联网上所处的位置）。可见URL是URI的子集。

**浏览器输入URL到显示网页过程？**

**1.域名解析**（域名 www.baidu.com 变为 ip 地址）：浏览器搜索自己的DNS缓存；若没有，则搜索操作系统的DNS缓存；若没有，则搜索操作系统的hosts文件。若都没有，则找 tcp/ip 参数中设置的首选 dns 服务器，即本地 dns 服务器（***递归查询***），本地域名服务器查询自己的dns缓存，***如果没有，则进行迭代查询***。将本地dns服务器将IP返回给操作系统，同时缓存IP。

**2.发起 tcp 的三次握手，建立 tcp 连接**。浏览器会以一个随机端口（1024-65535）向服务端的 web 程序 80 端口发起 tcp 的连接。

**3.建立 tcp 连接后发起 http 请求。**

**4.服务器响应 http 请求，客户端得到 html 代码。**服务器 web 应用程序收到 http 请求后，就开始处理请求，处理之后就返回给浏览器 html 文件。

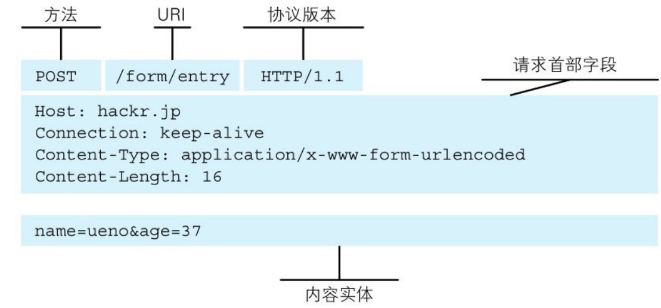
**5.浏览器解析 html 代码，并请求 html 中的资源**。

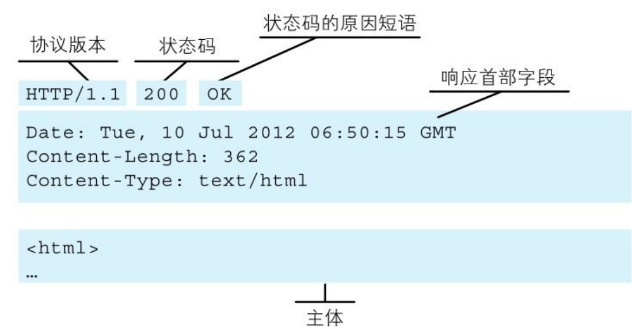
**6.浏览器对页面进行渲染，并呈现给用户。**

**13.HTTP请求报文和响应报文的格式**

HTTP是应用层协议，用于客户端和服务器端之间的通信

**从客户端发送给某个HTTP服务器端的请求报文的构成：**

1. 请求行（请求方法+URI协议+版本）
2. 请求头部
3. 空行
4. 请求主体

**服务器端响应报文的构成：**

1. 状态行（版本+状态码+原因短语）
2. 响应首部
3. 空行
4. 响应主体

**14.HTTP常用请求方法？**

| **方法** | **作用** |
| --- | --- |
| **GET** | **获取资源** |
| **POST** | **传输实体主体** |
| **PUT** | **上传文件** |
| **DELETE** | **删除文件** |
| HEAD | 和GET方法类似，但只返回报文首部，不返回报文实体主体部分 |
| PATCH | 对资源进行部分修改 |
| OPTIONS | 查询指定的URL支持的方法 |
| CONNECT | 要求用隧道协议连接代理 |
| TRACE | 服务器会将通信路径返回给客户端 |

为了方便记忆，可以将PUT、DELETE、POST、GET理解为客户端对服务端的增删改查。

* PUT：上传文件，向服务器添加数据，可以看作增
* DELETE：删除文件
* POST：传输数据，向服务器提交数据，对服务器数据进行更新。
* GET：获取资源，查询服务器资源



**GET和POST请求的区别？**

**使用上的区别：**

* GET使用URL或Cookie传参，而POST将数据放在BODY中
* GET方式提交的数据有长度限制，则POST的数据则可以非常大
* POST比GET安全，因为数据在地址栏上不可见**本质区别**

GET和POST最大的区别主要是***GET请求是幂等性的，POST请求不是***

幂等性是指一次和多次请求某一个资源应该具有同样的副作用，即对同一URL的多个请求应该返回同样的结果。

**15.返回结果的HTTP状态码**

常见状态码：

* 200：服务器已成功处理了请求。
* 301：(永久移动) 请求的网页已永久移动到新位置。 服务器返回此响应(对 GET 或 HEAD 请求的响应)时，会自动将请求者转到新位置。
* 302：(临时移动) 服务器目前从不同位置的网页响应请求，但请求者应继续使用原有位置来进行以后的请求。
* 400 ：客户端请求有语法错误，不能被服务器所理解。
* 403 ：服务器收到请求，但是拒绝提供服务。
* 404 ：(未找到) 服务器找不到请求的网页。
* 500： (服务器内部错误) 服务器遇到错误，无法完成请求。

[](https://camo.githubusercontent.com/03c46d524c6baa29152bcd835cc28ffa42cbee544388b1b22a50403fdeecadc6/687474703a2f2f626c6f672d696d672e636f6f6c73656e2e636e2f696d672f696d6167652d32303231303532353131343433393734382e706e67)

**301和302的区别？**

都表示重定向，浏览器拿到服务器返回的这个状态码后会自动跳转到一个新的URL地址

***不同点：***301表示旧地址A的资源已被永久地移除了(这个资源不可访问了)；302表示旧地址A的资源还在（仍可访问），这个重定向只是临时从旧地址A跳转到地址B，搜索引擎会抓取新内容而保存旧的网址。

**16. HTTP1.0和HTTP1.1的区别?**

* **长连接和管线化**：HTTP 1.1支持***长连接***(不用每进行一次http就断开tcp连接，在响应头上加入：Connection:keep-alive)和请求的***流水线处理***(不用等待响应亦可直接发送下一个请求,这样就能够做到同时并行发送多个请求)。减少了建立和关闭连接的消耗和延迟，弥补了HTTP1.0每次请求都要创建连接的缺点。
* **缓存处理**：在HTTP1.0中主要使用header里的If-Modified-Since,Expires来做为缓存判断的标准，HTTP1.1则引入了更多的缓存控制策略，可供选择的缓存头来控制缓存策略。
* **带宽优化及网络连接的使用**：HTTP1.0中存在浪费带宽现象且不支持断点续传，HTTP1.1在请求头引入了range头域，允许只请求资源的某个部分，即返回码是206，方便自由选择以便于充分利用带宽和连接。
* **错误通知的管理**：HTTP1.1中新增了24个错误状态响应码，如409（Conflict）表示请求的资源与资源的当前状态发生冲突；410（Gone）表示的某资源被永久删除。
* **Host头处理**：在HTTP1.0中认为每台服务器都绑定一个唯一的IP地址，因此请求消息中的URL并没有传递主机名（hostname）。但现在一台物理服务器上可以存在多个虚拟主机并共享一个IP地址。HTTP1.1的请求消息和响应消息都应支持Host头域，且请求消息中如果没有Host头域会报告一个错误（400 Bad Request）

**17.HTTP与HTTPS的区别？**

HTTP主要有这些不足:

1通信使用明文（不加密），内容可能会被窃听

2不验证通信方的身份，因此有可能遭遇伪装

3无法证明报文的完整性，所以有可能已遭篡改

**把添加了加密及认证机制的HTTP称为 HTTPS（HTTP Secure）**

|  | HTTP |
| --- | --- |
| 端口 | 80 | 443 |
| 安全性 | 无加密，安全性较差 | 有加密机制，安全性较高 |
| 资源消耗 | 较少 | 由于加密处理，资源消耗更多 |
| 是否需要证书 | 不需要 | 需要 |
| 协议 | 运行在TCP协议上 | 运行在SSL协议上，SSL运行在TCP协议上 |

**使用SSL时，则演变成先和SSL通信，再由SSL和TCP通信了。简言之，所谓HTTPS，其实就是身披SSL协议这层外壳的HTTP**

**18.什么是cookie和session及其区别？**

***Cookie****：*根据从服务器端发送的响应报文内的一个叫做Set-Cookie的首部字段信息，通知客户端保存Cookie，当下次客户端再往该服务器发送请求时，客户端会自动在请求报文中加入Cookie值后发送出去。使基于无状态的 HTTP 协议记录稳定的状态信息成为可能.

主要用于以下三个方面：

* 会话状态管理（如用户登录状态、购物车、游戏分数或其它需要记录的信息）
* 个性化设置（如用户自定义设置、主题等）
* 浏览器行为跟踪（如跟踪分析用户行为等）

***Session***：代表着服务器和客户端一次会话的过程。Session 对象存储特定用户会话所需的属性及配置信息，***在应用程序的 Web 页之间跳转时****，存储在 Session 对象中的变量将不会丢失而是在整个会话中一直存在下去*。客户端关闭会话或 Session 超时失效时会话结束。

***两者如何配合的？***

第一次请求服务器 → 服务器根据用户提交的信息创建对应 Session

请求返回时将此 Session 的唯一标识信息 SessionID 返回给浏览器，浏览器接收到后存入 Cookie 中，同时Cookie记录此SessionID属于哪个域名。

用户第二次访问服务器 → 请求自动判断此域名下是否存在 Cookie 信息，如存在自动将 Cookie 信息也发给服务端，服务端从 Cookie 中获取 SessionID，再根据 SessionID 查找对应的 Session 信息。

如没有找到Session说明用户没有登录或登录失效，如找到 Session 证明用户已经登录可执行后面操作。SessionID 是连接 Cookie 和 Session 的一道桥梁，大部分系统也是根据此原理来验证用户登录状态。

***Cookie和Session的区别？***

**1.作用范围不同**：Cookie 保存在客户端（浏览器），Session 保存在服务器端。

**2.存取方式的不同**：Cookie 只能保存 ASCII，Session 可以存任意数据类型**(UserId)** 等。

**3.有效期不同**：Cookie 可设置为长时间保持，比如我们经常使用的默认登录功能，Session 一般失效时间较短，客户端关闭或者 Session 超时都会失效。

**4.隐私策略不同**：Cookie 存储在客户端容易遭到不法获取；Session 存储在服务端，安全性相对 Cookie 要好一些。

**5.存储大小不同**：单个 Cookie 保存的数据不超过 4K，Session 可存数据远高于

# 大数据组件

1.

2.

# 操作系统

1.什么是操作系统

2.