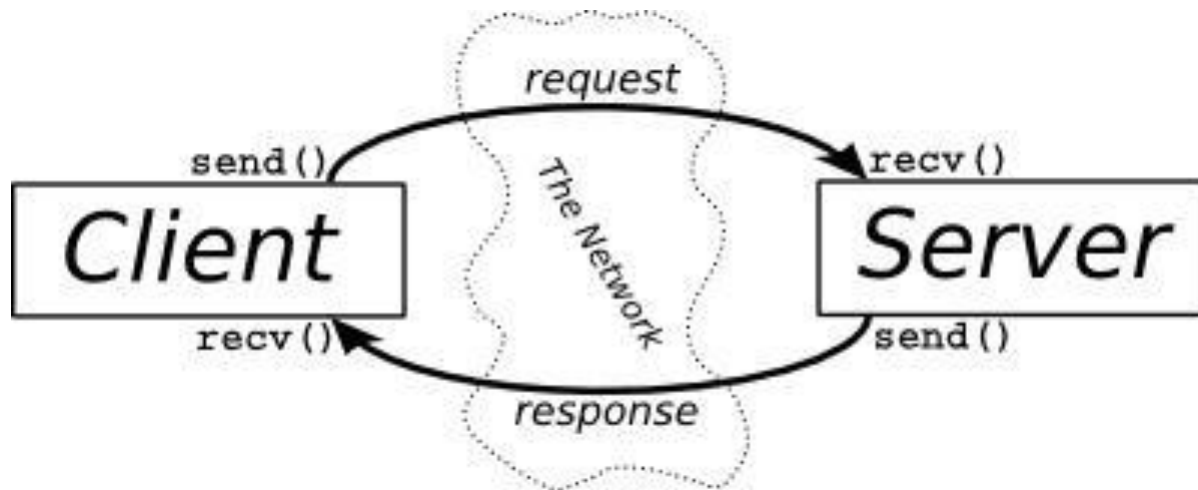


Leader.us

2016年版教程

# 网络编程篇

# Java网络编程入门



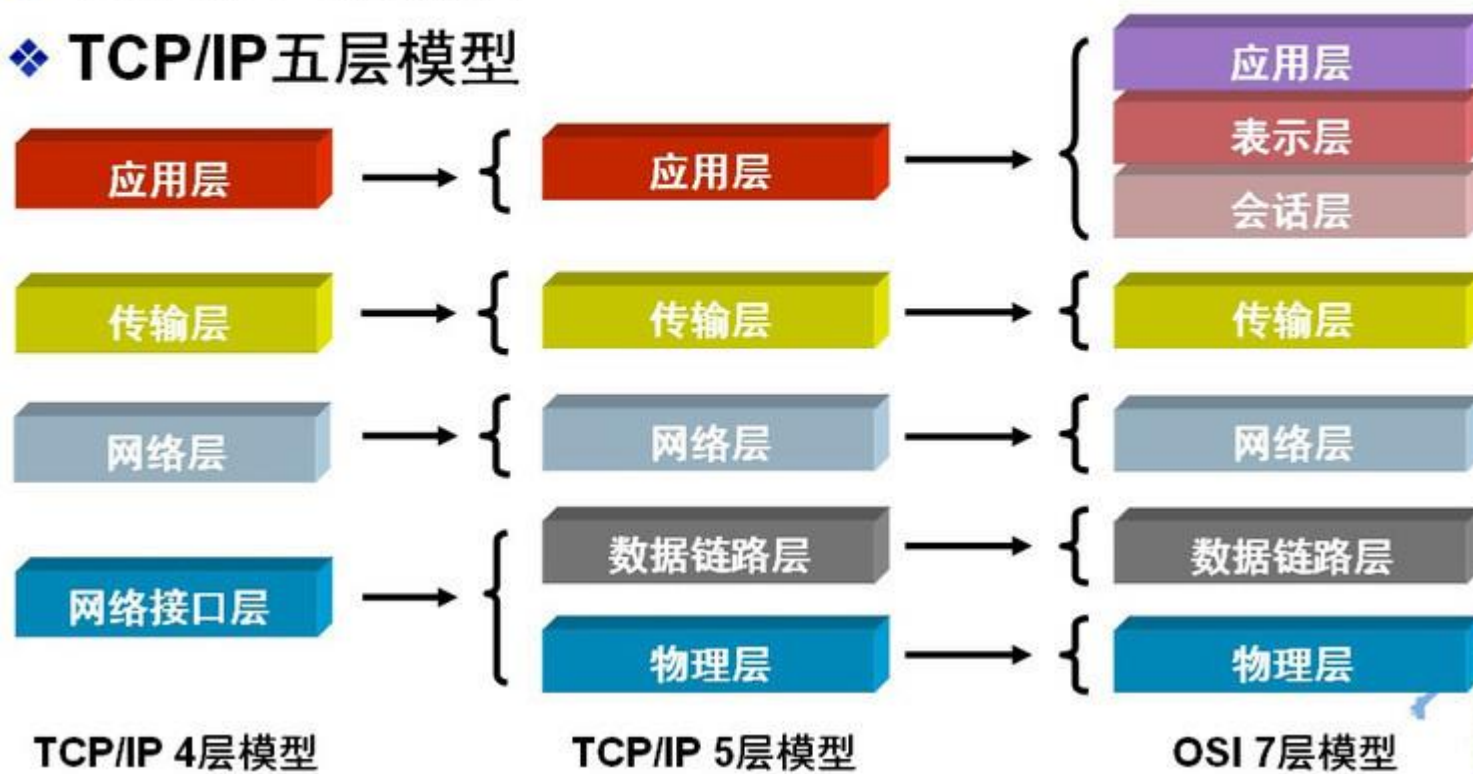
# TCP/IP编程基础.....

# TCP/IP模型

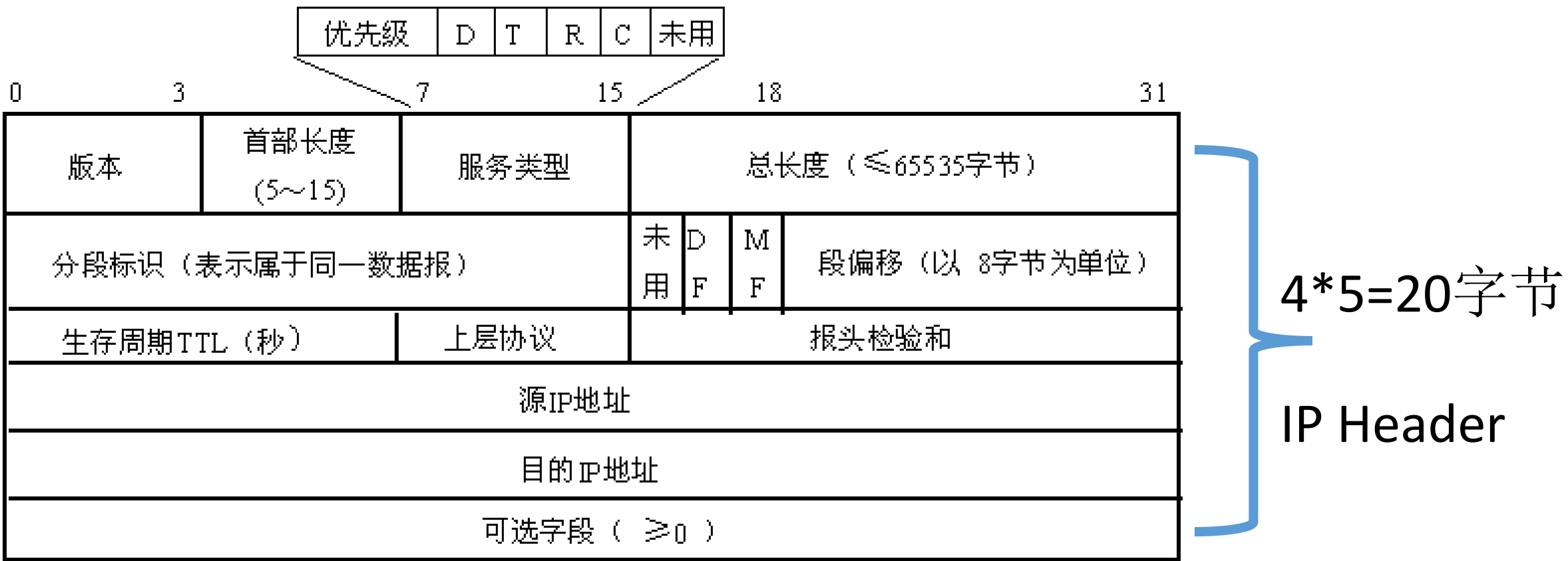
❖ OSI七层模型

❖ TCP/IP四层模型

❖ TCP/IP五层模型



# IP报文结构



DF: 是否分段; MF: 是否有后续分段

以太网的MTU最大传输单元是1500, IP报文小于这个数字就无需分段了

报文经过多少跳的路由, 从哪个IP发往哪个IP, 报文有多大, 优先级高还是低

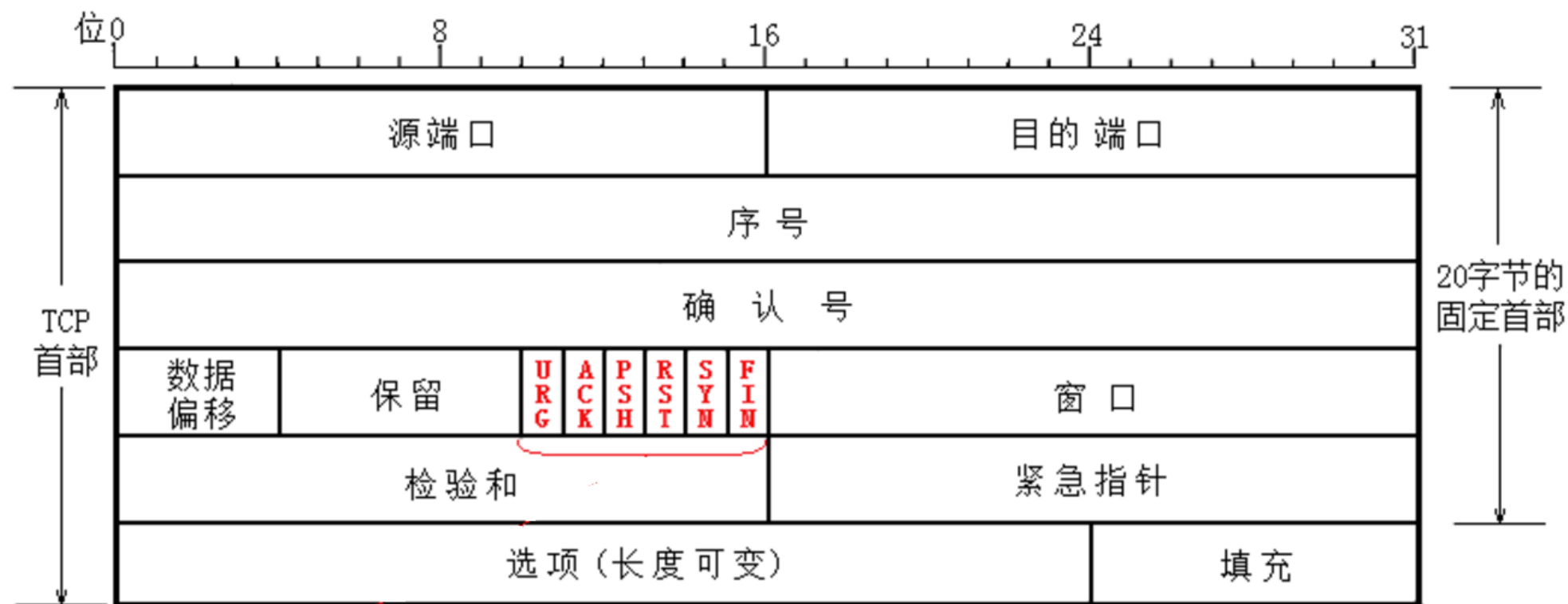


# TCP报文结构

IP Header (20字节)

TCP Header (20字节)

User Data



从什么端口发给什么端口，报文序号，TCP连接标志位，滑动窗口大小等

# UDP报文结构

IP Header (20字节)

UDP Header (8字节)

*User Data*



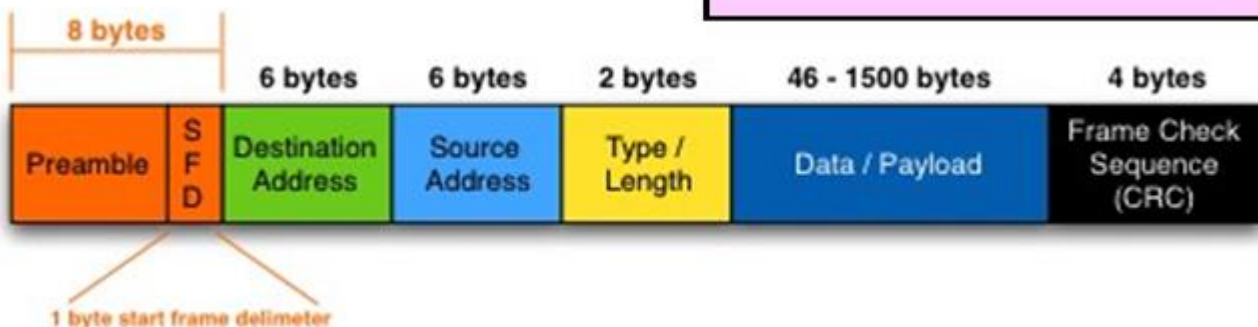
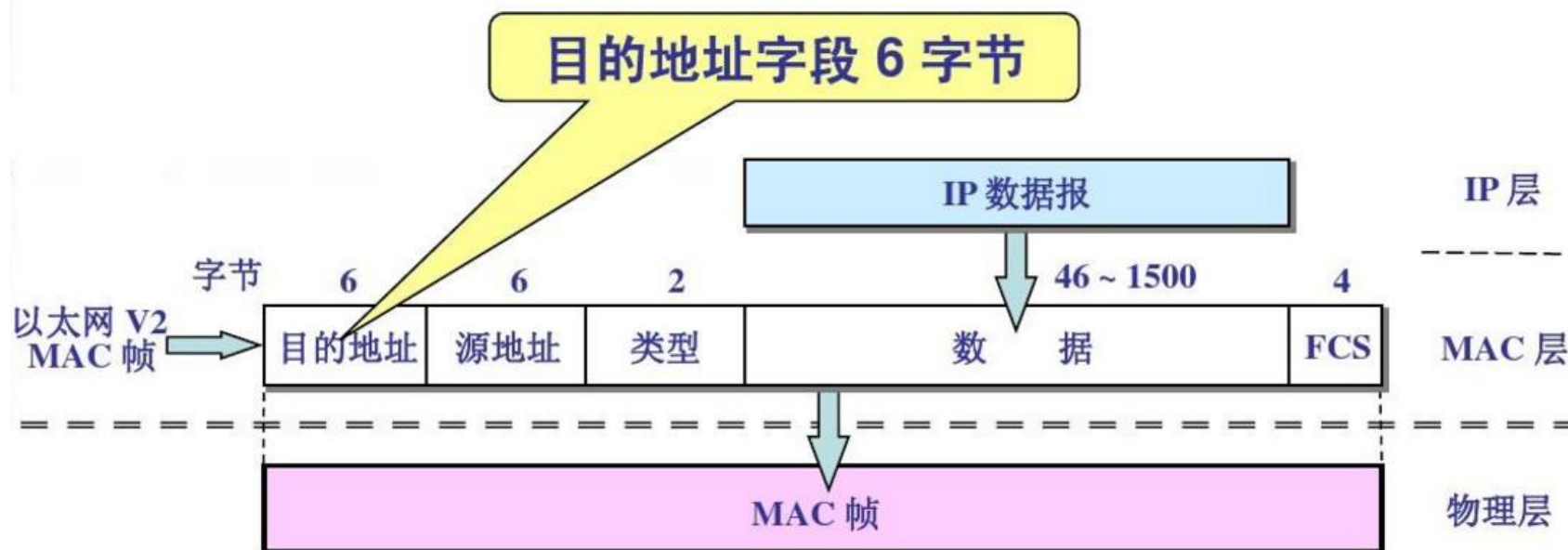
从什么端口发给什么端口，不管发送过的报文是否被确认，一个报文就是一个消息

# 还有一个以太帧...

EthernetII Header (22字节)

IP Header (20字节)

TCP/UDP Header



从哪个Mac地址到哪个Mac地址



# 网络报文设计的特点

分层叠加，每一层都实现一种特殊协议，类似装饰者模型

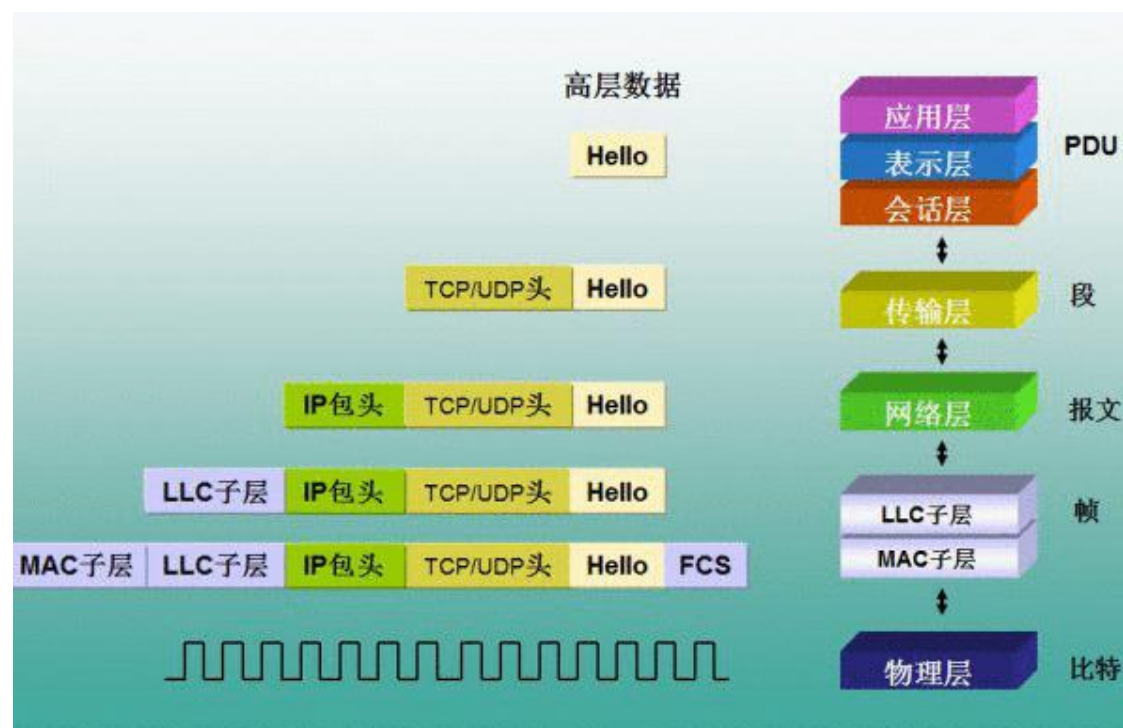
EthernetII Header

*LLC Header*

IP Header

TCP Header

*User Data*



# TCP/UDP报文总结

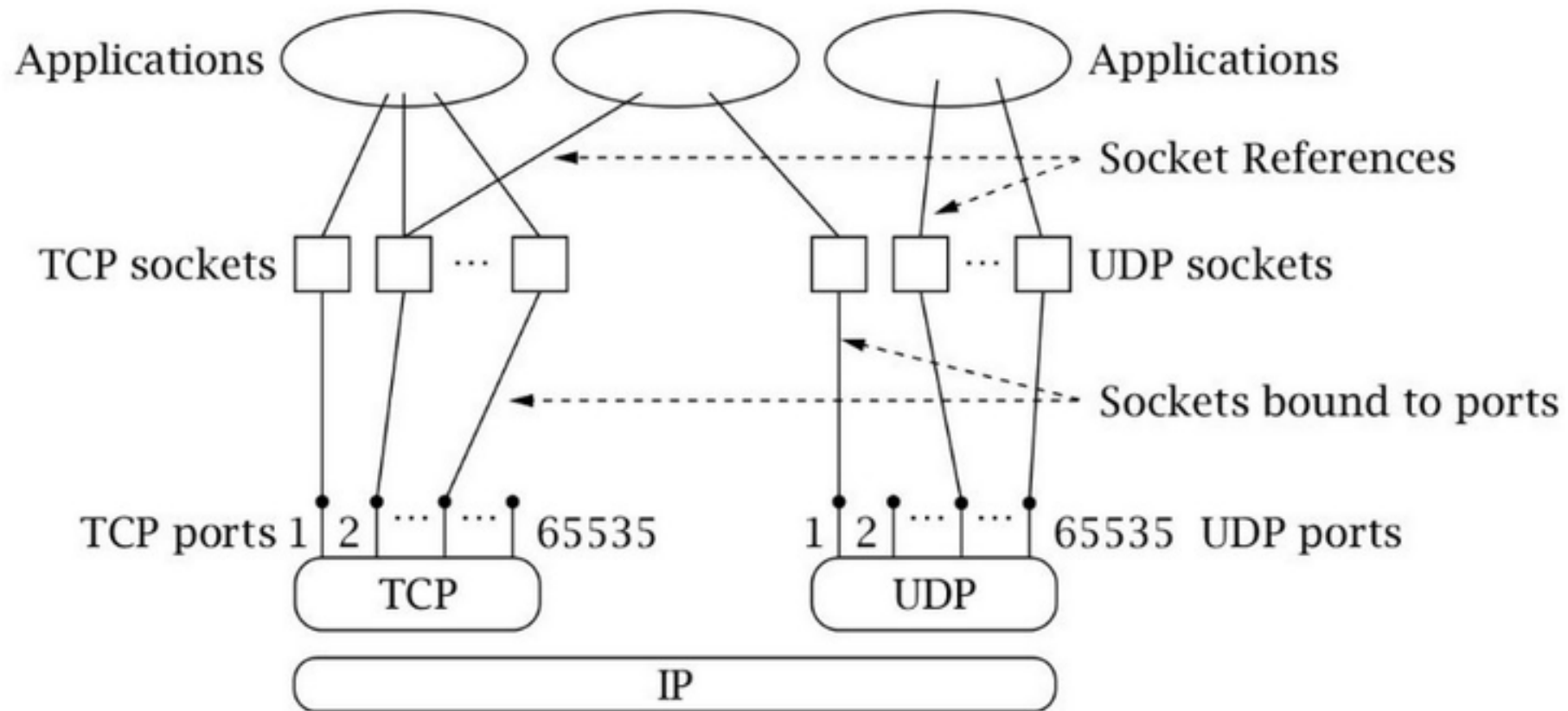
不管上层协议如何，底层传输还是一个报文一个报文传输的，每个报文的数据也就最多1500字节而已。

学以致用  
机智如我

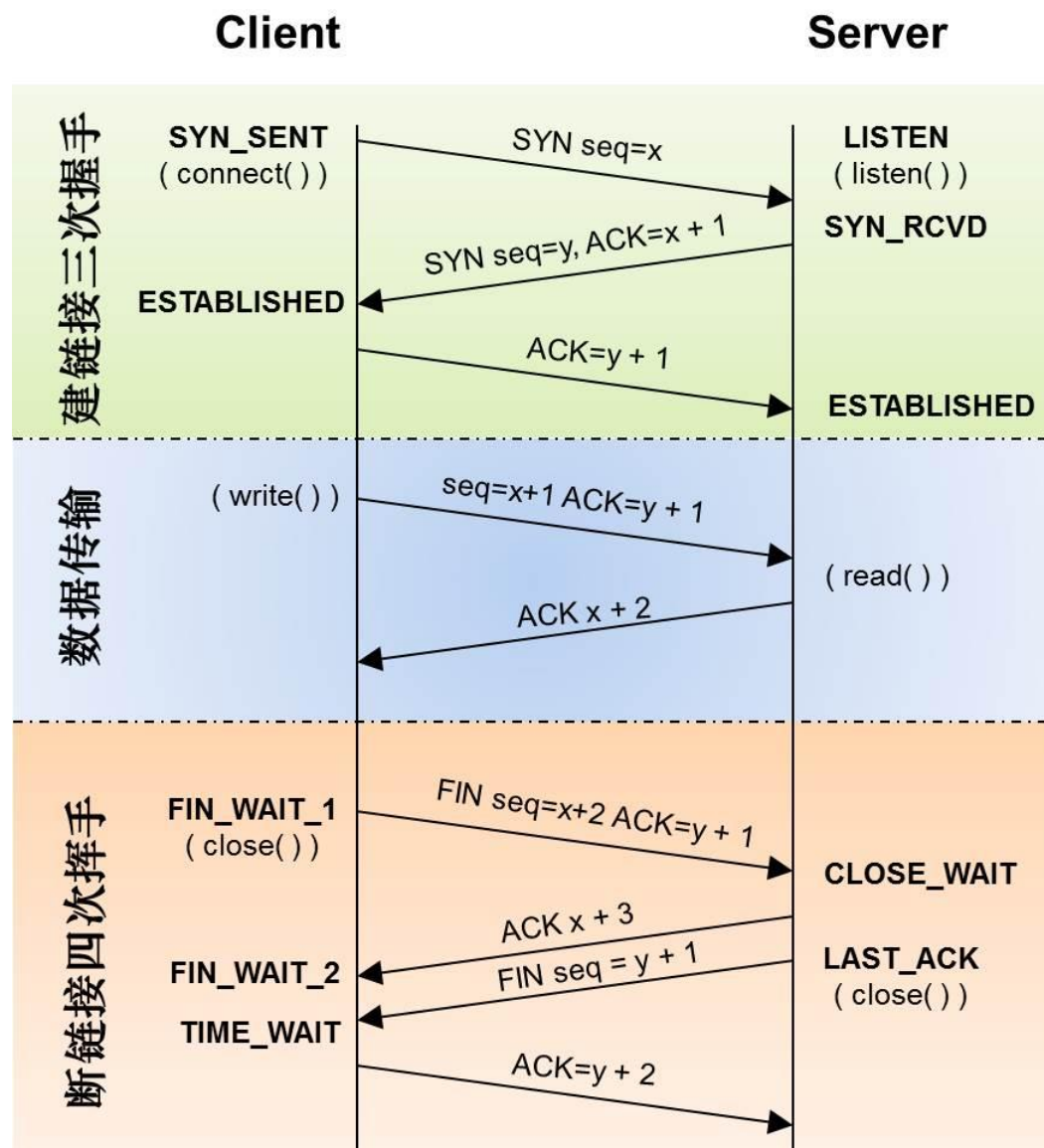
所以，你的一封情谊绵绵的万言情书实际上是分为十几个网络报文发送出去的，考虑到网络的不可靠性，建议情书的章节目录要“先说重点”，重要的表白建议重复3遍，然后再慢慢编造各种花言巧语的陈述或者感言。。。让对方来不及思考。。。



# TCP/UDP编程模型（一）



# TCP/UDP编程模型（二）



TCP编程，哥看一眼就走了....

- 建立连接的过程有点损耗时间
- 适合长时间传输数据
- 需要定义一种应用层的报文协议
- 可能需要复用连接（连接池机制）
- 要有合适的心跳机制验证连接正常
- 服务端要承受大量的客户端连接
- 处理意外断链事件
- 需要具备重连机制

.....

描线懵逼



# TCP/UDP编程模型（三）

**TCP是面向连接的网络通讯模型**

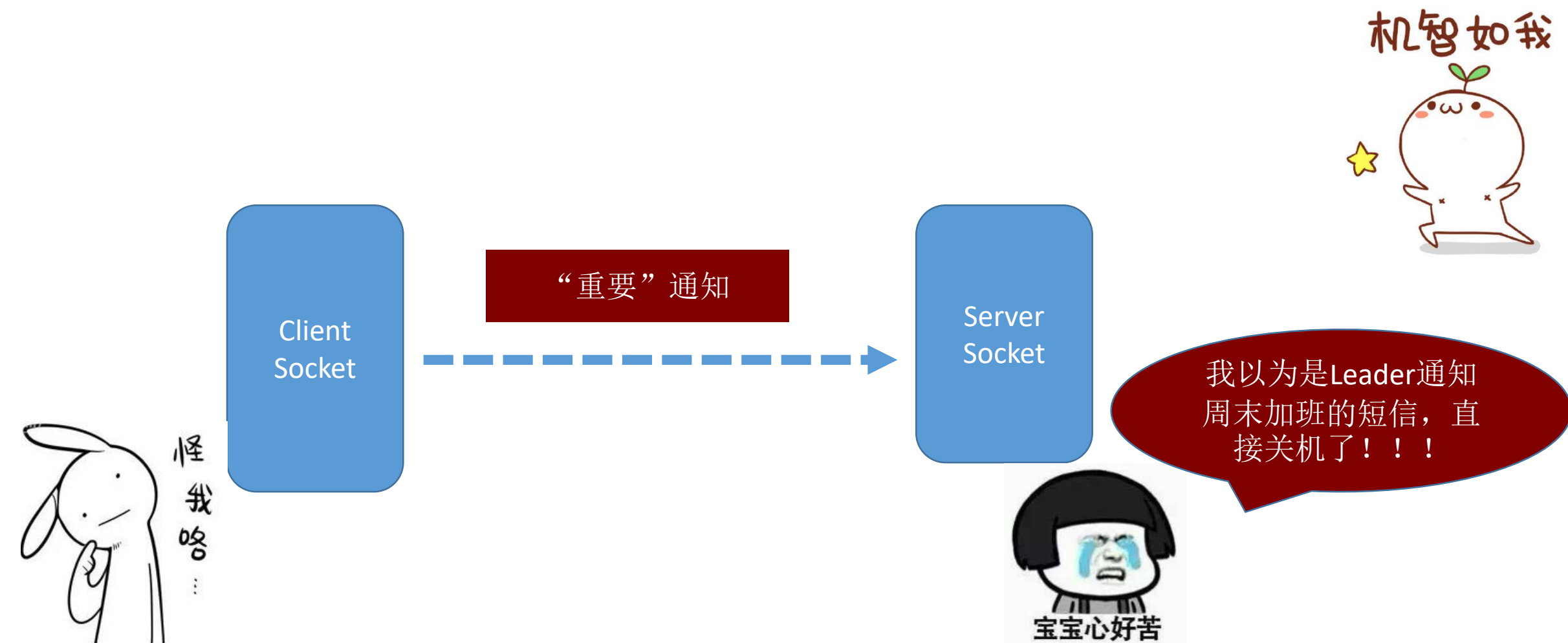
**TCP通讯屏蔽了网络报文的概念，提供了Stream的字节流传输管道**



## ReadBuffer & WriteBuffer

# TCP/UDP编程模型（四）

UDP倒是很简单：“明天加班来公司，人人发1万现金，没收到通知的，只怪命不好，哥不負責任！！（妹子发3遍）”





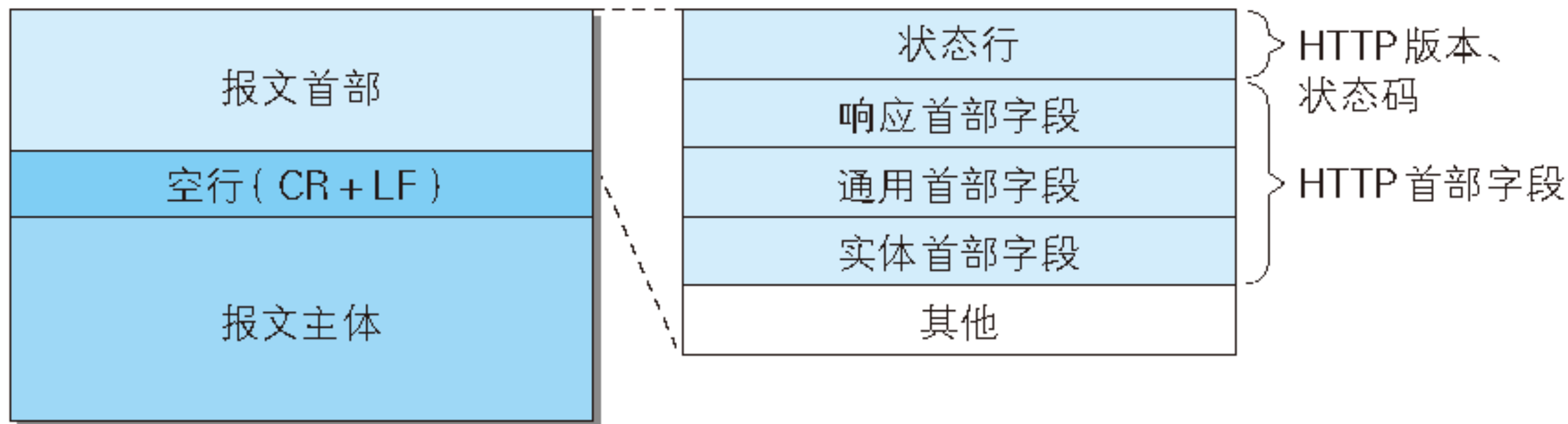
# DIY HTTP Server.....

# HTTP的报文结构（上）

TCP Header

HTTP Data

HTTP采用文本协议（ASCII字符）编码，文本行模式



# HTTP的报文结构（中）

HTTP请求报文协议格式 - Linux大棚版						
方法	空格	URL	空格	版本	回车符	换行符
头部域名称		:	头部域值		回车符	换行符
....						
头部域名称		:	头部域值		回车符	换行符
回车符	换行符					
附属体						

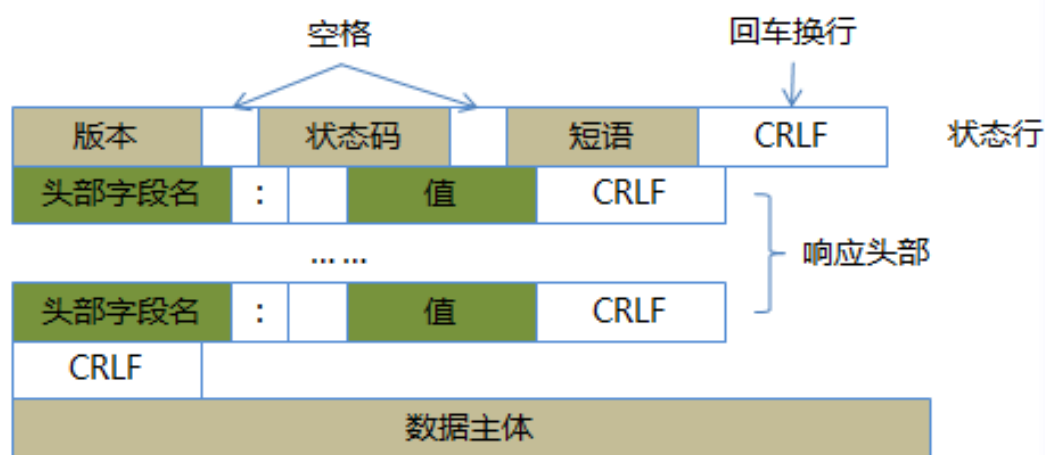
request line  
(GET, POST, HEAD commands)

GET /somedir/page.html HTTP/1.1  
 Host: www.someschool.edu  
 User-agent: Mozilla/4.0  
 Connection: close  
 Accept-language: fr

header lines

Carriage return, line feed indicates end of message  
 (extra carriage return, line feed)

# HTTP的报文结构（下）



HTTP响应报文格式



# HTTP & Base64

为什么会有Base64编码呢？因为有些网络传送渠道并不支持所有的字节，例如传统的邮件只支持可见字符的传送，像ASCII码的控制字符就不能通过邮件传送。这样用途就受到了很大的限制，比如图片二进制流的每个字节不可能全部是可见字符，所以就传送不了。最好的方法就是在不改变传统协议的情况下，做一种扩展方案来支持二进制文件的传送。把不可打印的字符也能用可打印字符来表示，问题就解决了。Base64编码应运而生，**Base64就是一种基于64个可打印字符来表示二进制数据的表示方法。Base64编码主要用在传输、存储、表示二进制等领域，还可以用来加密**

Base64要求把每三个8Bit的字节转换为四个6Bit的字节（ $3 \times 8 = 4 \times 6 = 24$ ），然后把6Bit再添两位高位0，组成四个8Bit的字节，也就是说，转换后的字符串理论上将要比原来的长1/3，此外，每76个字符加一个换行符。编码后的数据是一个字符串，其中包含的字符为：A-Z、a-z、0-9、+、/等共64个字符： $26 + 26 + 10 + 1 + 1 = 64$

其实迅雷的“专用地址”也是用Base64"加密"的，其过程如下：

一、在地址的前后分别添加AA和ZZ

二、对新的字符串进行Base64编码

另：Flashget的与迅雷类似，只不过在第一步时加的“料”不同罢了，

Flashget在地址前后加的“料”是[FLASHGET]

而QQ旋风的干脆不加料，直接就对地址进行Base64编码了





Expert Ja  
Leader.



只要网络学的好，明年也有美娇娘！



A young woman with long brown hair, wearing a white dress with a pink and green floral pattern, is sitting on a large, flat, grey rock. She is barefoot and holding an open book with both hands, looking down at the pages. The rock is surrounded by some green moss and small plants. The background is a dark, rocky area with some greenery.

许多年以后，哥成了一个传说

**梦醒了...开始编程了.....**

先做个小练习：  
开发个简易版**Web Server**

...

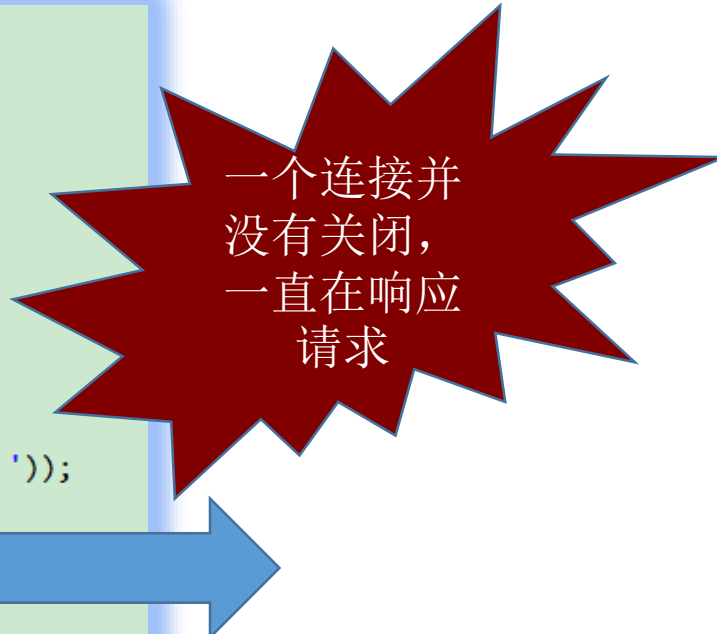


# 开发服务端代码

- HTTP/1.0协议使用非持久连接,即在非持久连接下,一个tcp连接只传输一个Web对象
- HTTP/1.1默认使用持久连接

```
try {
    ServerSocket serverSocket = new ServerSocket(80);
    while (true) {

        Socket socket = serverSocket.accept();
        System.out.println("Request: " + socket.toString() + " connected");
        LineNumberReader in = new LineNumberReader(new InputStreamReader(socket.getInputStream()));
        String lineInput;
        String requestPage = null;
        while ((lineInput = in.readLine()) != null) {
            System.out.println(lineInput);
            if (in.getLineNumber() == 1) {
                requestPage = lineInput.substring(lineInput.indexOf('/') + 1, lineInput.lastIndexOf(' '));
                System.out.println("request page :" + requestPage);
            } else {
                if (lineInput.isEmpty()) {
                    System.out.println("header finished");
                    doResponseGet(requestPage, socket);
                }
            }
        }
    }
} catch (IOException e) {
```



一个连接并没有关闭，  
一直在响应请求

# 开发服务端代码

```
private static void doResponseGet(String requestPage, Socket socket) throws IOException {  
    final String WEB_ROOT = "c:";  
    File theFile = new File(WEB_ROOT, requestPage);  
    OutputStream out = socket.getOutputStream();  
    if (theFile.exists()) {  
        // 从服务器根目录下找到用户请求的文件并发送回浏览器  
        InputStream fileIn = new FileInputStream(theFile);  
        byte[] buf = new byte[fileIn.available()];  
        fileIn.read(buf);  
        fileIn.close();  
        out.write(buf);  
        out.flush();  
        socket.close();  
        System.out.println("request complete.");  
    } else {  
        String msg = " I can't find bao zang...cry..\r\n";  
        String response = "HTTP/1.1 200 OK\r\n";  
        response += "Server: Leader Server/0.1\r\n";  
        response += "Content-Length: " + (msg.length() - 4) + "\r\n";  
        response += "\r\n";  
        response += msg;  
        out.write(response.getBytes());  
        out.flush();  
    }  
}
```



你让我说点什么好.....




# 开发服务端代码

```
Request: Socket[addr=/0:0:0:0:0:0:0:1,port=56588,localport=80] connected
GET /adsfsdf HTTP/1.1
request page :adsfsdf
Host: localhost
Connection: keep-alive
Cache-Control: max-age=0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.8

header finished
GET /adsfsdf HTTP/1.1
Host: localhost
Connection: keep-alive
Cache-Control: max-age=0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.8

header finished
GET /adsfsdf HTTP/1.1
Host: localhost
Connection: keep-alive
Cache-Control: max-age=0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.8
```



传说中的多  
路复用，我  
竟然懂了

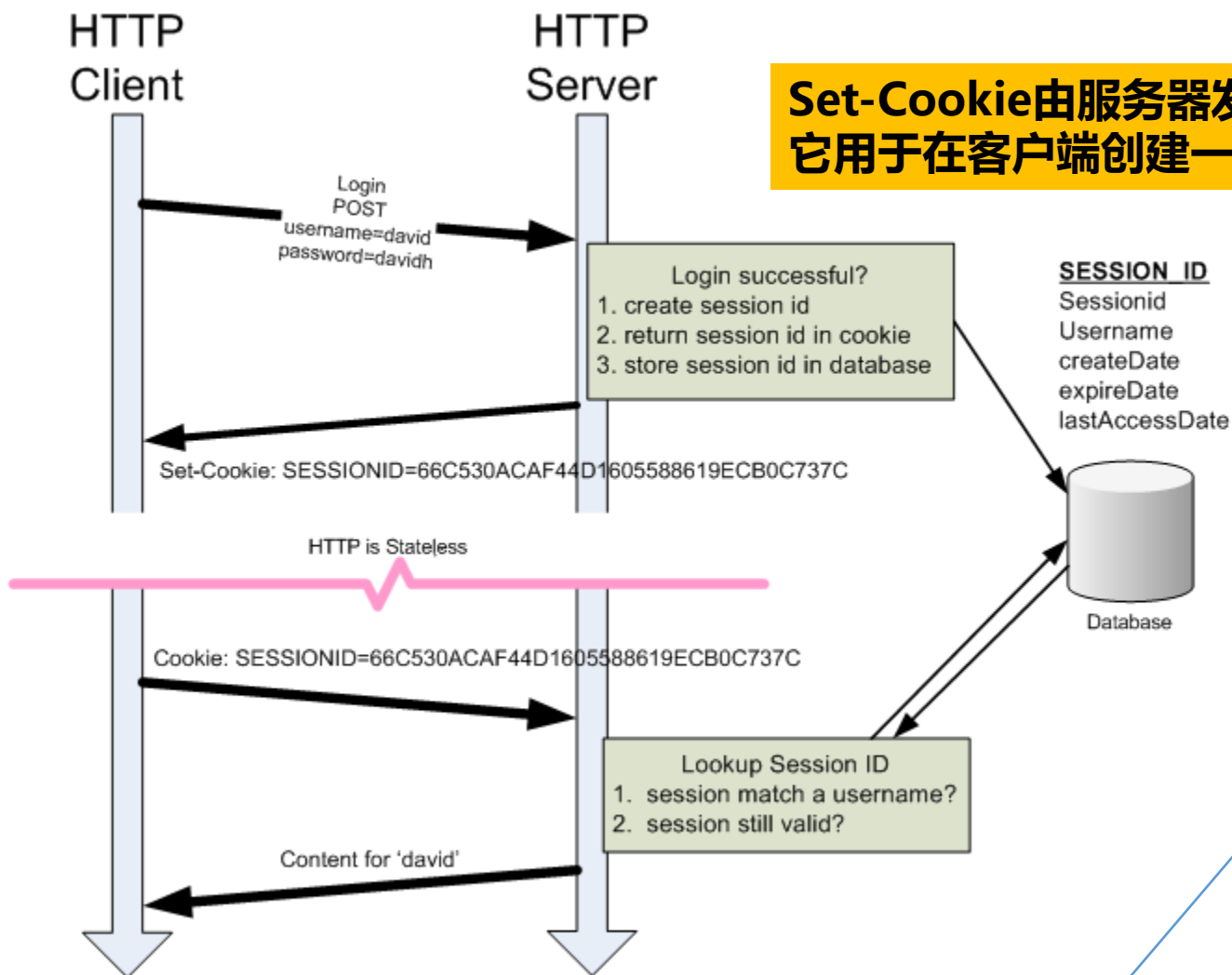


# HTTP会话机制（一）

HTTP协议是无状态的协议。一旦数据交换完毕，客户端与服务器端的TCP连接就可以关闭，再次交换数据需要建立新的TCP连接。这就意味着服务器无法从连接上跟踪会话。解决办法：服务器端鉴定客户端身份并为它生成一个唯一标识的ID，通过HTTP Header里的Cookie传递给客户端（浏览器），浏览器将Cookie保持到本地的文件中，下次请求的时候，自动带上Cookie信息，服务端就能跟踪识别客户身份了。



# HTTP会话机制（二）



**Set-Cookie**由服务器发送，它包含在响应请求的头部中。  
它用于在客户端创建一个Cookie

Java用了jsessionid这个cookie条目来表示会话ID  
tomcat：长度为32个字节的随机字符串  
weblogic：长度为52个字节

~~domain:baidu.com  
request url: http://news.sina.com~~

domain:baidu.com  
request url: http://news.baidu.com  
request url: http://images.baidu.com

**Cookie头由客户端发送，只有cookie的domain和path与请求的URL匹配才会发送这个cookie**

# HTTP会话机制（三）

```
Set-Cookie: <name>=<value>[; <name>=<value>]...  
            [; expires=<date>][; domain=<domain_name>]  
            [; path=<some_path>][; secure][; httponly]
```

cookie中设置了HttpOnly属性，那么通过js脚本将无法读取到cookie信息，这样能有效的防止XSS攻击，只能阻击小白黑客，servlet3.0以前不能添加httpOnly属性，secure表示创建的Cookie只能以HTTPS的连接传递到服务器端

- IE :原先为20个，后来升级为50个
- Firefox: 50个
- Opera:30个
- Chrome:180个
- Safari:无限制

One  
domain

- Firefox和Safari:4079字节
- Opera:4096字节
- IE:4095字节

One  
Cookie

# HTTP会话机制（四）

```
String userInfo=null;
while ((lineInput = in.readLine()) != null) {
    System.out.println(lineInput);

    if (in.getLineNumber() == 1) {
        requestPage = lineInput.substring(lineInput.indexOf('/') + 1, lineInput.lastIndexOf(' '));
        System.out.println("request page : " + requestPage);
    } else {
        if (lineInput.startsWith("Cookie: "))
        {
            userInfo=lineInput;
            System.out.println("new User "+lineInput);
        }
        else if (lineInput.isEmpty()) {
            System.out.println("header finished");
            doResponseGet(userInfo,requestPage, socket);
        }
    }
}
```

Cookie与  
Session模拟

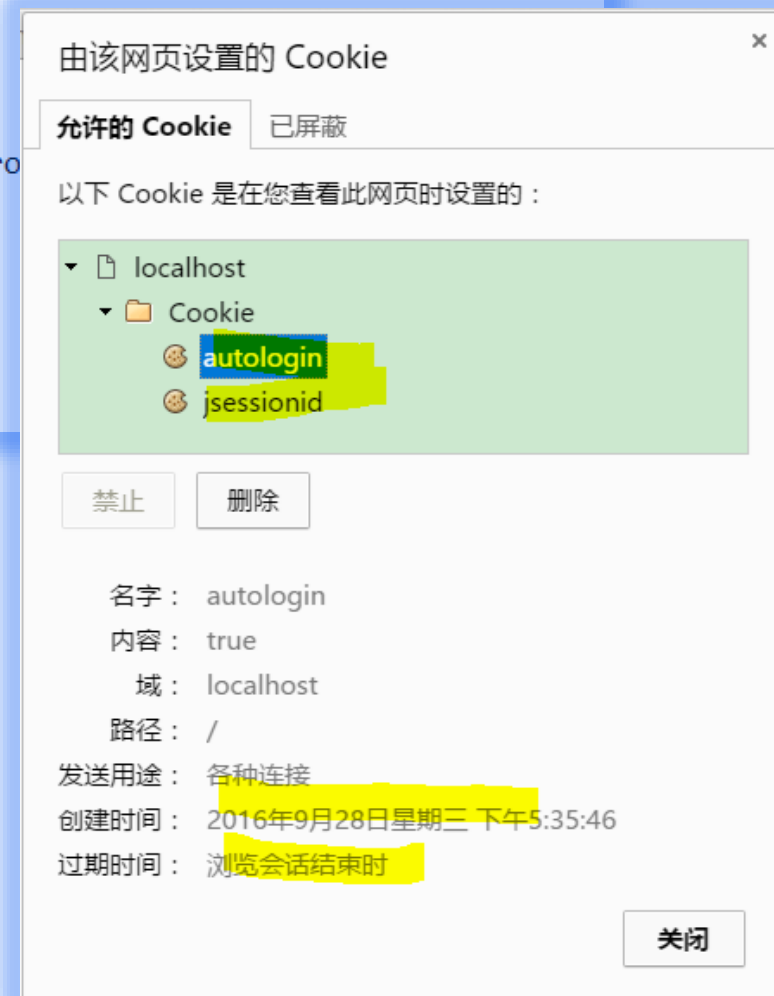
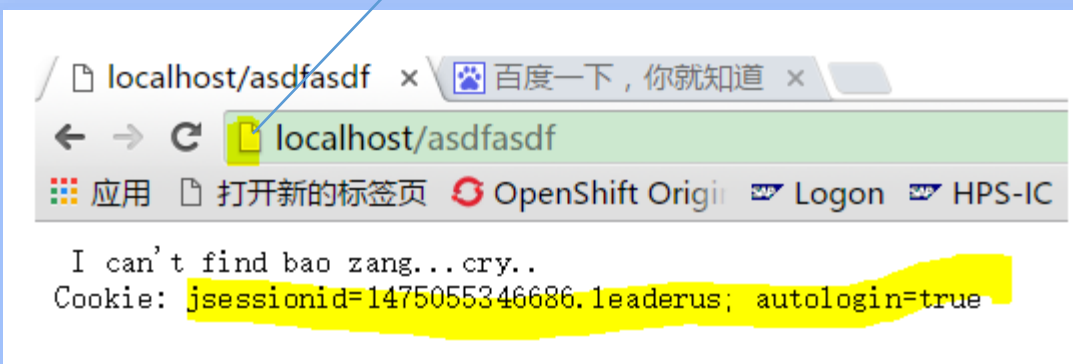
```
String msg=" I can't find bao zang...cry.. \r\n"+((userInfo==null)?" new user ...":userInfo);
String response = "HTTP/1.1 200 OK\r\n";
response += "Server: Leader Server/0.1\r\n";
if(userInfo==null)
{
    response+=genCookieHeader();
}
response += "Content-Length: "+msg.length()+"\r\n";
response += "\r\n";
response+=msg;
out.write(response.getBytes());
out.flush();
```

```
private static String genCookieHeader()
{
    String header="Set-Cookie: jsessionid="+System.currentTimeMillis()+".leaderus; domain=localhost+"\r\n";
    header+="Set-Cookie: autologin=true; domain=localhost+"\r\n";
    return header;
}
```

# HTTP会话机制（五）

```
header finished
GET /favicon.ico HTTP/1.1
Host: localhost
Connection: keep-alive
Pragma: no-cache
Cache-Control: no-cache
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/54.0.2840.99 Safari/537.36
Accept: */*
Referer: http://localhost/asdfasdf
Accept-Encoding: gzip, deflate, sdch
Accept-Language: zh-CN,zh;q=0.8
Cookie: jsessionid=1475055346686.leaderus; autologin=true
new User Cookie: jsessionid=1475055346686.leaderus; autologin=true
```

## 查看网页的Cookie



你可以完成更多.....



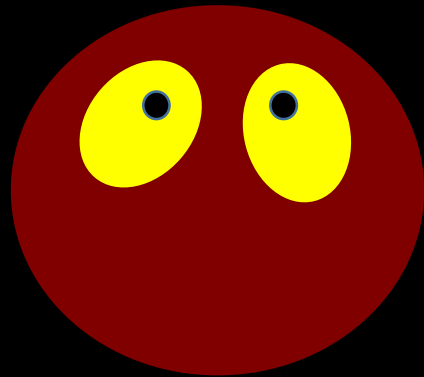
# 用NIO实现一个HTTP Server

**<https://github.com/NanoHttpd/nanohttpd>**

## Core

- Only one Java file, providing HTTP 1.1 support.
- No fixed config files, logging, authorization etc. (Implement by yourself if you need them. Errors are passed to `java.util.logging`, though.)
- Support for HTTPS (SSL).
- Basic support for cookies.
- Supports parameter parsing of GET and POST methods.
- Some built-in support for HEAD, POST and DELETE requests. You can easily implement/customize any HTTP method, though.
- Supports file upload. Uses memory for small uploads, temp files for large ones.
- Never caches anything.
- Does not limit bandwidth, request time or simultaneous connections by default.
- All header names are converted to lower case so they don't vary between browsers/clients.
- Persistent connections (Connection "keep-alive") support allowing multiple requests to be served over a single socket connection.

还是赶紧做Leader作业吧.....



To Be Continued