马哥教育
IT人的高薪职业学院

HTTP协议和APACHE

APACHE
HTTP SERVER PROJECT

讲师：王晓春

本章内容

- Internet
- SOCKET概念
- http协议
- Httpd介绍
- Httpd配置
- http报文格式
- 编译安装httpd

互联网Internet

Internet与中国

- Internet最早来源于美国国防部高级研究计划局ARPA建立的ARPANet，1969年投入运行。1983年，ARPAnet分裂为两部分：ARPAnet和纯军事用的MILNET。当年1月，ARPA把TCP/IP协议作为ARPAnet的标准协议，这个以ARPAnet为主干网的网际互联网便被称为Internet。1986年，美国国家科学基金会建立计算机通信网络NSFnet。此后，NSFNet逐渐取代ARPANet在Internet的地位。1990年，ARPANet正式关闭
- 北京时间1987年9月20日，钱天白建立起一个网络节点，通过电话拨号连接到国际互联网，向他的德国朋友发出来自中国的第一封电子邮件：Across the Great Wall we can reach every corner in the world，自此，中国与国际计算机网络开始连接在一起

Internet与中国

(Message # 50: 1532 bytes, KEEP, Forwarded)
Received: from unika1 by irau11.germany.csnet id aa21216; 20 Sep 87 17:36 MET
Received: from Peking by unika1; Sun, 20 Sep 87 16:55 (MET dst)
Date: Mon, 14 Sep 87 21:07 China Time
From: Mail Administration for China <MAIL@ze1>
To: Zorn@germany, Rotert@germany, Wacker@germany, Finken@unika1
CC: lhl@parmesan.wisc.edu, farber@udel.edu,
jennings%irlean.bitnet@germany, cic%relay.cs.net@germany, Wang@ze1,
RZLI@ze1
Subject: First Electronic Mail from China to Germany

"Ueber die Grosse Mauer erreichen wie alle Ecken der Welt"
"Across the Great Wall we can reach every corner in the world"
Dies ist die erste ELECTRONIC MAIL, die von China aus ueber Rechnerkopplung in die internationalen Wissenschaftsnetze geschickt wird.
This is the first ELECTRONIC MAIL supposed to be sent from China into the international scientific networks via computer interconnection between Beijing and Karlsruhe, West Germany (using CSNET/PMDF BS2000 Version).

University of Karlsruhe	Institute for Computer Application of
-Informatik Rechnerabteilung-	State Commission of Machine Industry
(IRA)	(ICA)
Prof. Werner Zorn	Prof. Wang Yuen Fung
Michael Finken	Dr. Li Cheng Chiung

Internet与中国

- 1990年10月

  钱天白教授代表中国正式在国际互联网络信息中心的前身DDN-NIC注册登记了我国的顶级域名CN，并且从此开通了使用中国顶级域名CN的国际电子邮件服务。由于当时中国尚未正式连入Internet，所以委托德国卡尔斯鲁厄大学运行CN域名服务器

- 1993年3月2日

  中国科学院高能物理研究所租用AT&T公司的国际卫星信道接入美国斯坦福线性加速器中心（SLAC）的64K专线正式开通,专线开通后，美国政府以Internet上有许多科技信息和其它各种资源，不能让社会主义国家接入为由，只允许这条专线进入美国能源网而不能连接到其它地方。尽管如此，这条专线仍是我国部分连入Internet的第一根专线

- 1994年4月20日

  中国通过一条64k的国际专线全功能接入国际互联网，成为国际互联网大家庭中的第77个成员，正式开启了互联网时代。随后，中科院高能物理研究所推出第一个WWW网站和第一套网页

- 1994年5月21日

  在钱天白教授和德国卡尔斯鲁厄大学的协助下，中国科学院计算机网络信息中心完成了中国国家顶级域名(CN)服务器的设置，改变了中国CN顶级域名服务器一直放在国外的历史

Internet与中国

- 1995年5月17日

  第27个世界电信日，邮电部正式宣布，向国内社会开放计算机互联网接入服务。

- 1995年5月

  北京的中关村南大街上出现了一块巨大的广告牌，“中国离信息高速公路还有多远？向北1500米。”那个位置就是一家叫“瀛海威”的网络科教馆，瀛海威正是information highway的音译，作为中国第一个互联网接入服务商，瀛海威几乎就是当时互联网的代名词

- 1996年1月

  中国互联网全国骨干网建成并正式开通，开始提供服务

- 1995年4月

  马云凑了两万块钱，成立杭州海博网络公司，专门给企业做主页

- 1997年5月

  丁磊创立网易

Internet与中国

- 1998年

  张朝阳创立搜狐。

- 1998年6月18日

  刘强东在中关村创办京东公司，代理销售光磁产品

- 1998年11月

  马化腾和张志东成立深圳市腾讯计算机系统有限公司，OICQ开通

- 1998年12月

  新浪网成立，关键人物：王志东

- 1999年5月18日

  中国第一家电子商务企业8848.com成立，创始人王峻涛也曾被誉为“中国电子商务教父”。2000年底，调查显示接近70%的人说上网买东西首选8848

- 2000年1月

  李彦宏创建了百度

Internet与中国

- 2003年5月
  阿里巴巴集团在创立淘宝网
- 2003年10月
  淘宝网首次推出支付宝服务
- 2004年1月
  京东多媒体网正式开通，启用域名www.jdlaser.com
- 2010年4月
  雷军创办小米
- 2011年1月21日
  腾讯公司推出微信 (WeChat)
- 2012年3月
  今日头条由张一鸣于创建
- 2012年7月10日
  北京小桔科技有限公司成立，滴滴司机端3个月后北京上线
- 下一个又是谁呢？

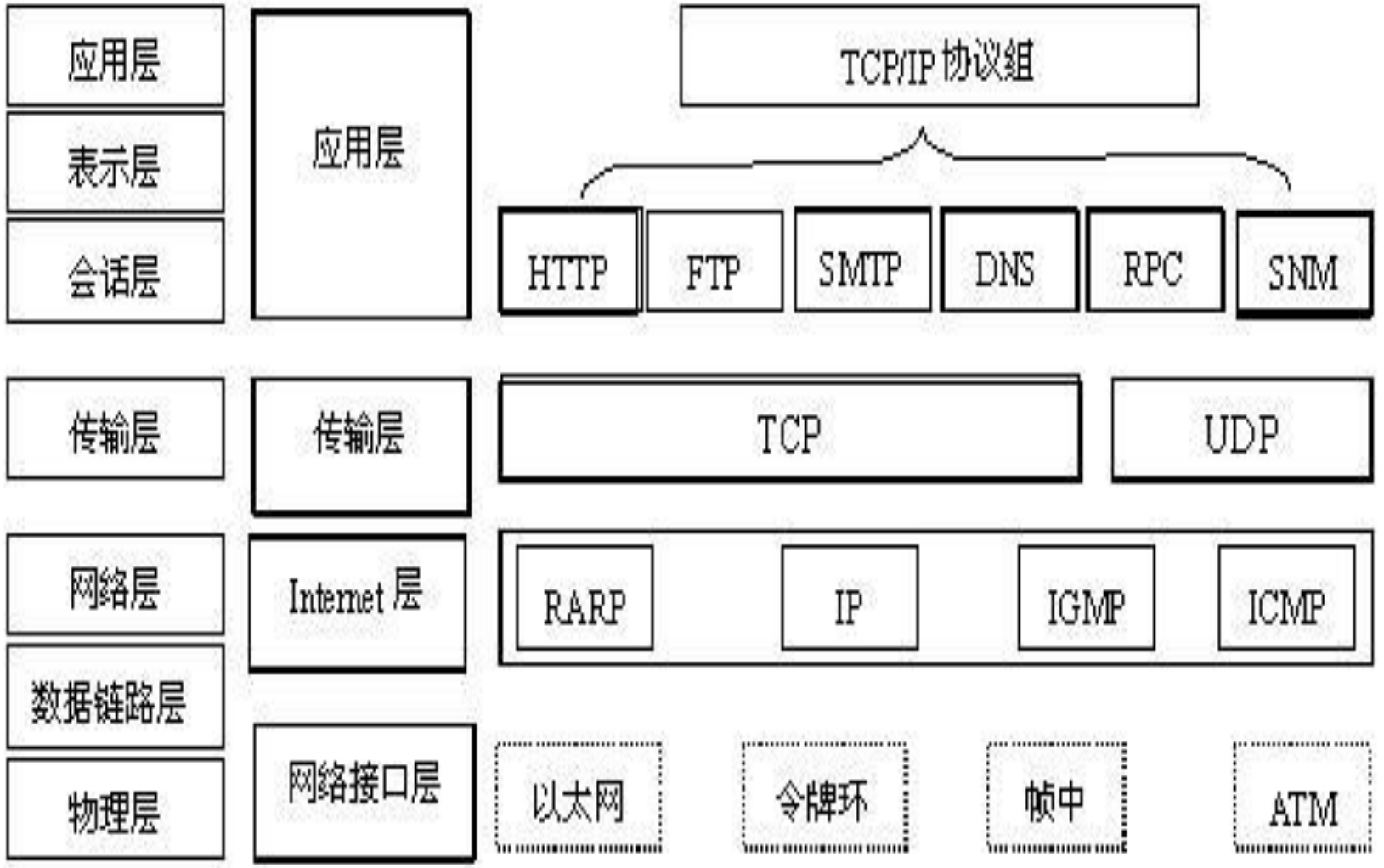
中国互联网连接世界

- 1885年台湾建省，首任巡抚刘铭传派人与福州船政联系，使用船政电报学堂毕业生为技术人员，于1887年铺设成功台湾淡水至福州川石海底电缆，全长117海里。这是我国自行设计安装的第一条海底电缆。此电缆毁于第二次世界大战
- 我国于1989年开始投入到全球海底光缆的投资与建设中来，并于1993年实现了首条国际海底光缆的登陆（中日之间C-J海底光缆系统）；随后在1997年，我国参与建设的全球海底光缆系统（FLAG）建成并投入运营，这也是第一条在我国登陆的洲际海底光缆
- 中国连接世界目前共有8条光缆，四个登陆站允许入境，目前我国的登陆站设立在三个城市的四个地区，分别是山东青岛登陆站（隶属中国联通）、上海崇明登陆站（隶属中国电信）、上海南汇登陆站（隶属中国联通）和广东汕头登陆站（隶属中国电信）

中国互联网连接世界

- 1987年9月20日，在北京ICA王运丰教授和西德卡尔斯鲁厄大学维尔纳·措恩教授的主导下，中华人民共和国大陆地区与外界互联网创建了首个连接。而中国第一封成功对外发出的电邮则是在1987年9月14日发出，内容为“Across the Great Wall, we can reach every corner in the world”（越过长城，走向世界每个角落）
- 然而，我们还是不能访问Google！

TCP/IP协议



跨网络的主机间通讯

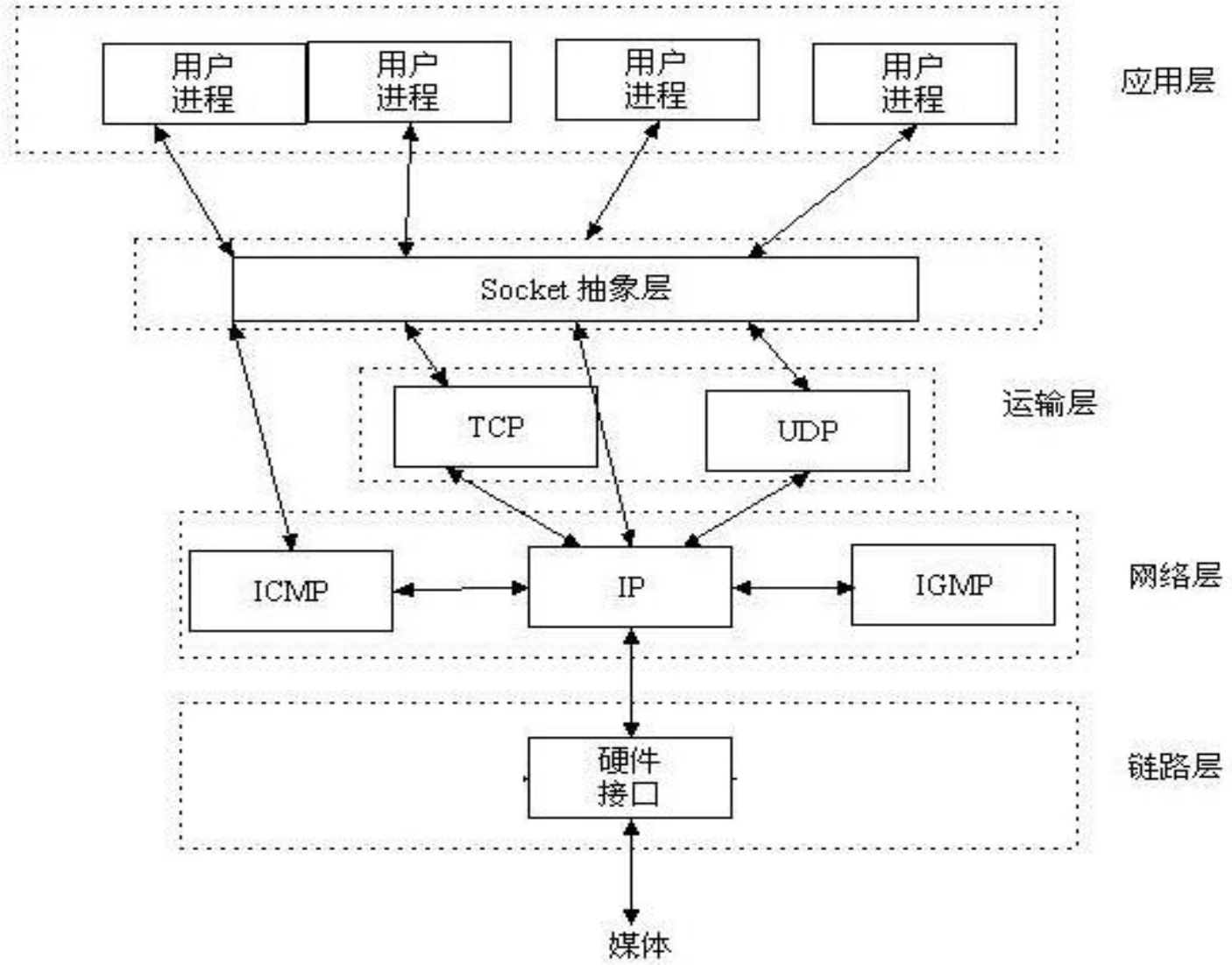
- 在建立通信连接的每一端，进程间的传输要有两个标志：
- IP地址和端口号，合称为套接字地址 socket address
- 客户机套接字地址定义了一个唯一的客户进程
- 服务器套接字地址定义了一个唯一的服务器进程

主机的IP地址　　进程的端口号

172.16.0.100　　53

套接字地址　172.16.0.100　53

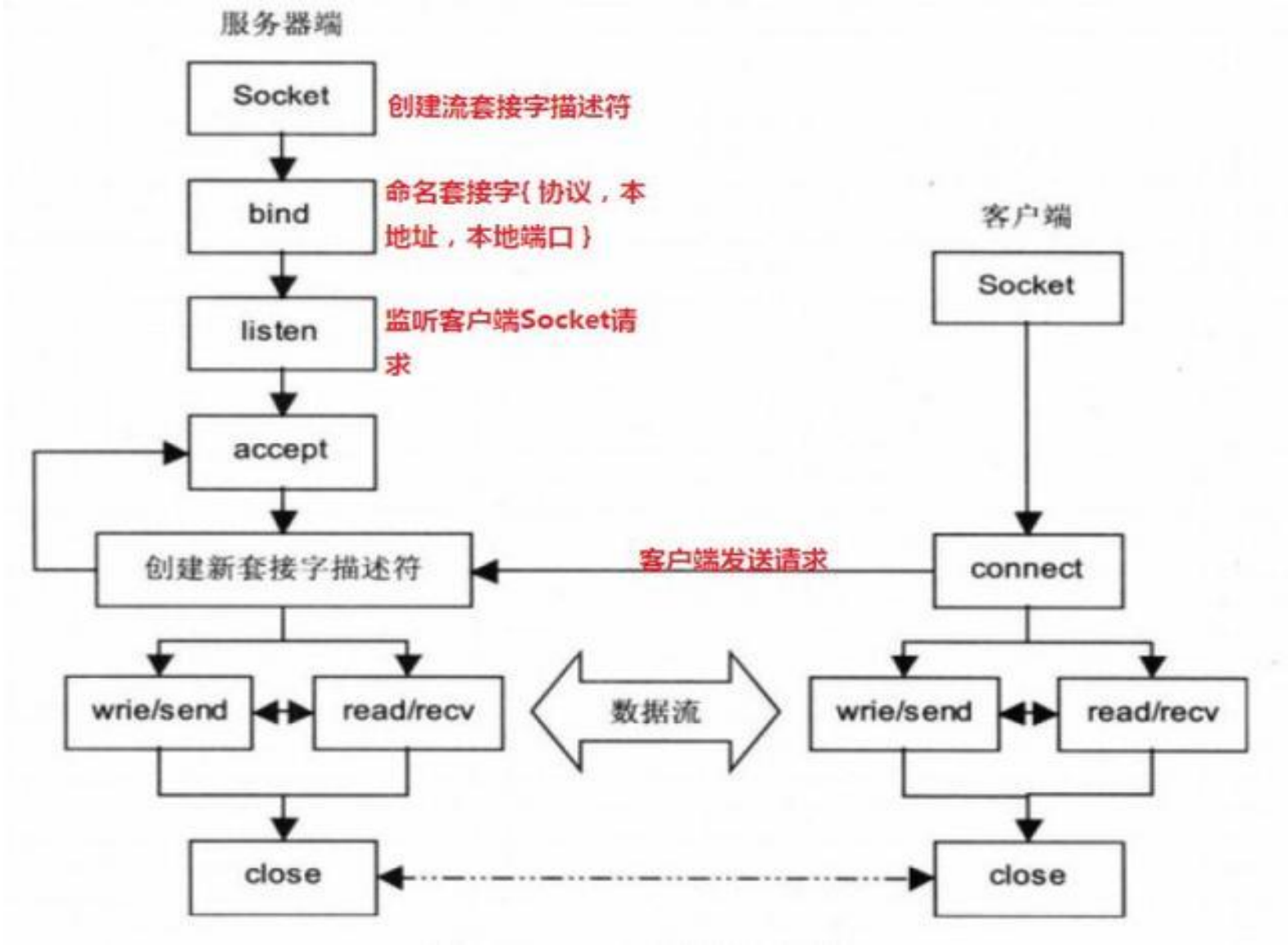
Socket套接字



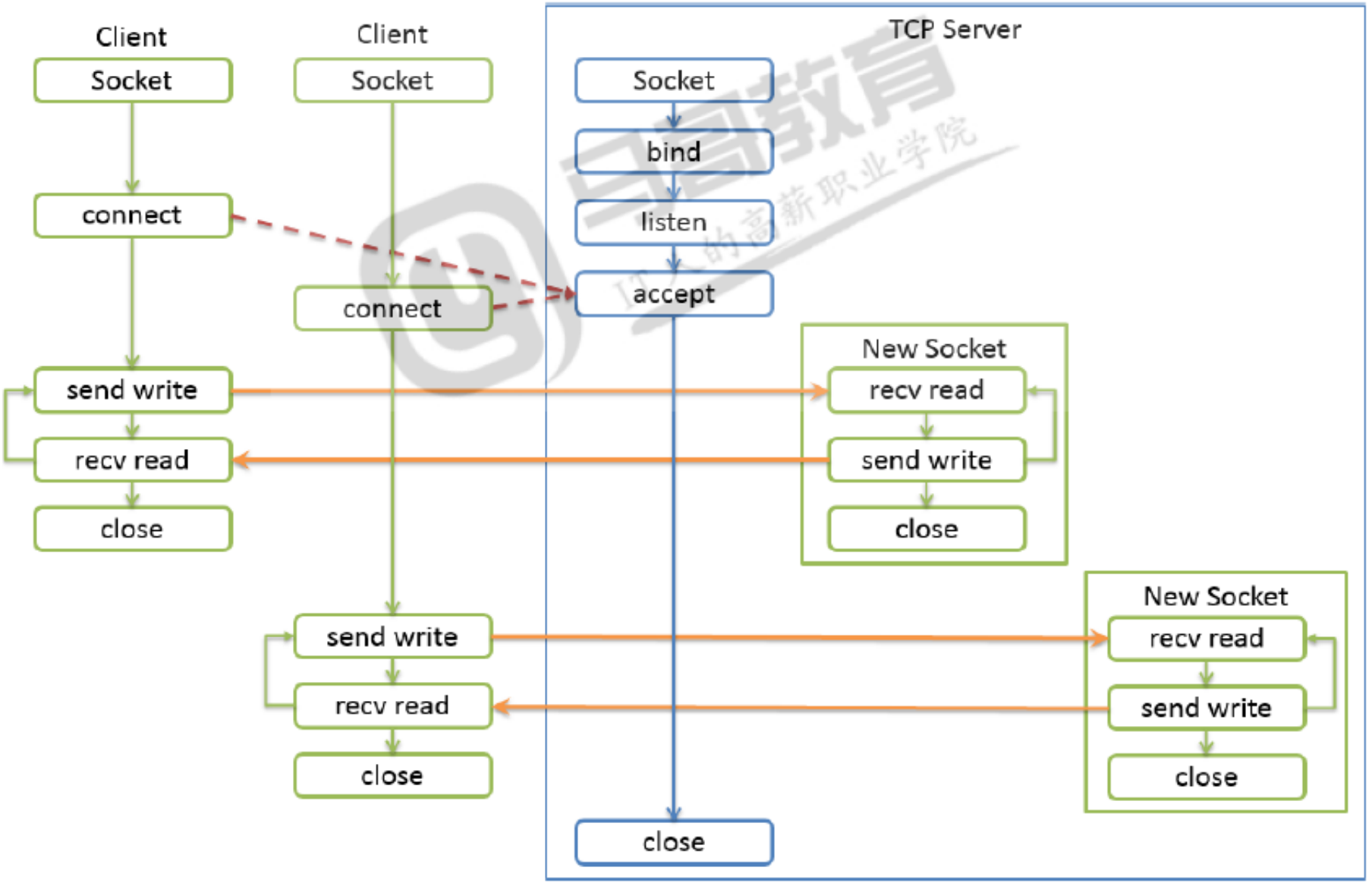
Socket套接字

- Socket:套接字，进程间通信IPC的一种实现，允许位于不同主机（或同一主机）上不同进程之间进行通信和数据交换，SocketAPI出现于1983年，4.2 BSD实现
- Socket API：封装了内核中所提供的socket通信相关的系统调用
- Socket Domain：根据其所使用的地址
  - AF_INET：Address Family，IPv4
  - AF_INET6：IPv6
  - AF_UNIX：同一主机上不同进程之间通信时使用
- Socket Type：根据使用的传输层协议
  - SOCK_STREAM：流，tcp套接字，可靠地传递、面向连接
  - SOCK_DGRAM：数据报，udp套接字，不可靠地传递、无连接
  - SOCK_RAW: 裸套接字,无须tcp或udp,APP直接通过IP包通信

客户/服务器程序的套接字函数



客户/服务器程序的套接字函数



系统调用

- 套接字相关的系统调用：

socket():	创建一个套接字
bind()：	绑定IP和端口
listen()：	监听
accept()：	接收请求
connect()：	请求连接建立
write()：	发送
read()：	接收
close():	关闭连接

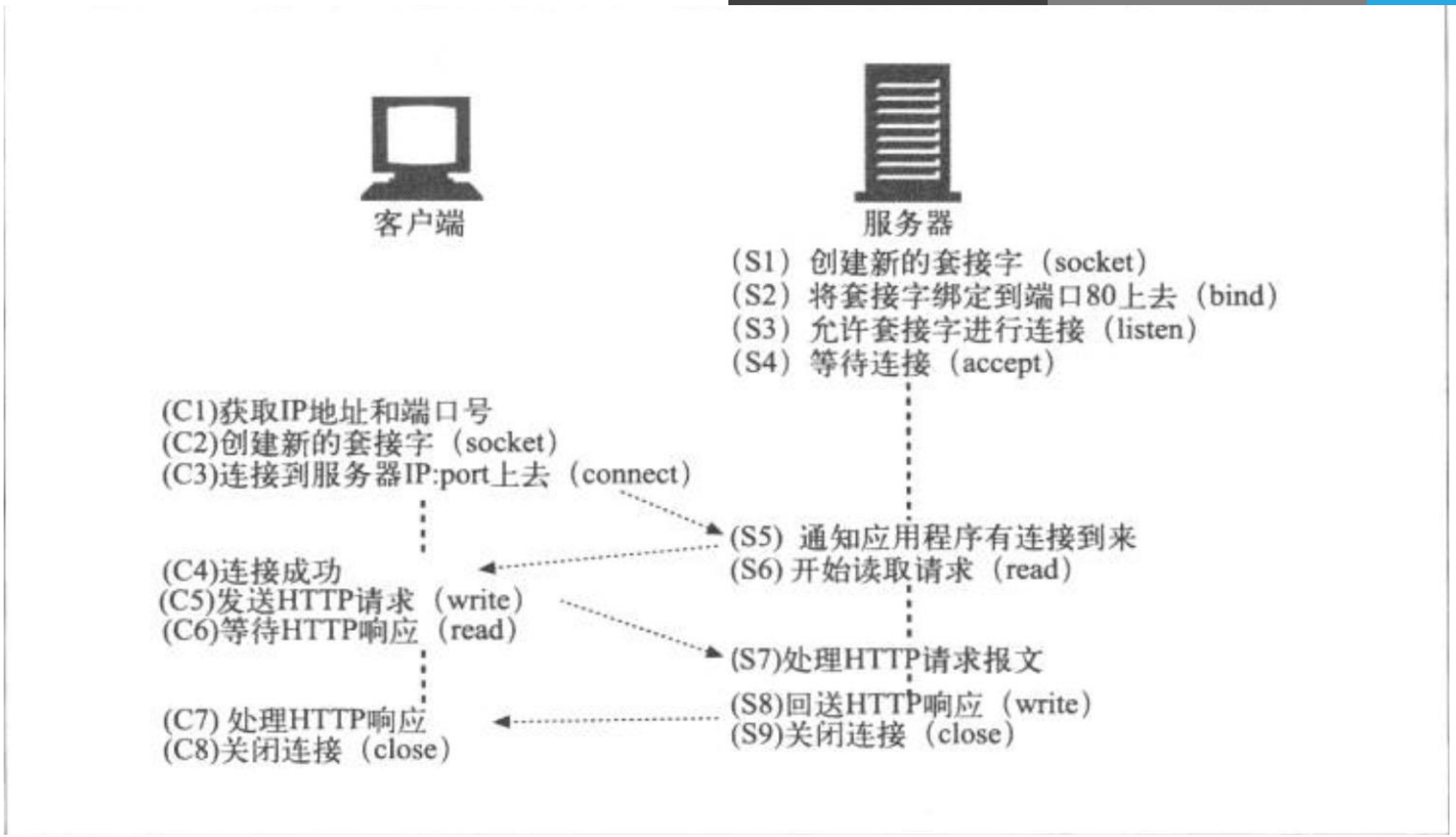
Socket通信示例：服务器端tcpserver.py


```
import socket
HOST='127.0.0.1'
PORT=9527
BUFFER=4096
sock=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
sock.bind((HOST,PORT))
sock.listen(3)
print('tcpServer listen at: %s:%s\n\r' %(HOST,PORT))
while True:
   client_sock,client_addr=sock.accept()
   print('%s:%s connect' %client_addr)
   while True:
     recv=client_sock.recv(BUFFER)
     if not recv:
       client_sock.close()
       break
     print('[Client %s:%s said]:%s' %(client_addr[0],client_addr[1],recv))
     client_sock.send(   'I am tcpServer and has received your message')
sock.close()
```

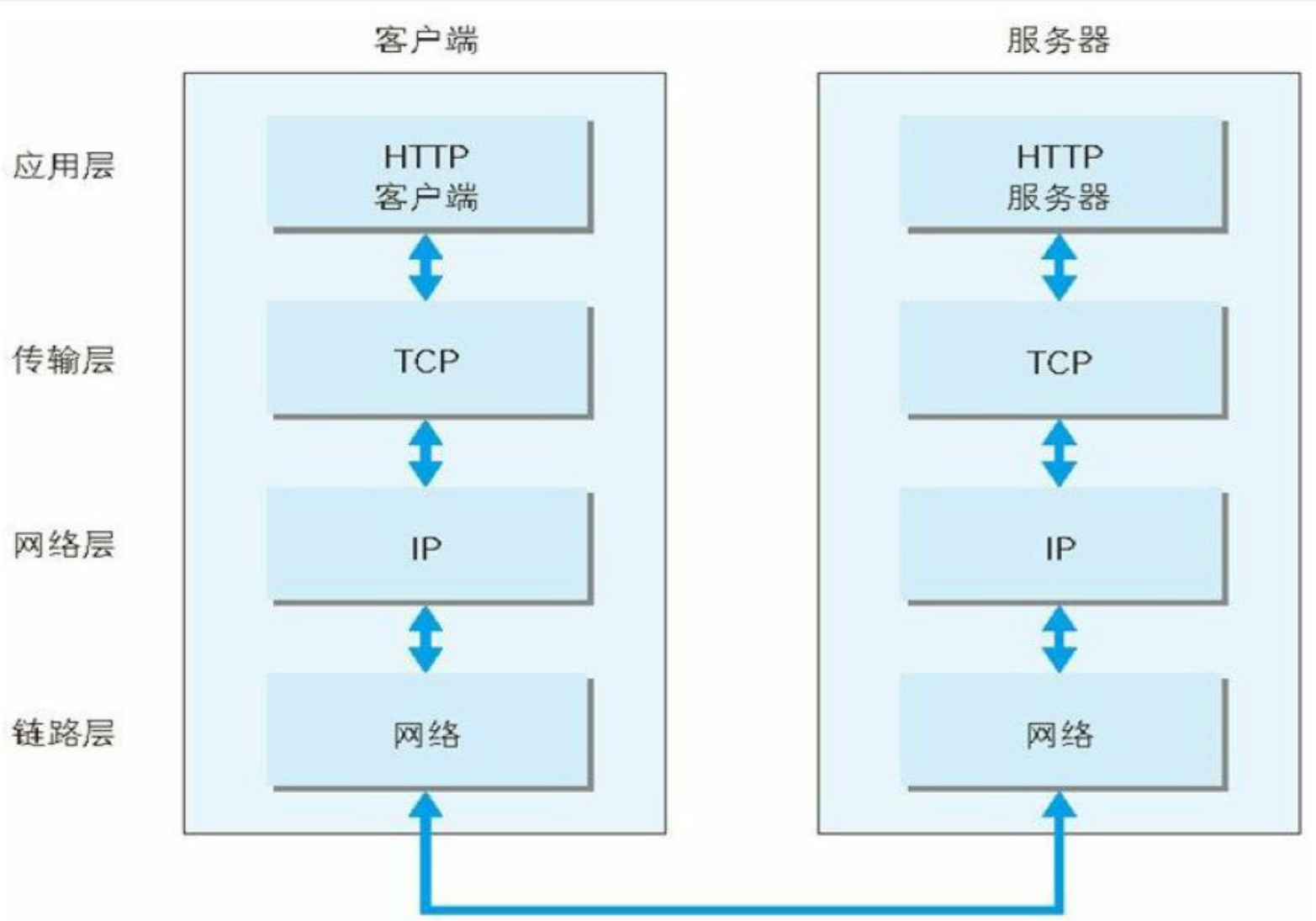
Socket通信示例：服务器端tcpclient.py


```
import socket
HOST='127.0.0.1'
PORT=9527
BUFFER=4096
sock=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
sock.connect((HOST,PORT))
sock.send(  ‘hello, tcpServer! , I am TcpClient')
recv=sock.recv(BUFFER)
print('[tcpServer said]: %s' % recv)
sock.close()
```

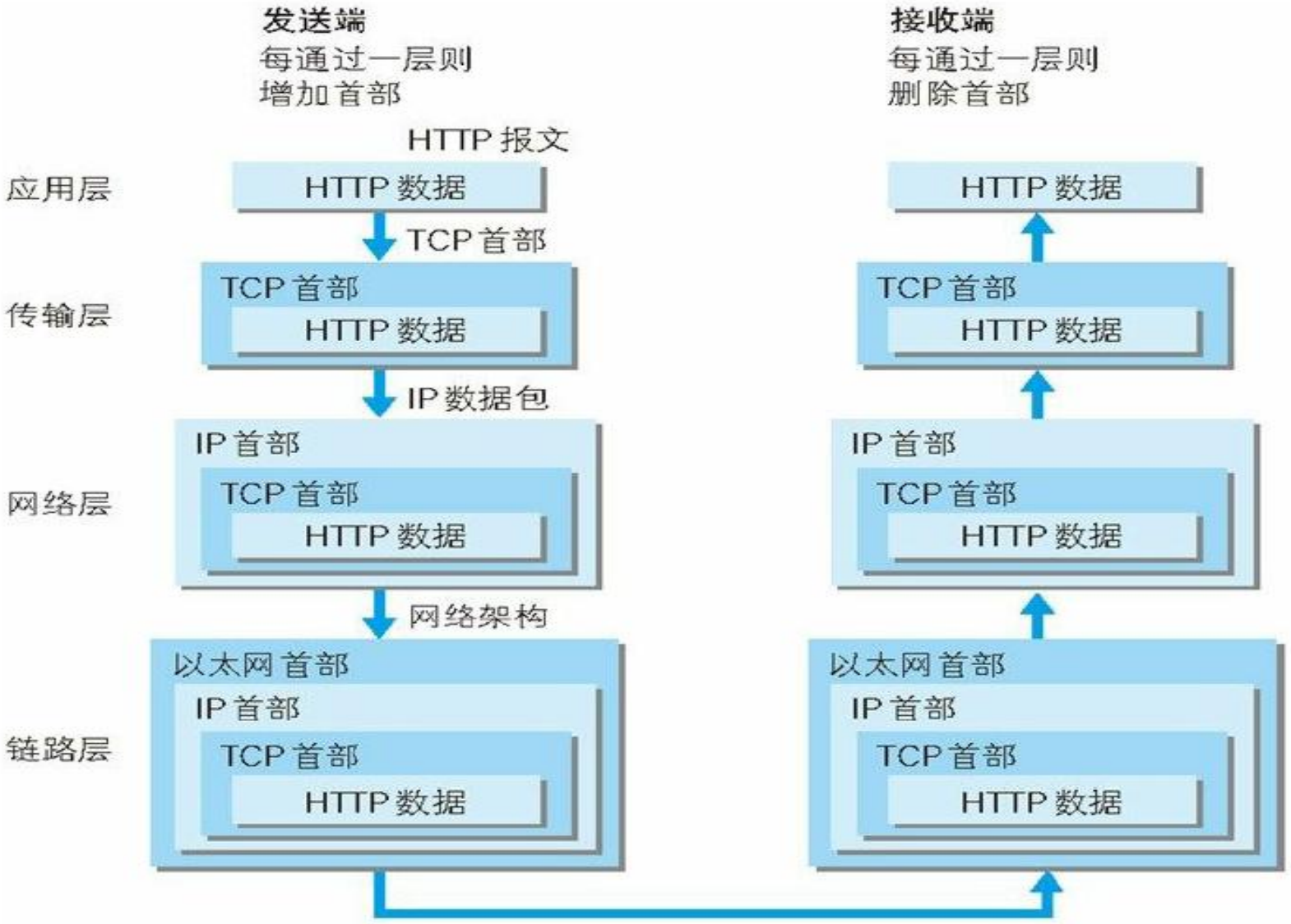
HTTP服务通信过程


客户端
服务器
(S1) 创建新的套接字（socket）
(S2) 将套接字绑定到端口80上去（bind）
(S3) 允许套接字进行连接（listen）
(S4) 等待连接（accept）
(C1)获取IP地址和端口号
(C2)创建新的套接字（socket）
(C3)连接到服务器IP:port上去（connect）
(S5) 通知应用程序有连接到来
(S6) 开始读取请求（read）
(C4)连接成功
(C5)发送HTTP请求（write）
(C6)等待HTTP响应（read）
(S7)处理HTTP请求报文
(S8)回送HTTP响应（write）
(S9)关闭连接（close）
(C7) 处理HTTP响应
(C8)关闭连接（close）

HTTP服务通信过程



HTTP服务通信过程


发送端
每通过一层则
增加首部
接收端
每通过一层则
删除首部
HTTP 报文
应用层
传输层
网络层
链路层
HTTP 数据
TCP 首部
IP 数据包
IP 首部
网络架构
以太网 首部

HTTP相关术语

- http: Hyper Text Transfer Protocol, 80/tcp
- html: Hyper Text Markup Language 超文本标记语言，编程语言
- 示例：

```
<html>
<head>
<meta http-equiv=Content-Type content="text/html;charset=utf-8">
<title>html语言</title>
</head>
<body>
<img src="http://www.magedu.com/wp-content/uploads/2017/09/logo.png" >
<h1>你好</h1>
<p><a href=http://www.magedu.com>马哥教育</a>欢迎你</p>
</body>
</html>
```

- CSS：Cascading Style Sheet 层叠样式表
- Js：javascript

Http相关术语

- MIME： Multipurpose Internet Mail Extensions 多用途互联网邮件扩展
  /etc/mime.types
- 格式：major/minor
  text/plain
  text/html
  text/css
  image/jpeg
  image/png
  video/mp4
  application/javascript
- 参考：https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Basics_of_HTTP/MIME_Types
  http://www.w3school.com.cn/media/media_mimeref.asp

HTTP工作机制

- 工作机制：

  http请求：http request

  http响应：http response

  一次http事务：请求<-->响应

- Web资源：web resource

  一个网页由多个资源（文件）构成，打开一个页面，通常会有多个资源展示出来，但是每个资源都要单独请求。因此，一个“Web 页面”通常并不是单个资源，而是一组资源的集合

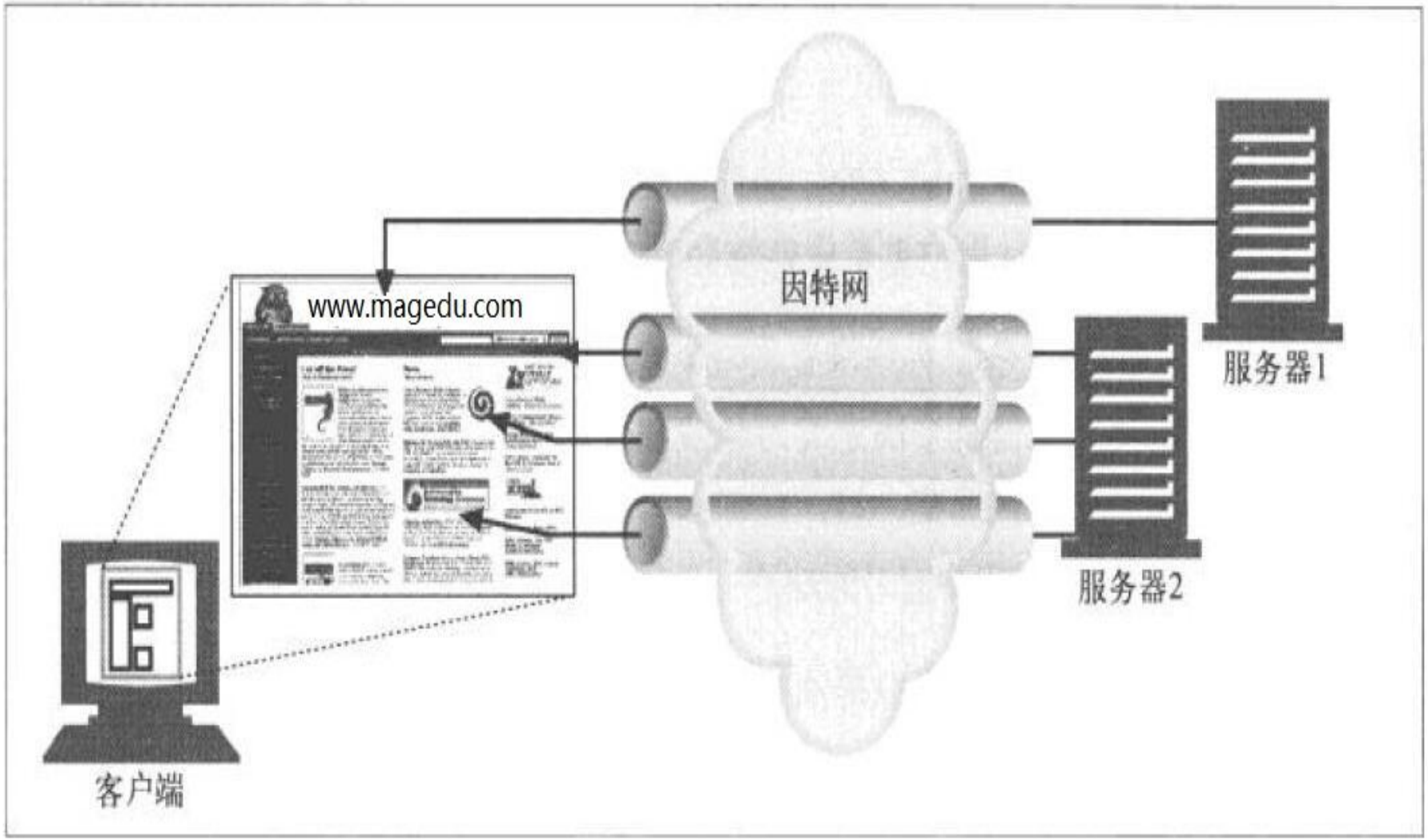
  - 静态文件：无需服务端做出额外处理

    文件后缀：.html, .txt, .jpg, .js, .css, .mp3, .avi

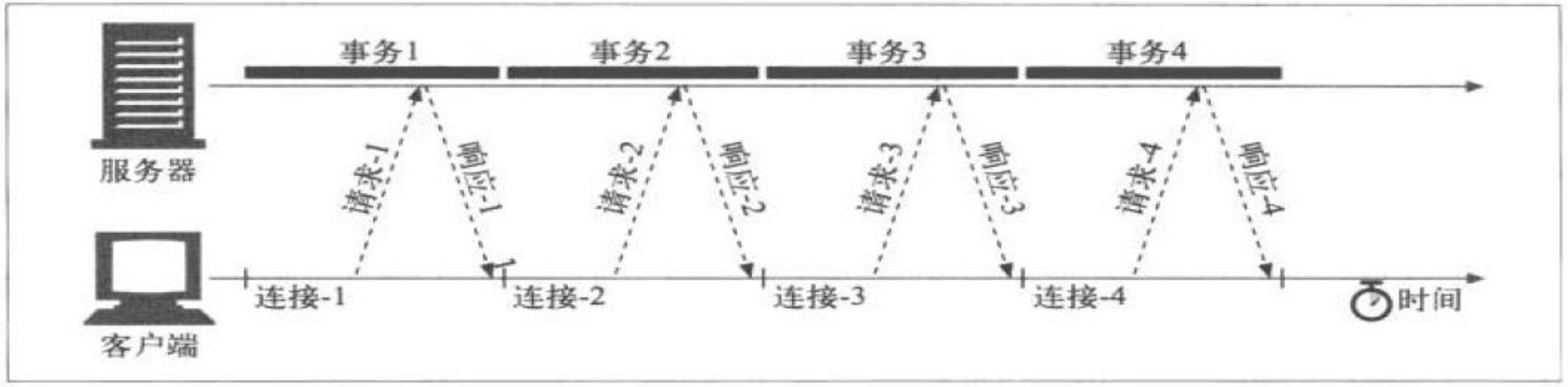
  - 动态文件：服务端执行程序，返回执行的结果

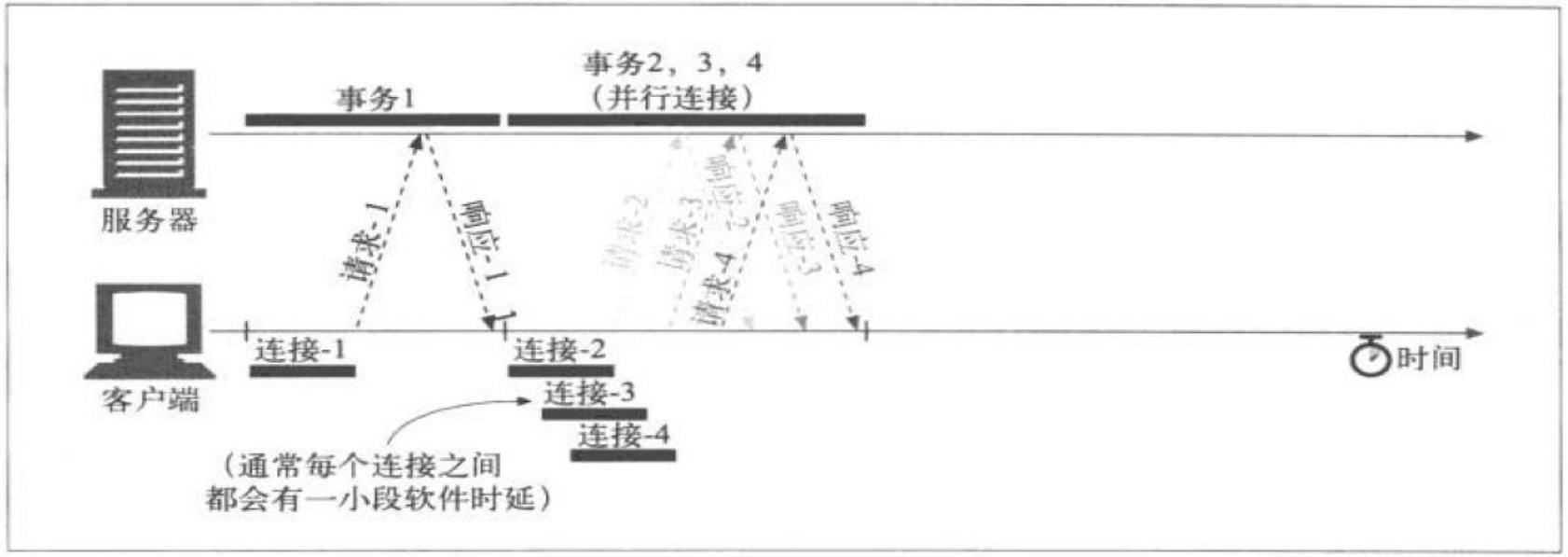
    文件后缀：.php, .jsp ,.asp

HTTP连接请求

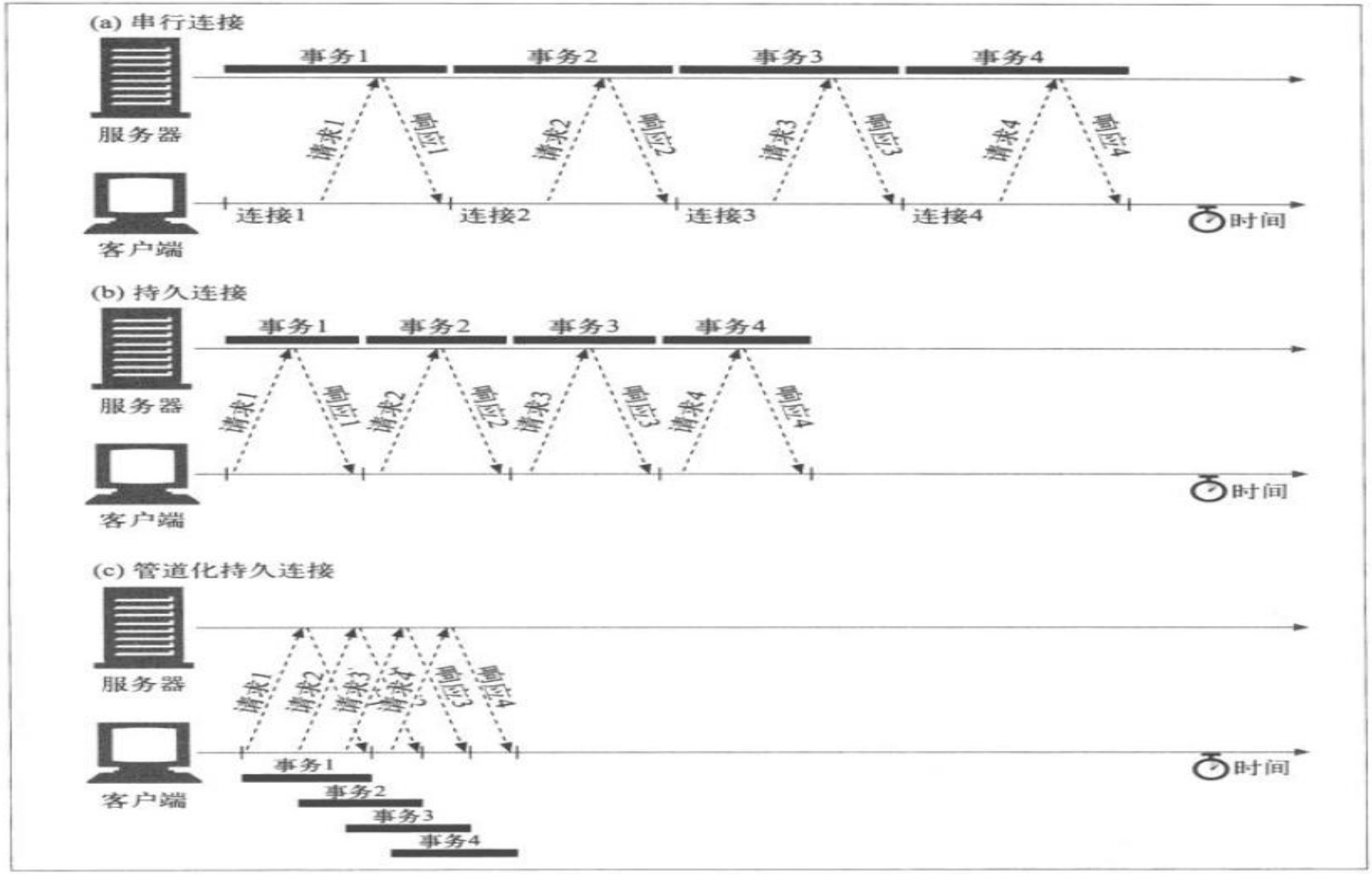


串行和并行连接


事务1
事务2
事务3
事务4
服务器
客户端
请求-1
1-响应
请求-2
2-响应
请求-3
3-响应
请求-4
4-响应
连接-1
连接-2
连接-3
连接-4
时间


事务2，3，4
（并行连接）
事务1
服务器
客户端
请求-1
1-响应
请求-4
4-响应
连接-1
连接-2
连接-3
连接-4
时间
（通常每个连接之间
都会有一小段软件时延）

串行,持久连接和管道


(a) 串行连接
事务1
事务2
事务3
事务4
服务器
请求1
响应1
请求2
响应2
请求3
响应3
请求4
响应4
连接1
连接2
连接3
连接4
客户端
时间
(b) 持久连接
事务1
事务2
事务3
事务4
服务器
请求1
响应1
请求2
响应2
请求3
响应3
请求4
响应4
客户端
时间
(c) 管道化持久连接
服务器
请求1
请求2
请求3
请求4
响应1
响应2
客户端
事务1
事务2
事务3
事务4
时间

HTTP工作机制

- 提高HTTP连接性能
  - 并行连接：通过多条TCP连接发起并发的HTTP请求
  - 持久连接：keep-alive,长连接，重用TCP连接，以消除连接和关闭的时延,以事务个数和时间来决定是否关闭连接
  - 管道化连接：通过共享TCP连接发起并发的HTTP请求
  - 复用的连接：交替传送请求和响应报文（实验阶段）

HTTP协议

- http/0.9：1991，原型版本，功能简陋，只有一个命令GET。GET /index.html ,服务器只能回应HTML格式字符串，不能回应别的格式
- http/1.0: 1996年5月,支持cache, MIME, method
  - 每个TCP连接只能发送一个请求，发送数据完毕，连接就关闭，如果还要请求其他资源，就必须再新建一个连接
  - 引入了POST命令和HEAD命令
  - 头信息是 ASCII 码，后面数据可为任何格式。服务器回应时会告诉客户端，数据是什么格式，即Content-Type字段的作用。这些数据类型总称为MIME 多用途互联网邮件扩展，每个值包括一级类型和二级类型，预定义的类型，也可自定义类型, 常见Content-Type值：text/xml image/jpeg audio/mp3

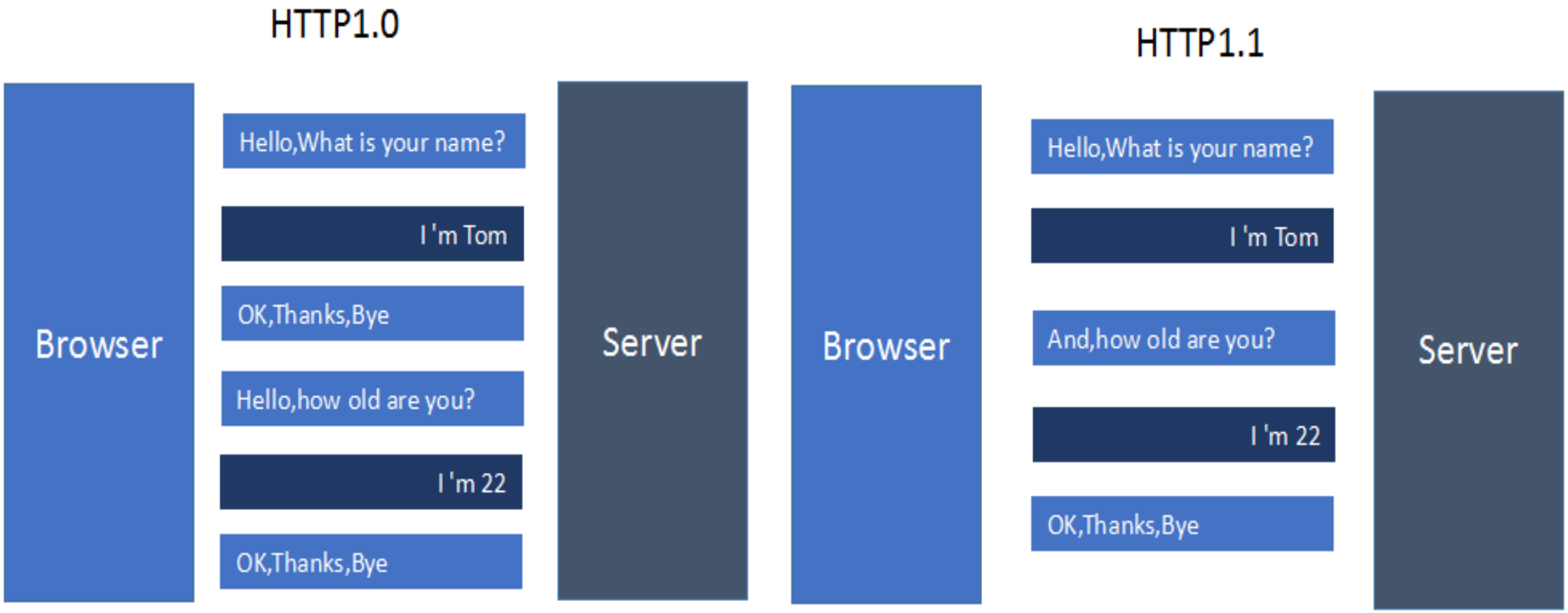
HTTP协议

- http/1.1：1997年1月
  - 引入了持久连接（persistent connection），即TCP连接默认不关闭，可以被多个请求复用，不用声明Connection: keep-alive。对于同一个域名，大多数浏览器允许同时建立6个持久连接
  - 引入了管道机制（pipelining），即在同一个TCP连接里，客户端可以同时发送多个请求，进一步改进了HTTP协议的效率
  - 新增方法：PUT、PATCH、OPTIONS、DELETE
  - 同一个TCP连接里，所有的数据通信是按次序进行的。服务器只能顺序处理回应，前面的回应慢，会有许多请求排队，造成"队头堵塞"（Head-of-line blocking）
  - 为避免上述问题，两种方法：一是减少请求数，二是同时多开持久连接。网页优化技巧，如合并脚本和样式表、将图片嵌入CSS代码、域名分片（domain sharding）等
  - HTTP 协议不带有状态，每次请求都必须附上所有信息。请求的很多字段都是重复的，浪费带宽，影响速度

HTTP1.0和HTTP1.1的区别

- 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例如Entity tag，If-Unmodified-Since, If-Match, If-None-Match等更多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，存在一些浪费带宽的现象，例如客户端只是需要某个对象的一部分，而服务器却将整个对象送过来了，并且不支持断点续传功能，HTTP1.1则在请求头引入了range头域，它允许只请求资源的某个部分，即返回码是206（Partial Content），方便了开发者自由的选择以便于充分利用带宽和连接
- 错误通知的管理，在HTTP1.1中新增24个状态响应码，如409（Conflict）表示请求的资源与资源当前状态冲突；410（Gone）表示服务器上的某个资源被永久性的删除
- 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）。但随着虚拟主机技术的发展，在一台物理服务器上可以存在多个虚拟主机（Multi-homed Web Servers），并且它们共享一个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）
- 长连接，HTTP 1.1支持长连接（PersistentConnection）和请求的流水线（Pipelining）处理，在一个TCP连接上可以传送多个HTTP请求和响应，减少了建立和关闭连接的消耗和延迟，在HTTP1.1中默认开启Connection： keep-alive，弥补了HTTP1.0每次请求都要创建连接的缺点

HTTP1.0和HTTP1.1的区别



HTTP1.0和1.1现存的问题

- HTTP1.x在传输数据时，每次都需要重新建立连接，无疑增加了大量的延迟时间，特别是在移动端更为突出
- HTTP1.x在传输数据时，所有传输的内容都是明文，客户端和服务器端都无法验证对方的身份，无法保证数据的安全性
- HTTP1.x在使用时，header里携带的内容过大，增加了传输的成本，并且每次请求header基本不怎么变化，尤其在移动端增加用户流量
- 虽然HTTP1.x支持了keep-alive，来弥补多次创建连接产生的延迟，但是keep-alive使用多了同样会给服务端带来大量的性能压力，并且对于单个文件被不断请求的服务(例如图片存放网站)，keep-alive可能会极大的影响性能，因为它在文件被请求之后还保持了不必要的连接很长时间

HTTPS

- 为解决安全问题，网景在1994年创建了HTTPS，并应用在网景导航者浏览器中。最初，HTTPS是与SSL一起使用的；在SSL逐渐演变到TLS时（其实两个是一个东西，只是名字不同而已），最新的HTTPS也由在2000年五月公布的RFC 2818正式确定下来。HTTPS就是安全版的HTTP，目前大型网站基本实现全站HTTPS
- HTTPS协议需要到CA申请证书，一般免费证书很少，需要交费
- HTTP协议运行在TCP之上，所有传输的内容都是明文，HTTPS运行在SSL/TLS之上，SSL/TLS运行在TCP之上，所有传输的内容都经过加密的
- HTTP和HTTPS使用的是不同的连接方式，端口不同，前者是80，后者是443
- HTTPS可以有效的防止运营商劫持，解决了防劫持的一个大问题
- HTTPS 中的SSL握手等过程降低用户访问速度，但是只要经过合理优化和部署，HTTPS 对速度的影响完全可以接受

SPDY

- SPDY：2009年,谷歌研发，综合HTTPS和HTTP两者有点于一体的传输协议，主要特点：
- 降低延迟，针对HTTP高延迟的问题，SPDY优雅的采取了多路复用（multiplexing）。多路复用通过多个请求stream共享一个tcp连接的方式，解决了HOL blocking的问题，降低了延迟同时提高了带宽的利用率
- 请求优先级（request prioritization）。多路复用带来一个新的问题是，在连接共享的基础之上有可能会导致关键请求被阻塞。SPDY允许给每个request设置优先级，重要的请求就会优先得到响应。比如浏览器加载首页，首页的html内容应该优先展示，之后才是各种静态资源文件，脚本文件等加载，可以保证用户能第一时间看到网页内容
- header压缩。HTTP1.x的header很多时候都是重复多余的。选择合适的压缩算法可以减小包的大小和数量
- 基于HTTPS的加密协议传输，大大提高了传输数据的可靠性
- 服务端推送（server push），采用了SPDY的网页，例如网页有一个sytle.css的请求，在客户端收到sytle.css数据的同时，服务端会将sytle.js的文件推送给客户端，当客户端再次尝试获取sytle.js时就可以直接从缓存中获取到，不用再发请求了

HTTP2

- http/2.0：2015年
  - HTTP2.0是SPDY的升级版
  - 头信息和数据体都是二进制，称为头信息帧和数据帧
  - 复用TCP连接，在一个连接里，客户端和浏览器都可以同时发送多个请求或回应，且不用按顺序一一对应，避免了“队头堵塞“,此双向的实时通信称为多工（Multiplexing）
  - 引入头信息压缩机制（header compression）,头信息使用gzip或compress压缩后再发送；客户端和服务器同时维护一张头信息表，所有字段都会存入这个表，生成一个索引号，不发送同样字段，只发送索引号，提高速度
  - HTTP/2 允许服务器未经请求，主动向客户端发送资源，即服务器推送（server push）
- HTTP2.0和SPDY区别：
  - HTTP2.0 支持明文 HTTP 传输，而 SPDY 强制使用 HTTPS
  - HTTP2.0 消息头的压缩算法采用 HPACK，而非 SPDY 采用的 DEFLATE

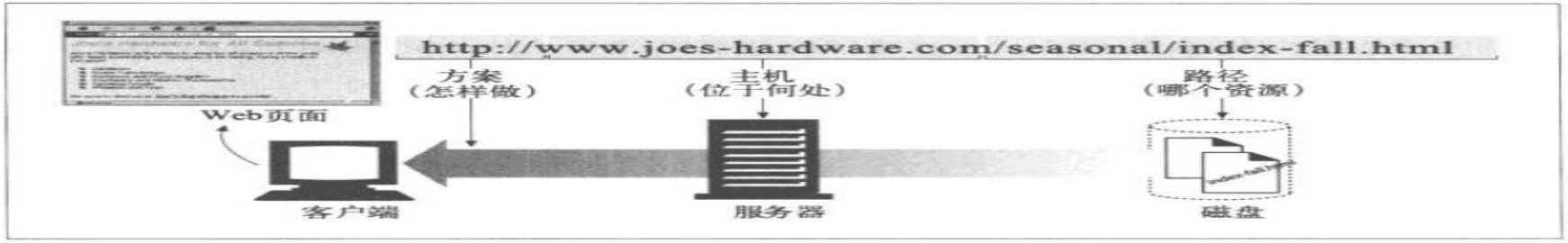
URI

- URI: Uniform Resource Identifier 统一资源标识，分为URL和URN
  - URN: Uniform Resource Naming，统一资源命名

    示例： P2P下载使用的磁力链接是URN的一种实现

    magnet:?xt=urn:btih:660557A6890EF888666
  - URL: Uniform Resorce Locator，统一资源定位符，用于描述某服务器某特定资源位置
  - 两者区别：URN如同一个人的名称，而URL代表一个人的住址。换言之，URN定义某事物的身份，而URL提供查找该事物的方法。URN仅用于命名，而不指定地址

URL组成



- <scheme>://<user>:<password>@<host>:<port>/<path>;<params>?<query>#<frag>
- scheme:方案，访问服务器以获取资源时要使用哪种协议
- user:用户，某些方案访问资源时需要的用户名
- password:密码，用户对应的密码，中间用：分隔
- Host:主机，资源宿主服务器的主机名或IP地址
- port:端口,资源宿主服务器正在监听的端口号，很多方案有默认端口号
- path:路径,服务器资源的本地名，由一个/将其与前面的URL组件分隔
- params:参数，指定输入的参数，参数为名/值对，多个参数，用;分隔
- query:查询，传递参数给程序，如数据库，用？分隔,多个查询用&分隔
- frag:片段,一小片或一部分资源的名字，此组件在客户端使用，用#分隔

URL示例

- http://www.magedu.com:8080/images/logo.jpg
- ftp://mage:password@172.16.0.1/pub/linux.ppt
- rtsp://videoserver/video_demo/

  Real Time Streaming Protocol
- http://www.magedu.com/bbs/hello;gender=f/send;type=title
- https://list.jd.com/list.html?cat=670,671,672&ev=149_2992&sort=sort_totalsales15_desc&trans=1
- http://apache.org/index.html#projects-list

网站访问量

- IP(独立IP)：即Internet Protocol,指独立IP数。一天内来自相同客户机IP地址只计算一次，记录远程客户机IP地址的计算机访问网站的次数，是衡量网站流量的重要指标
- PV(访问量)： 即Page View, 页面浏览量或点击量，用户每次刷新即被计算一次，PV反映的是浏览某网站的页面数，PV与来访者的数量成正比，PV并不是页面的来访者数量，而是网站被访问的页面数量
- UV(独立访客)：即Unique Visitor,访问网站的一台电脑为一个访客。一天内相同的客户端只被计算一次。可以理解成访问某网站的电脑的数量。网站判断来访电脑的身份是通过来访电脑的cookies实现的。如果更换了IP后但不清除cookies，再访问相同网站，该网站的统计中UV数是不变的
- 网站统计：http://www.alexa.cn/rank/

网站访问统计示例

- 示例：
- 甲乙丙三人在同一台通过ADSL上网的电脑上（中间没有断网），分别访问www.magedu.com网站，并且每人各浏览了2个页面，那么网站的流量统计是：

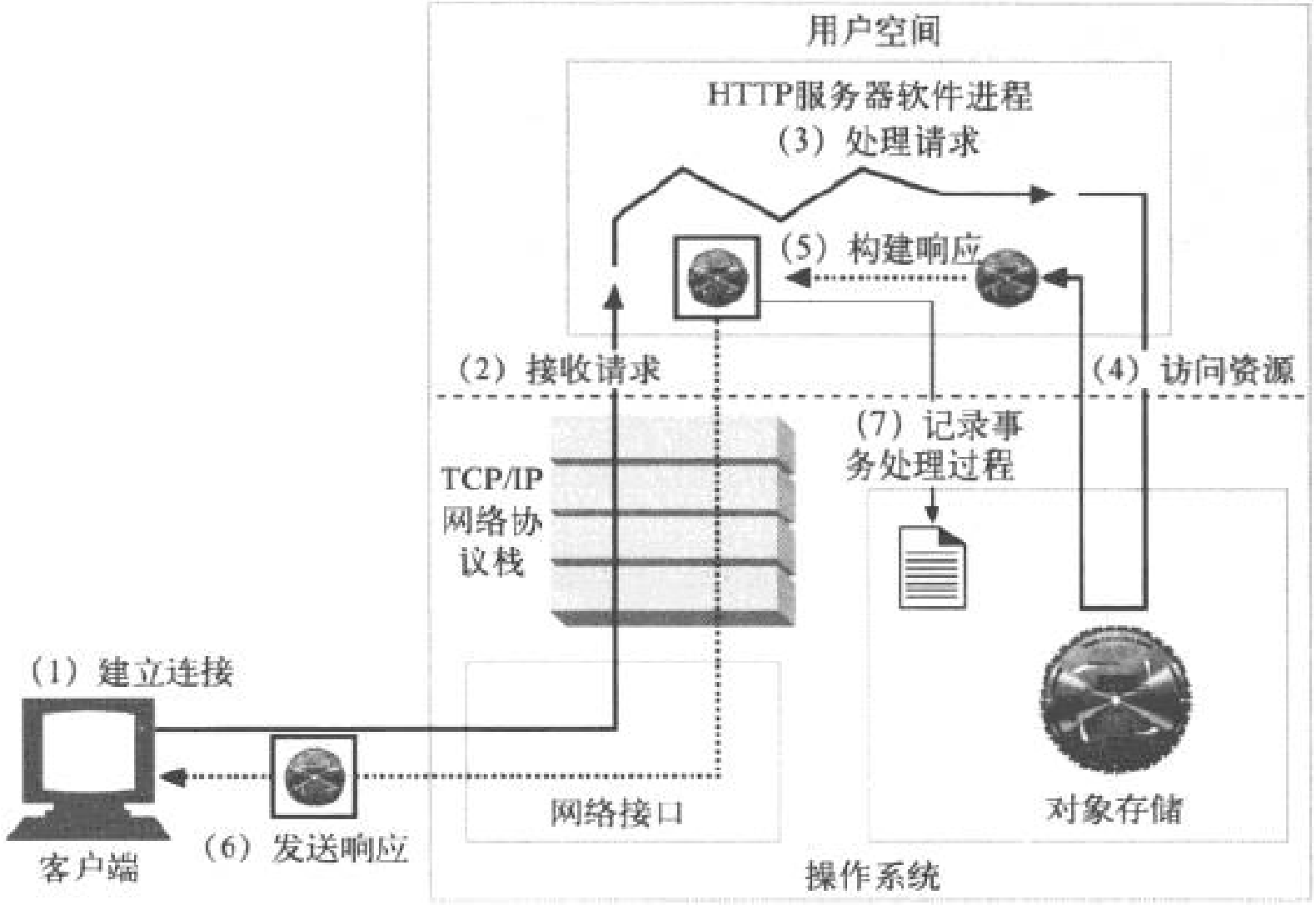
  IP:　1　　PV:6　　UV：1
- 若三人都是ADSL重新拨号后,各浏览了2个页面，则

  IP:　3　　PV:6　　UV：1

网站访问量

- QPS：request per second，每秒请求数
- PV，QPS,并发连接数换算公式
  - QPS= PV* 页面衍生连接次数/ 统计时间（86400）
  - 并发连接数 =QPS * http平均响应时间
- 峰值时间：每天80%的访问集中在20%的时间里，这20%时间为峰值时间
- 峰值时间每秒请求数(QPS)=（总PV数 *页面衍生连接次数）*80%）/（每天秒数 * 20%）

Web服务请求处理步骤



一次完整的http请求处理过程

- 1、建立连接：接收或拒绝连接请求
- 2、接收请求：接收客户端请求报文中对某资源的一次请求的过程
- Web访问响应模型（Web I/O）

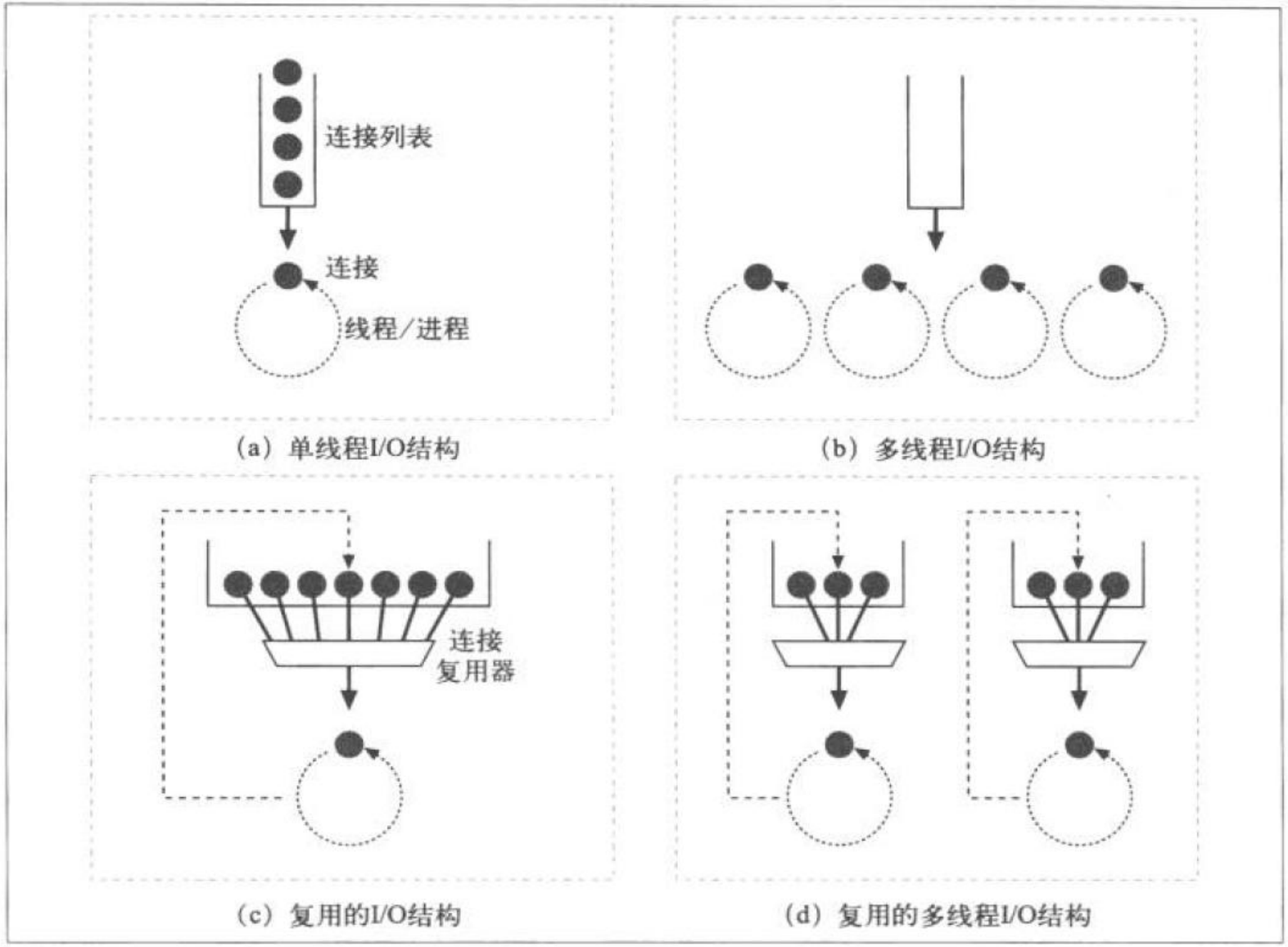
  单进程I/O模型：启动一个进程处理用户请求，而且一次只处理一个，多个请求被串行响应

  多进程I/O模型：并行启动多个进程,每个进程响应一个连接请求

  复用I/O结构：启动一个进程，同时响应N个连接请求

  复用的多进程I/O模型：启动M个进程，每个进程响应N个连接请求，同时接收M*N个请求

Web访问响应模型


连接列表
连接
线程/进程
(a) 单线程I/O结构
(b) 多线程I/O结构
连接
复用器
(c) 复用的I/O结构
(d) 复用的多线程I/O结构

一次完整的HTTP请求处理过程

- 3、处理请求：服务器对请求报文进行解析，并获取请求的资源及请求方法等相关信息，根据方法，资源，首部和可选的主体部分对请求进行处理

  元数据：请求报文首部

  <method> <URL> <VERSION>

  HEADERS　格式 name:value

  <request body>

  示例：

  Host: www.magedu.com　请求的主机名称

  Server: Apache/2.4.7

  - HTTP常用请求方式，Method

    GET、POST、HEAD、PUT、DELETE、TRACE、OPTIONS

一次完整的HTTP请求处理过程

- 4、访问资源：

服务器获取请求报文中请求的资源web服务器，即存放了web资源的服务器，负责向请求者提供对方请求的静态资源，或动态运行后生成的资源

资源放置于本地文件系统特定的路径：DocRoot

DocRoot ➔ /var/www/html

/var/www/html/images/logo.jpg

http://www.magedu.com/images/logo.jpg

- web服务器资源路径映射方式：

(a) docroot

(b) alias

(c) 虚拟主机docroot

(d) 用户家目录docroot

一次完整的HTTP请求处理过程

- 5、构建响应报文：

一旦Web服务器识别除了资源，就执行请求方法中描述的动作，并返回响应报文。响应报文中 包含有响应状态码、响应首部，如果生成了响应主体的话，还包括响应主体

1）响应实体：如果事务处理产生了响应主体，就将内容放在响应报文中回送过去。响应报文中通常包括：

描述了响应主体MIME类型的Content-Type首部

描述了响应主体长度的Content-Length

实际报文的主体内容

2）URL重定向：web服务构建的响应并非客户端请求的资源，而是资源另外一个访问路径

永久重定向：http://www.360buy.com

临时重定向：http://www.taobao.com

一次完整的HTTP请求处理过程

3）MIME类型：

Web服务器要负责确定响应主体的MIME类型。多种配置服务器的方法可将MIME类型与资源管理起来

魔法分类：Apache web服务器可以扫描每个资源的内容，并将其与一个已知模式表(被称为魔法文件)进行匹配，以决定每个文件的MIME类型。这样做可能比较慢，但很方便，尤其是文件没有标准扩展名时

显式分类：可以对Web服务器进行配置，使其不考虑文件的扩展名或内容，强制特定文件或目录内容拥有某个MIME类型

类型协商： 有些Web服务器经过配置，可以以多种文档格式来存储资源。在这种情况下，可以配置Web服务器，使其可以通过与用户的协商来决定使用哪种格式(及相关的MIME类型)"最好"

一次完整的HTTP请求处理过程

◆ 6、发送响应报文

Web服务器通过连接发送数据时也会面临与接收数据一样的问题。服务器可能有很多条到各个客户端的连接，有些是空闲的，有些在向服务器发送数据，还有一些在向客户端回送响应数据。服务器要记录连接的状态，还要特别注意对持久连接的处理。对非持久连接而言，服务器应该在发送了整条报文之后，关闭自己这一端的连接。对持久连接来说，连接可能仍保持打开状态，在这种情况下，服务器要正确地计算Content-Length首部，不然客户端就无法知道响应什么时候结束了

◆ 7、记录日志

最后，当事务结束时，Web服务器会在日志文件中添加一个条目，来描述已执行的事务

HTTP服务器应用

- http服务器程序
  - httpd apache，存在C10K（10K connections）问题
  - nginx 解决C10K问题
  - lighttpd
- 应用程序服务器
  - IIS .asp
  - tomcat .jsp
  - jetty 开源的servlet容器，基于Java的web容器
  - Resin CAUCHO公司，支持servlets和jsp的引擎
  - webshpere(IBM), weblogic(BEA), jboss,oc4j(Oracle)
- 市场占有率统计
  - http://www.netcraft.com

Httpd介绍

- httpd
  - 20世纪90年代初，国家超级计算机应用中心NCSA开发
  - 1995年开源社区发布apache（a patchy server）
    - ASF: apache software foundation
    - FSF：Free Software Foundation
- 特性：
  - 高度模块化：core + modules
  - DSO：Dynamic Shared Object 动态加/卸载
  - MPM：multi-processing module多路处理模块

MPM工作模式

- prefork：多进程I/O模型，每个进程响应一个请求，默认模型

  一个主进程：生成和回收n个子进程，创建套接字，不响应请求

  多个子进程：工作work进程，每个子进程处理一个请求；系统初始时，预先生成多个空闲进程，等待请求，最大不超过1024个

- worker：复用的多进程I/O模型,多进程多线程，IIS使用此模型

  一个主进程：生成m个子进程，每个子进程负责生个n个线程，每个线程响应一个请求，并发响应请求：m*n

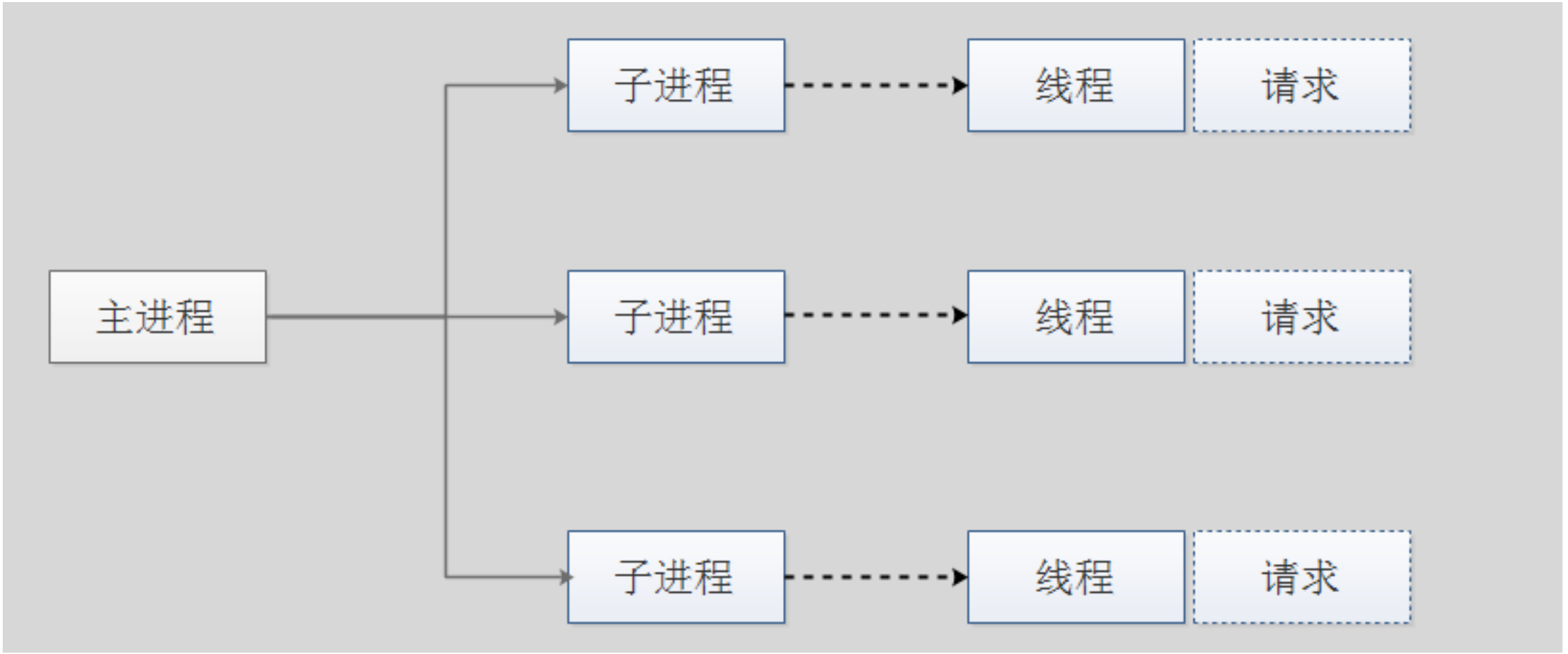
- event：事件驱动模型（worker模型的变种）

  一个主进程：生成m个子进程，每个子进程负责生个n个线程，每个线程响应一个请求，并发响应请求：m*n，有专门的监控线程来管理这些keep-alive类型的线程，当有真实请求时，将请求传递给服务线程，执行完毕后，又允许释放。这样增强了高并发场景下的请求处理能力

  httpd-2.2：event 测试版，centos6默认

  httpd-2.4：event 稳定版，centos7默认

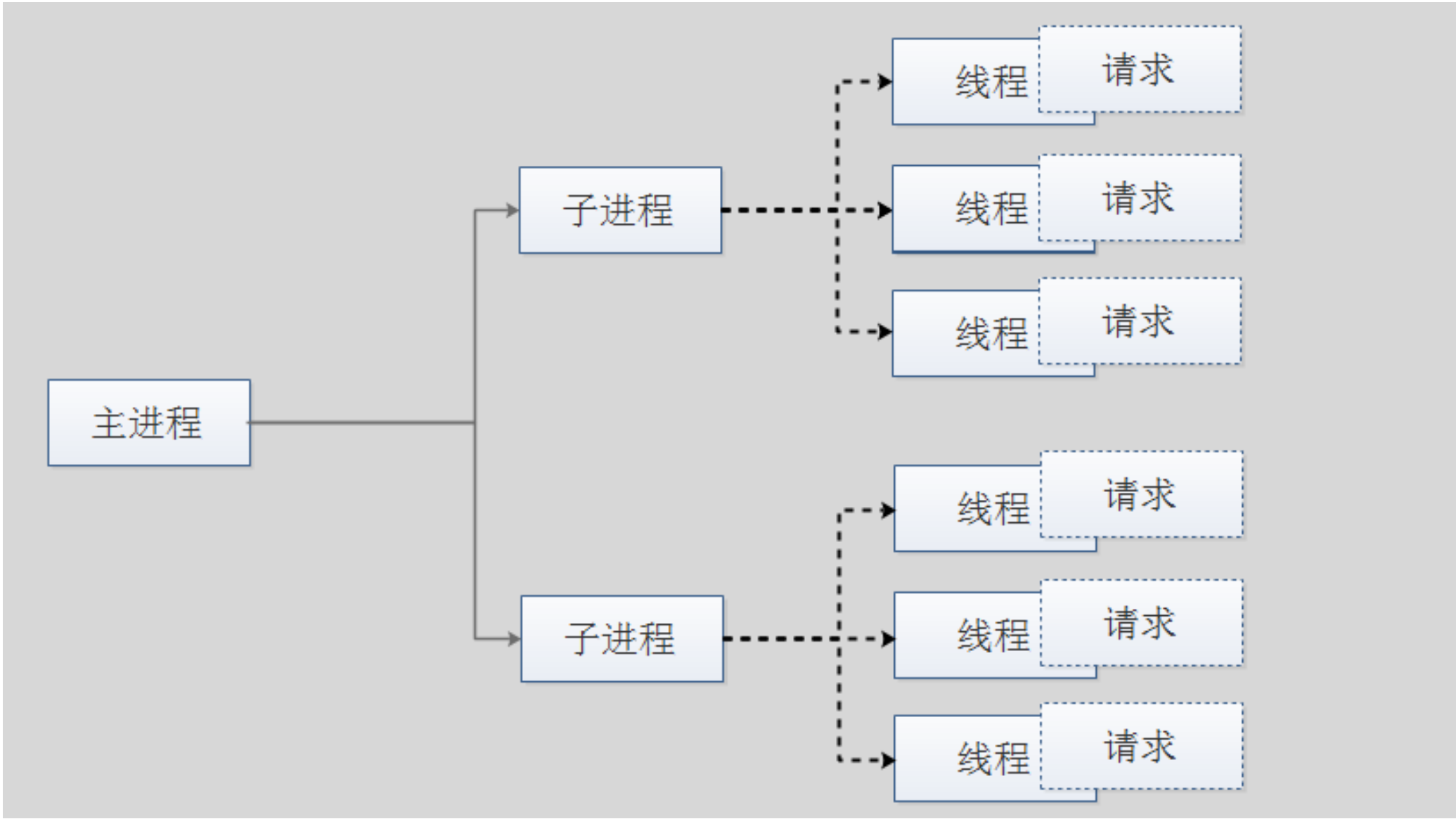
prefork MPM



Prefork MPM

- Prefork MPM: 预派生模式，有一个主控制进程，然后生成多个子进程,每个子进程有一个独立的线程响应用户请求，相对比较占用内存，但是比较稳定，可以设置最大和最小进程数，是最古老的一种模式，也是最稳定的模式，适用于访问量不是很大的场景
- 优点：稳定
- 缺点：慢，占用资源，不适用于高并发场景

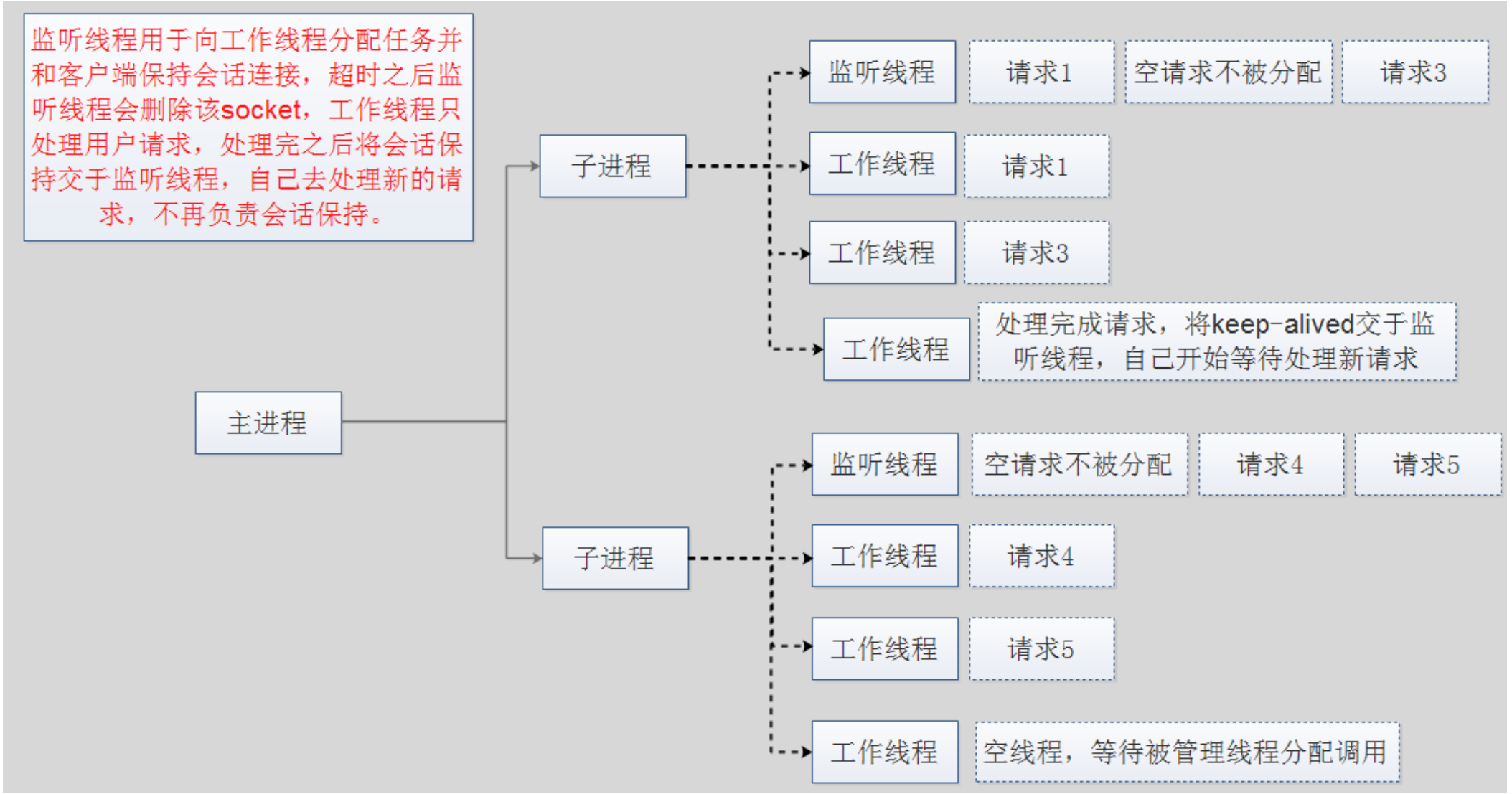
worker MPM



worker MPM

- worker MPM：是一种多进程和多线程混合的模型，有一个控制进程，启动多个子进程，每个子进程里面包含固定的线程，使用线程程来处理请求，当线程不够使用的时候会再启动一个新的子进程，然后在进程里面再启动线程处理请求，由于其使用了线程处理请求，因此可以承受更高的并发。
- 优点：相比prefork 占用的内存较少，可以同时处理更多的请求
- 缺点：使用keep-alive的长连接方式，某个线程会一直被占据，即使没有传输数据，也需要一直等待到超时才会被释放。如果过多的线程，被这样占据，也会导致在高并发场景下的无服务线程可用。（该问题在prefork模式下，同样会发生）

event MPM



event MPM

- event MPM：Apache中最新的模式，属于事件驱动模型(epoll)，每个进程响应多个请求，在现在版本里的已经是稳定可用的模式。它和worker模式很像，最大的区别在于，它解决了keep-alive场景下，长期被占用的线程的资源浪费问题（某些线程因为被keep-alive，空挂在哪里等待，中间几乎没有请求过来，甚至等到超时）。event MPM中，会有一个专门的线程来管理这些keep-alive类型的线程，当有真实请求过来的时候，将请求传递给服务线程，执行完毕后，又允许它释放。这样增强了高并发场景下的请求处理能力
- event只在有数据发送的时候才开始建立连接，连接请求才会触发工作线程，即使用了TCP的一个选项，叫做延迟接受连接TCP_DEFER_ACCEPT，加了这个选项后，若客户端只进行TCP连接，不发送请求，则不会触发Accept操作，也就不会触发工作线程去干活，进行了简单的防攻击（TCP连接）
- 优点：单线程响应多请求，占据更少的内存，高并发下表现更优秀，会有一个专门的线程来管理keep-alive类型的线程，当有真实请求过来的时候，将请求传递给服务线程，执行完毕后，又允许它释放
- 缺点：没有线程安全控制

httpd功能特性

- 虚拟主机
  - IP、Port、FQDN
- CGI：Common Gateway Interface，通用网关接口
- 反向代理
- 负载均衡
- 路径别名
- 丰富的用户认证机制
  - basic
  - digest
- 支持第三方模块

httpd-2.4

- 新特性
  - MPM支持运行为DSO机制；以模块形式按需加载
  - event MPM生产环境可用
  - 异步读写机制
  - 支持每模块及每目录的单独日志级别定义
  - 每请求相关的专用配置
  - 增强版的表达式分析式
  - 毫秒级持久连接时长定义
  - 基于FQDN的虚拟主机不需要NameVirutalHost指令
  - 新指令，AllowOverrideList
  - 支持用户自定义变量
  - 更低的内存消耗

Httpd 安装

- 版本:
  - CentOS 7: 2.4
  - CentOS 6: 2.2
- 安装方式：
  - rpm：centos发行版，稳定，建议使用
  - 编译：定制或特殊需求
- CentOS 7程序环境：httpd-2.4
  - 配置文件：
    - /etc/httpd/conf/httpd.conf
    - /etc/httpd/conf.d/*.conf
  - 检查配置语法：
    - httpd –t

Httpd 程序环境

- 服务单元文件： /usr/lib/systemd/system/httpd.service
  配置文件：/etc/sysconfig/httpd
- 服务控制和启动：
  systemctl enable|disable httpd.service
  systemctl {start|stop|restart|status|reload} httpd.service
- 站点网页文档根目录：
  /var/www/html
- 模块文件路径：
  /etc/httpd/modules
  /usr/lib64/httpd/modules

Httpd 程序环境

- 主程序文件：
  - /usr/sbin/httpd
- 主进程文件：
  - /etc/httpd/run/httpd.pid
- 日志文件目录：
  - /var/log/httpd
  - access_log: 访问日志
  - error_log：错误日志
- 帮助文档包：
  - httpd-manual

Httpd 常见配置

- httpd配置文件的组成：
- 主要组成
  - Global Environment
  - Main server configuration
  - virtual host
- 配置格式：directive value
  - directive　　不区分字符大小写
  - value　　为路径时，是否区分大小写，取决于文件系统
- 官方帮助
  - http://httpd.apache.org/docs/2.4/

Httpd 常见配置

- 显示服务器版本信息

  ServerTokens Major|Minor|Min[imal]|Prod[uctOnly]|OS|Full

  ServerTokens Prod[uctOnly] ：Server: Apache

  ServerTokens Major: Server: Apache/2

  ServerTokens Minor: Server: Apache/2.0

  ServerTokens Min[imal]: Server: Apache/2.0.41

  ServerTokens OS: Server: Apache/2.0.41 (Unix)

  ServerTokens Full (or not specified): Server: Apache/2.0.41 (Unix) PHP/4.2.2 MyMod/1.2

  This setting applies to the entire server and cannot be enabled or disabled on a virtualhost-by-virtualhost basis.

  After version 2.0.44, this directive also controls the information presented by the ServerSignature directive.

  建议使用：ServerTokens Prod

Httpd 常见配置

- 监听的IP和Port

  Listen [IP:]PORT

  (1) 省略IP表示为本机所有IP

  (2) Listen指令至少一个，可重复出现多次

  Listen 80

  Listen 8080

  示例：

  Listen 192.168.1.100:8080

  Lsten 80

Httpd 常见配置

- 持久连接

Persistent Connection：连接建立，每个资源获取完成后不会断开连接，而是继续等待其它的请求完成，默认关闭持久连接

断开条件：时间限制：以秒为单位， 默认5s，httpd-2.4 支持毫秒级

副作用：对并发访问量大的服务器，持久连接会使有些请求得不到响应

折衷：使用较短的持久连接时间

设置：KeepAlive On|Off

KeepAliveTimeout 15

MaxKeepAliveRequests 500 持久连接最大接收的请求数

测试：telnet WEB_SERVER_IP PORT

GET /URL HTTP/1.1

Host: WEB_SERVER_IP

Httpd 常见配置

- DSO： Dynamic Shared Object
  加载动态模块配置，不需重启即生效
  /etc/httpd/conf/httpd.conf
  Include conf.modules.d/*.conf
  配置指定实现模块加载格式：
  LoadModule <mod_name> <mod_path>
  模块文件路径可使用相对路径：相对于ServerRoot（默认/etc/httpd）
  示例：LoadModule auth_basic_module modules/mod_auth_basic.so
- 动态模块路径： /usr/lib64/httpd/modules/
- 查看静态编译的模块
  httpd -l
- 查看静态编译及动态装载的模块
  httpd –M

Httpd 常见配置

- MPM（ Multi-Processing Module ）多路处理模块
  prefork, worker, event
- 切换使用的MPM
  /etc/httpd/conf.modules.d/00-mpm.conf
  启用要启用的MPM相关的LoadModule指令即可

Httpd 常见配置

- prefork的配置：

StartServers	2000
MinSpareServers	2000
MaxSpareServers	2000
ServerLimit	2560 最多进程数,最大值 20000
MaxRequestWorkers	2560 最大的并发连接数，默认256
MaxRequestsPerChild	4000 子进程最多能处理的请求数量。在处理MaxRequestsPerChild 个请求之后,子进程将会被父进程终止，这时候子进程占用的内存就会释放(为0时永远不释放）
MaxConnectionsPerChild	4000 子进程最多能处理的连接数量，代替上面MaxRequestsPerChild ，httpd.2.4.9开始支持

Httpd 常见配置

- worker的配置：

```
ServerLimit          16
StartServers          2
MaxRequestWorkers  150
MinSpareThreads      25
MaxSpareThreads      75
ThreadsPerChild      25
```

Httpd 常见配置

- 定义'Main' server的文档页面路径

  DocumentRoot “/path”

  文档路径映射：

  DocumentRoot指向的路径为URL路径的起始位置

  示例：

  DocumentRoot "/app/data “

  http://HOST:PORT/test/index.html --> /app/data/test/index.html

  注意：SELinux和iptables的状态

- 定义站点主页面

  DirectoryIndex index.html

Httpd 常见配置

- 站点访问控制常见机制

  可基于两种机制指明对哪些资源进行何种访问控制

  访问控制机制有两种：客户端来源地址，用户账号

  - 文件系统路径：

```
<Directory  “/path">
...
</Directory>
<File  “/path/file”  >
...
</File>
<FileMatch  "PATTERN">
...
</FileMatch>
```

Httpd 常见配置

- URL路径：

```
<Location  "">
...
</Location>
<LocationMatch "">
...
</LocationMatch>
```

- 示例：

```
<FilesMatch "\.(gif|jpe?g|png)$">
<Files  “?at.*”  >  通配符
<Location /status>
<LocationMatch "/(extra|special)/data">
```

Httpd 常见配置

- <Directory>中“基于源地址”实现访问控制
  - (1) Options：后跟1个或多个以空白字符分隔的选项列表

    在选项前的+，- 表示增加或删除指定选项

    常见选项：

    Indexes：指明的URL路径下不存在与定义的主页面资源相符的资源文件时，返回索引列表给用户

    FollowSymLinks：允许访问符号链接文件所指向的源文件

    None：全部禁用

    All： 全部允许

Httpd 常见配置

- 示例：

```
<Directory /web/docs>
    Options Indexes FollowSymLinks
</Directory>
<Directory /web/docs/spec>
    Options FollowSymLinks
</Directory>
```

Httpd 常见配置

- (2) AllowOverride

  与访问控制相关的哪些指令可以放在指定目录下的.htaccess（由AccessFileName指定）文件中，覆盖之前的配置指令

  只对<directory>语句有效

  AllowOverride All: .htaccess中所有指令都有效

  AllowOverride None： .htaccess 文件无效

  AllowOverride AuthConfig .htaccess 文件中，除了AuthConfig 其它指令都无法生效

Httpd 常见配置

- 基于IP的访问控制:

  无明确授权的目录，默认拒绝

  允许所有主机访问：Require all granted

  拒绝所有主机访问：Require all denied

  控制特定的IP访问：

  Require ip IPADDR：授权指定来源的IP访问

  Require not ip IPADDR：拒绝特定的IP访问

  控制特定的主机访问：

  Require host HOSTNAME：授权特定主机访问

  Require not host HOSTNAME：拒绝

  HOSTNAME：

  FQDN：特定主机

  domin.tld：指定域名下的所有主机

Httpd 常见配置

- 不能有失败，至少有一个成功匹配才成功，即失败优先

```
<RequireAll>
        Require all granted
        Require not ip 172.16.1.1 拒绝特定IP
</RequireAll>
```

- 多个语句有一个成功，则成功，即成功优先

```
<RequireAny>
        Require all denied
        require ip  172.16.1.1  允许特定IP
</RequireAny>
```

Httpd 常见配置

- 日志设定

  日志类型：

  访问日志

  错误日志

  错误日志：

  ErrorLog logs/error_log

  LogLevel warn

  LogLevel 可选值: debug, info, notice, warn,error, crit, alert, emerg

Httpd 常见配置

- 访问日志：
  - 定义日志格式：LogFormat format strings

    LogFormat "%h %l %u %{%F %T}t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" testlog
  - 使用日志格式：

    CustomLog logs/access_log testlog

  参考帮助：
  http://httpd.apache.org/docs/2.4/mod/mod_log_config.html#formats
  - %h 客户端IP地址
  - %l 远程用户,启用mod_ident才有效，通常为减号“-”
  - %u 验证（basic，digest）远程用户,非登录访问时，为一个减号“-”

Httpd 常见配置

- %t 服务器收到请求时的时间
- %r First line of request，即表示请求报文的首行；记录了此次请求的“方法”，“URL”以及协议版本
- %>s 响应状态码
- %b 响应报文的大小，单位是字节；不包括响应报文http首部
- %{Referer}i 请求报文中首部“referer”的值；即从哪个页面中的超链接跳转至当前页面的
- %{User-Agent}i 请求报文中首部“User-Agent”的值；即发出请求的应用程序

◆ 设定默认字符集

AddDefaultCharset UTF-8 此为默认值

中文字符集：GBK, GB2312, GB18030

Httpd 常见配置

- 定义路径别名

格式：Alias /URL/ "/PATH/"

DocumentRoot "/www/htdocs"

http://www.magedu.com/download/bash.rpm

==>/www/htdocs/download/bash.rpm

Alias /download/ "/rpms/pub/"

http://www.magedu.com/download/bash.rpm

==>/rpms/pub/bash.rpm

http://www.magedu.com/images/logo.png

==>/www/htdocs/images/logo.png

Httpd 常见配置

- 基于用户的访问控制
  - 认证质询：WWW-Authenticate：响应码为401，拒绝客户端请求，并说明要求客户端提供账号和密码
  - 认证：Authorization：客户端用户填入账号和密码后再次发送请求报文；认证通过时，则服务器发送响应的资源
  - 认证方式两种：

    basic：明文

    digest：消息摘要认证,兼容性差
  - 安全域：需要用户认证后方能访问的路径；应该通过名称对其进行标识，以便于告知用户认证的原因
  - 用户的账号和密码

    虚拟账号：仅用于访问某服务时用到的认证标识

    存储：文本文件，SQL数据库，ldap目录存储，nis等

Httpd 常见配置

- basic认证配置示例：

(1) 定义安全域

```
<Directory “/path">
        Options None
        AllowOverride None
        AuthType Basic
        AuthName "String “
        AuthUserFile "/PATH/HTTPD_USER_PASSWD_FILE"
        Require user username1 username2 ...
</Directory>
```

允许账号文件中的所有用户登录访问：

```
        Require valid-user
```

Httpd 常见配置

- (2) 提供账号和密码存储（文本文件）

  使用专用命令完成此类文件的创建及用户管理

  htpasswd [options] /PATH/HTTPD_PASSWD_FILE username

  -c 自动创建文件，仅应该在文件不存在时使用

  -p 明文密码

  -d CRYPT格式加密，默认

  -m md5格式加密

  -s sha格式加密

  -D 删除指定用户

Httpd 常见配置

- 基于组账号进行认证
  - (1) 定义安全域

```
<Directory “/path">
AuthType Basic
AuthName "String “
AuthUserFile "/PATH/HTTPD_USER_PASSWD_FILE"
AuthGroupFile "/PATH/HTTPD_GROUP_FILE"
Require group grpname1 grpname2 ...
</Directory>
```

  - (2) 创建用户账号和组账号文件

组文件：每一行定义一个组

GRP_NAME: username1 username2 ...

Httpd 常见配置

- 示例：

```
<Directory "/www/htdocs/admin">
    Options None
    AllowOverride None
    AuthType Basic
    AuthName "Administator private"
    AuthUserFile "/etc/httpd/conf.d/.htpasswd"
    AuthGroupFile "/etc/httpd/conf.d/.htgroup"
    Require group webadmins
</Directory>

vim /etc/httpd/conf.d/.htgroup
     webadmins:wang mage
```

Httpd 常见配置

- 远程客户端和用户验证的控制
- Satisfy ALL|Any

  ALL 客户机IP和用户验证都需要通过才可以

  Any客户机IP和用户验证,有一个满足即可
- 示例：

```
Require valid-user
 <RequireAll>
Require all granted
Require not ip 172.16.1.1
</RequireAll>
Satisfy Any
```

Httpd 常见配置

- 实现用户家目录的http共享
- 基于模块mod_userdir.so实现
- 相关设置：

```
vim /etc/httpd/conf.d/userdir.conf
<IfModule mod_userdir.c>
    #UserDir disabled
    UserDir public_html   #指定共享目录的名称
</IfModule>
```

  - 准备目录

```
su – wang;mkdir ~/public_html
setfacl –m u:apache:x ~wang
```

  - 访问

    http://localhost/~wang/index.html

Httpd 常见配置

- ServerSignature On | Off | EMail

  默认值Off，当客户请求的网页并不存在时，服务器将产生错误文档，如果打开了 ServerSignature选项，错误文档的最后一行将包含服务器的名字、Apache的版本等信息，如果不对外显示这些信息，就可以将这个参数设置为Off

  设置为Email，将显示ServerAdmin 的Email提示

- TraceEnable [on|off|extended]

  是否支持trace方法，默认on，基于安全风险，建议关闭

Httpd 常见配置

- status页面

```
LoadModule  status_module  modules/mod_status.so
<Location "/status">
   SetHandler server-status
</Location>
ExtendedStatus On  显示扩展信息
```

Httpd 常见配置

- 虚拟主机
- 站点标识： socket

  IP相同，但端口不同

  IP不同，但端口均为默认端口

  FQDN不同：请求报文中首部 Host: www.magedu.com
- 有三种实现方案：

  基于ip：为每个虚拟主机准备至少一个ip地址

  基于port：为每个虚拟主机使用至少一个独立的port

  基于FQDN：为每个虚拟主机使用至少一个FQDN

Httpd 常见配置

- 虚拟主机的配置方法：

```
<VirtualHost  IP:PORT>
        ServerName FQDN
        DocumentRoot    “/path"
</VirtualHost>
```

建议：上述配置存放在独立的配置文件中

- 其它可用指令：

ServerAlias：虚拟主机的别名；可多次使用

ErrorLog： 错误日志

CustomLog：访问日志

<Directory “/path"> </Directory>

Alias

Httpd 常见配置

- 基于IP的虚拟主机示例：

```
<VirtualHost 172.16.100.6:80>
        ServerName www.a.com
        DocumentRoot "/www/a.com/htdocs"
</VirtualHost>
<VirtualHost 172.16.100.7:80>
        ServerName www.b.net
        DocumentRoot "/www/b.net/htdocs"
</VirtualHost>
<VirtualHost 172.16.100.8:80>
        ServerName www.c.org
        DocumentRoot "/www/c.org/htdocs"
</VirtualHost>
```

Httpd 常见配置

- 基于端口的虚拟主机：

```
listen 808
listen 8080
<VirtualHost 172.16.100.6:80>
        ServerName www.a.com
        DocumentRoot "/www/a.com/htdocs"
</VirtualHost>
<VirtualHost 172.16.100.6:808>
        ServerName www.b.net
        DocumentRoot "/www/b.net/htdocs"
</VirtualHost>
<VirtualHost 172.16.100.6:8080>
        ServerName www.c.org
        DocumentRoot "/www/c.org/htdocs"
</VirtualHost>
```

Httpd 常见配置

- 基于FQDN虚拟主机

  基于FQDN的虚拟主机不再需要NameVirutalHost指令

```
<VirtualHost *:80>
        ServerName www.b.net
        DocumentRoot "/apps/b.net/htdocs"
        <Directory "/apps/b.net/htdocs">
                Options None
                AllowOverride None
                Require all granted
        </Directory>
</VirtualHost>
```

- 注意：任意目录下的页面只有显式授权才能被访问

  三种方式的虚拟主机可以混和使用

mod_deflate模块

- 使用mod_deflate模块压缩页面优化传输速度
- 适用场景：

(1) 节约带宽，额外消耗CPU；同时，可能有些较老浏览器不支持

(2) 压缩适于压缩的资源，例如文本文件

```
LoadModule deflate_module modules/mod_deflate.so SetOutputFilter DEFLATE
SetOutputFilter DEFLATE
# Restrict compression to these MIME types
AddOutputFilterByType DEFLATE text/plain
AddOutputFilterByType DEFLATE text/html
AddOutputFilterByType DEFLATE application/xhtml+xml
AddOutputFilterByType DEFLATE text/xml
AddOutputFilterByType DEFLATE application/xml
AddOutputFilterByType DEFLATE application/x-javascript
AddOutputFilterByType DEFLATE text/javascript
AddOutputFilterByType DEFLATE text/css
```

mod_deflate模块

- Level of compression (Highest 9 - Lowest 1)
- DeflateCompressionLevel 9
- 排除特定旧版本的浏览器，不支持压缩

  Netscape 4.x 只压缩text/html

  BrowserMatch ^Mozilla/4 gzip-only-text/html

  Netscape 4.06-08三个版本 不压缩

  BrowserMatch ^Mozilla/4\.0[678] no-gzip

  Internet Explorer标识本身为“Mozilla / 4”，但实际上是能够处理请求的压缩。如果用户代理首部匹配字符串“MSIE”（“B”为单词边界”），就关闭之前定义的限制

  BrowserMatch \bMSI[E] !no-gzip !gzip-only-text/html

https

- https：http over ssl
- SSL会话的简化过程

  (1) 客户端发送可供选择的加密方式，并向服务器请求证书

  (2) 服务器端发送证书以及选定的加密方式给客户端

  (3) 客户端取得证书并进行证书验证

  如果信任给其发证书的CA

  (a) 验证证书来源的合法性；用CA的公钥解密证书上数字签名

  (b) 验证证书的内容的合法性：完整性验证

  (c) 检查证书的有效期限

  (d) 检查证书是否被吊销

  (e) 证书中拥有者的名字，与访问的目标主机要一致

  (4) 客户端生成临时会话密钥（对称密钥），并使用服务器端的公钥加密此数据发送给服务器，完成密钥交换

  (5) 服务用此密钥加密用户请求的资源，响应给客户端

- 注意：SSL是基于IP地址实现,单IP的主机仅可以使用一个https虚拟主机

https实现

- (1) 为服务器申请数字证书
  测试：通过私建CA发证书
    (a) 创建私有CA
    (b) 在服务器创建证书签署请求
    (c) CA签证
- (2) 配置httpd支持使用ssl，及使用的证书
  yum -y install mod_ssl
  配置文件：/etc/httpd/conf.d/ssl.conf
    DocumentRoot
    ServerName
    SSLCertificateFile
    SSLCertificateKeyFile
- (3) 测试基于https访问相应的主机
  openssl s_client [-connect host:port] [-cert filename] [-CApath directory] [-CAfile filename]

http重定向https

- 将http请求转发至https的URL
- 重定向

  Redirect [status] URL-path URL
- status状态：
  - permanent： 返回永久重定向状态码 301
  - temp：返回临时重定向状态码302. 此为默认值
- 示例：

  Redirect temp / https://www.magedu.com/

HSTS

- HSTS:HTTP Strict Transport Security

  服务器端配置支持HSTS后，会在给浏览器返回的HTTP首部中携带HSTS字段。浏览器获取到该信息后，会将所有HTTP访问请求在内部做307跳转到HTTPS。而无需任何网络过程

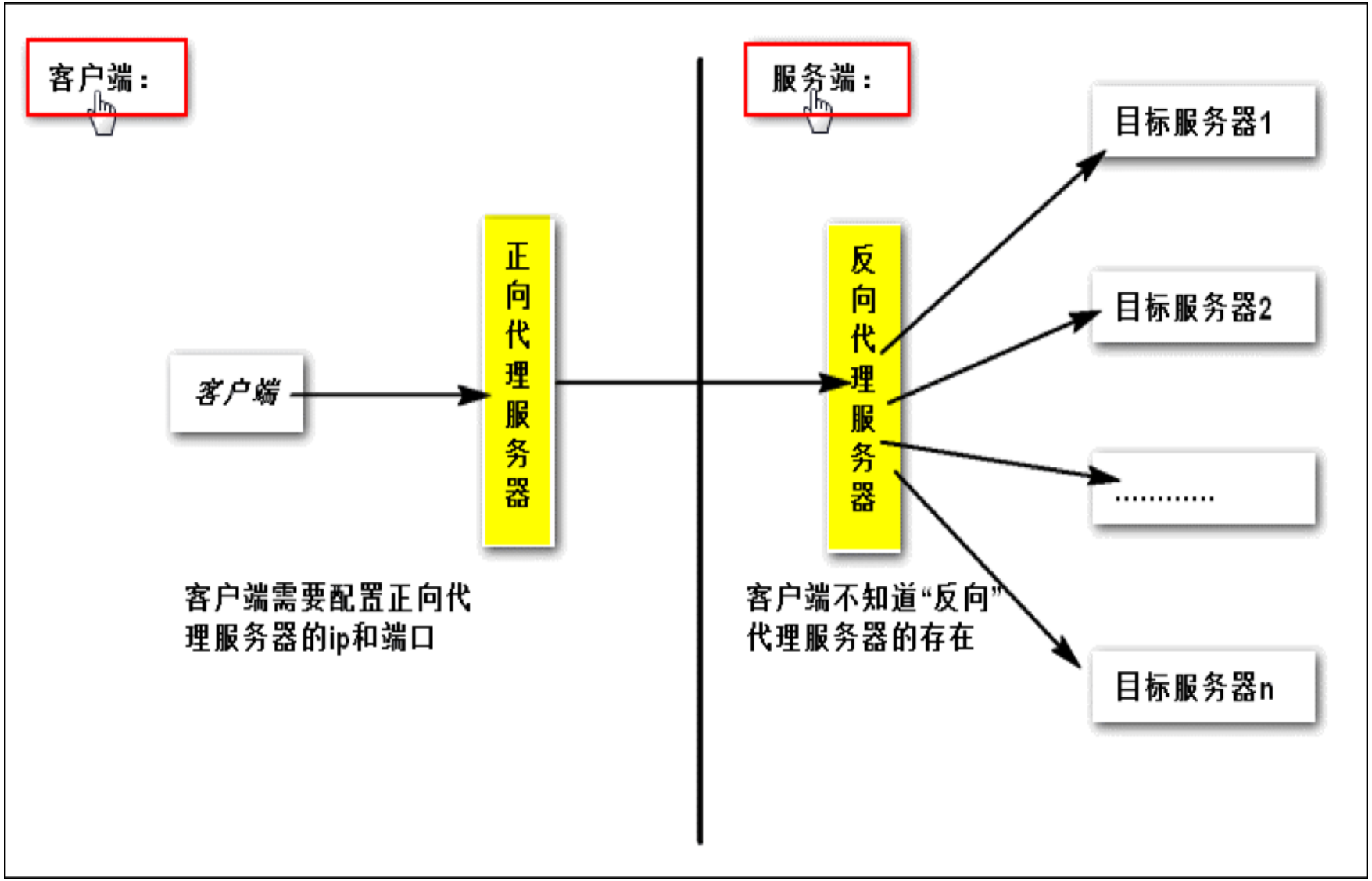
- HSTS preload list

  是Chrome浏览器中的HSTS预载入列表，在该列表中的网站，使用Chrome浏览器访问时，会自动转换成HTTPS。Firefox、Safari、Edge浏览器也会采用这个列表

- 实现HSTS示例：

```
vim /etc/httpd/conf/httpd.conf
Header always set Strict-Transport-Security "max-age=31536000"
RewriteEngine on
RewriteRule ^(/.*)$  https://%{HTTP_HOST}$1 [redirect=302]
```

正向代理和反向代理


客户端：
服务端：
客户端
正向代理服务器
反向代理服务器
目标服务器1
目标服务器2
.....
目标服务器n
客户端需要配置正向代理服务器的ip和端口
客户端不知道“反向”代理服务器的存在

反向代理功能

- 启用反向代理

```
ProxyPass "/"  "http://www.example.com/"
ProxyPassReverse "/"  "http://www.example.com/"
```

- 特定URL反向代理

```
ProxyPass "/images"  "http://www.example.com/"
ProxyPassReverse "/images"  http://www.example.com/
```

- 示例：

```
<VirtualHost *>
    ServerName www.magedu.com
    ProxyPass / http://localhost:8080/
    ProxyPassReverse / http://localhost:8080/
</VirtualHost>
```

Sendfile机制

- 不用 sendfile 的传统网络传输过程：
  - read(file, tmp_buf, len)
  - write(socket, tmp_buf, len)
- 硬盘 >> kernel buffer >> user buffer >> kernel socket buffer >> 协议栈
- 一般网络应用通过读硬盘数据，写数据到 socket 来完成网络传输,底层执行过程：
- 1 系统调用 read() 产生一个上下文切换：从 user mode 切换到 kernel mode，然后 DMA 执行拷贝，把文件数据从硬盘读到一个 kernel buffer 里。
- 2 数据从 kernel buffer 拷贝到 user buffer，然后系统调用 read() 返回，这时又产生一个上下文切换：从kernel mode 切换到 user mode
- 3 系统调用 write() 产生一个上下文切换：从 user mode 切换到 kernel mode，然后把步骤2读到 user buffer 的数据拷贝到 kernel buffer（数据第2次拷贝到 kernel buffer），不过这次是个不同的 kernel buffer，这个 buffer和 socket 相关联。
- 4 系统调用 write() 返回，产生一个上下文切换：从 kernel mode 切换到 user mode(第4次切换),然后DMA从 kernel buffer 拷贝数据到协议栈（第4次拷贝）
- 上面4个步骤有4次上下文切换，有4次拷贝，如能减少切换次数和拷贝次数将会有效提升性能

Sendfile机制

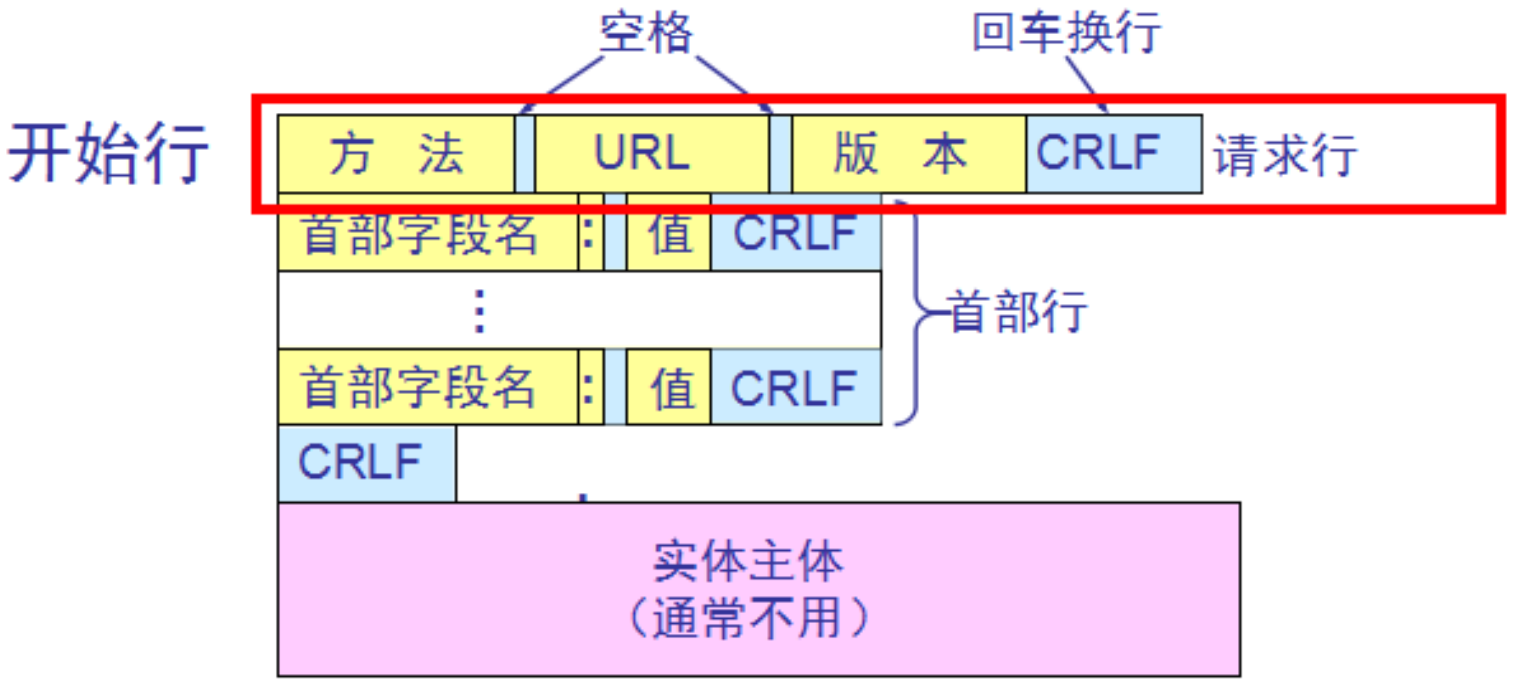
- 在kernel 2.0+ 版本中，系统调用 sendfile() 就是用来简化上面步骤提升性能的。sendfile() 不但能减少切换次数而且还能减少拷贝次数
- 用 sendfile() 来进行网络传输的过程：
  - sendfile(socket, file, len);
  - 硬盘 >> kernel buffer (快速拷贝到kernel socket buffer) >> 协议栈
- 1 系统调用 sendfile() 通过 DMA 把硬盘数据拷贝到 kernel buffer，然后数据被 kernel 直接拷贝到另外一个与 socket 相关的 kernel buffer。这里没有 user mode 和 kernel mode 之间的切换，在 kernel 中直接完成了从一个 buffer 到另一个 buffer 的拷贝
- 2 DMA 把数据从 kernel buffer 直接拷贝给协议栈，没有切换，也不需要数据从 user mode 拷贝到 kernel mode，因为数据就在 kernel 里

http协议

- http协议
  - http/0.9, http/1.0, http/1.1, http/2.0
- http协议：stateless 无状态
  - 服务器无法持续追踪访问者来源
- 解决http协议无状态方法
  - cookie 客户端存放
  - session 服务端存放
- http事务：一次访问的过程
  - 请求：request
  - 响应：response

HTTP请求报文

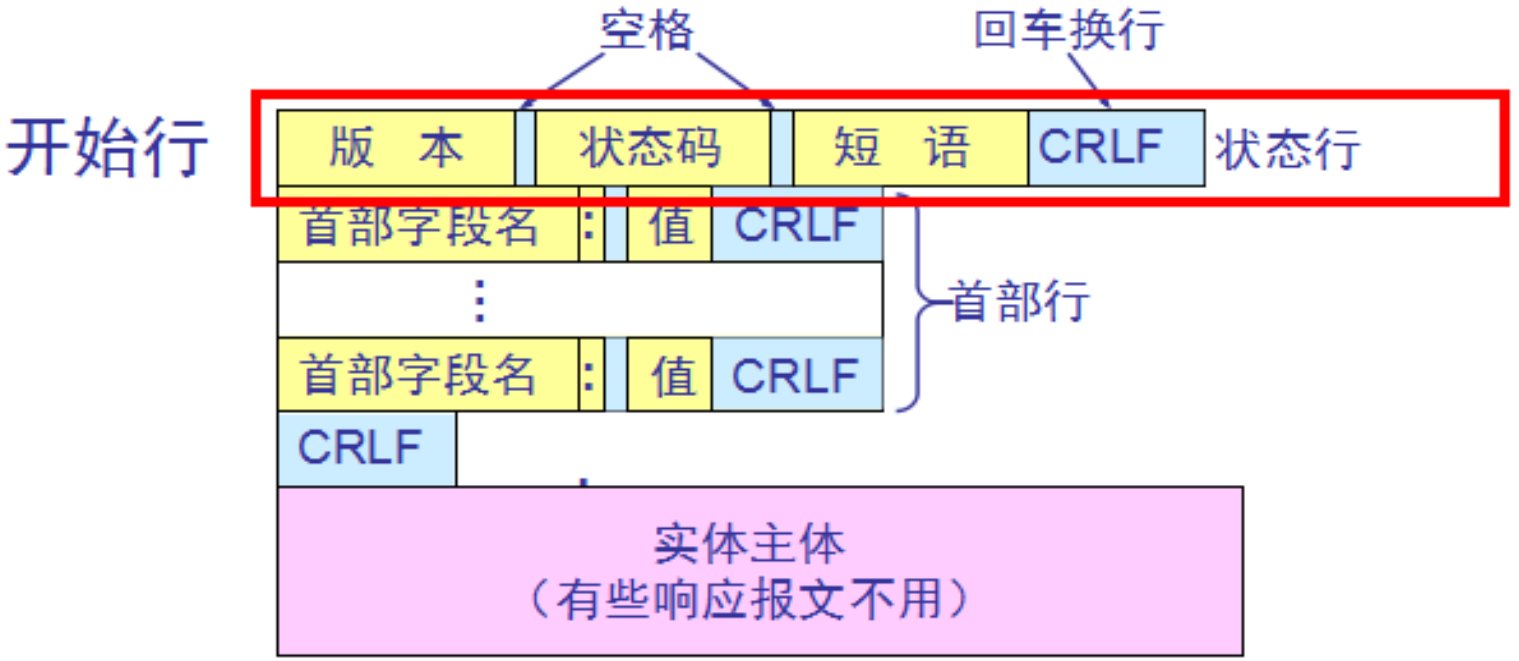
HTTP 的报文结构（请求报文）



报文由三个部分组成，即开始行、首部行和实体主体。
在请求报文中，开始行就是请求行。

HTTP响应报文

HTTP 的报文结构（响应报文）



响应报文的开始行是状态行。
状态行包括三项内容，即 HTTP 的版本，状态码，
以及解释状态码的简单短语。

http协议

- 报文语法格式：
- request报文

  <method> <request-URL> <version>

  <headers>

  <entity-body>
- response报文

  <version> <status> <reason-phrase>

  <headers>

  <entity-body>
- method: 请求方法，标明客户端希望服务器对资源执行的动作

  GET、HEAD、POST等

http协议

- version:

  HTTP/<major>.<minor>

- status:

  三位数字，如200，301, 302, 404, 502; 标记请求处理过程中发生的情况

- reason-phrase：

  状态码所标记的状态的简要描述

- headers：

  每个请求或响应报文可包含任意个首部；每个首部都有首部名称，后面跟一个冒号，而后跟一个可选空格，接着是一个值

- entity-body：请求时附加的数据或响应时附加的数据

http协议

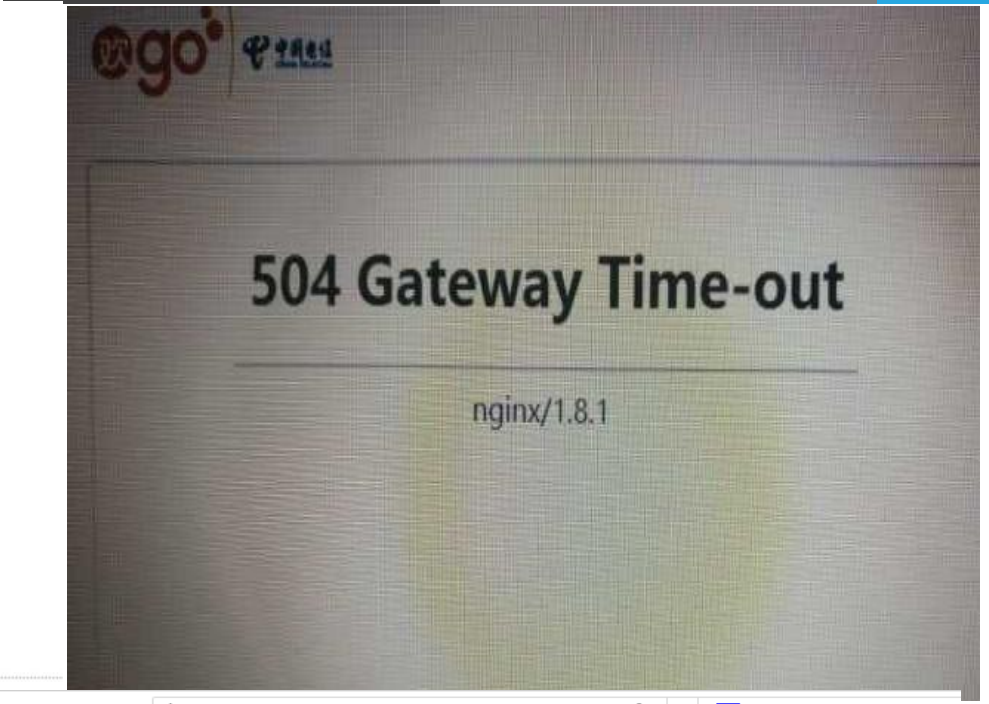
- Method 方法：

  GET：　从服务器获取一个资源
  HEAD：　只从服务器获取文档的响应首部
  POST：　向服务器输入数据，通常会再由网关程序继续处理
  PUT：　将请求的主体部分存储在服务器中，如上传文件
  DELETE：请求删除服务器上指定的文档
  TRACE：追踪请求到达服务器中间经过的代理服务器
  OPTIONS：请求服务器返回对指定资源支持使用的请求方法

- 协议查看或分析的工具：

  tcpdump, wireshark,tshark

http协议状态码分类



504 Gateway Time-out

nginx/1.9.10

http协议状态码分类

- status(状态码)：
- 1xx：100-101　信息提示
- 2xx：200-206　成功
- 3xx：300-305　重定向
- 4xx：400-415　错误类信息，客户端错误
- 5xx：500-505　错误类信息，服务器端错误

http协议常用的状态码

- 200： 成功，请求数据通过响应报文的entity-body部分发送;OK
- 301： 请求的URL指向的资源已经被删除；但在响应报文中通过首部Location指明了资源现在所处的新位置；Moved Permanently
- 302： 响应报文Location指明资源临时新位置 Moved Temporarily
- 304： 客户端发出了条件式请求，但服务器上的资源未曾发生改变，则通过响应此响应状态码通知客户端；Not Modified
- 401： 需要输入账号和密码认证方能访问资源；Unauthorized
- 403： 请求被禁止；Forbidden
- 404： 服务器无法找到客户端请求的资源；Not Found
- 500： 服务器内部错误；Internal Server Error
- 502： 代理服务器从后端服务器收到了一条伪响应，如无法连接到网关；Bad Gateway
- 503： 服务不可用，临时服务器维护或过载，服务器无法处理请求
- 504： 网关超时

http协议

- headers：
- 格式：
- Name: Value

Request URL:http://www.magedu.com/
Request Method:GET
Status Code:200 OK
Remote Address:101.200.188.230:80

Response Headers
view source
Connection:keep-alive
Content-Encoding:gzip
Content-Type:text/html; charset=UTF-8
Date:Sun, 29 Jan 2017 14:32:30 GMT
Server:Tengine
Transfer-Encoding:chunked
Vary:Accept-Encoding

http协议

Request Headers

view source

Accept:text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8

Accept-Encoding:gzip, deflate, sdch

Accept-Language:zh-CN,zh;q=0.8

Cache-Control:max-age=0

Connection:keep-alive

Cookie:53gid2=10104634518015; 53gid0=10104634518015; 53gid1=10104634518015; 53revisit=1485699843851; 53uvid=1; onliner_zdfq72145423=0; CNZZDATA1260642320=1664910013-1485697454-%7C1485697454; visitor_type=old; 53kf_72145423_keyword=; kf_72145423_keyword_ok=1; Hm_lvt_4a78dc1643884da1c990c4c878832e70=1485699844; Hm_lpvt_4a78dc1643884da1c990c4c878832e70=1485700088

Host:www.magedu.com

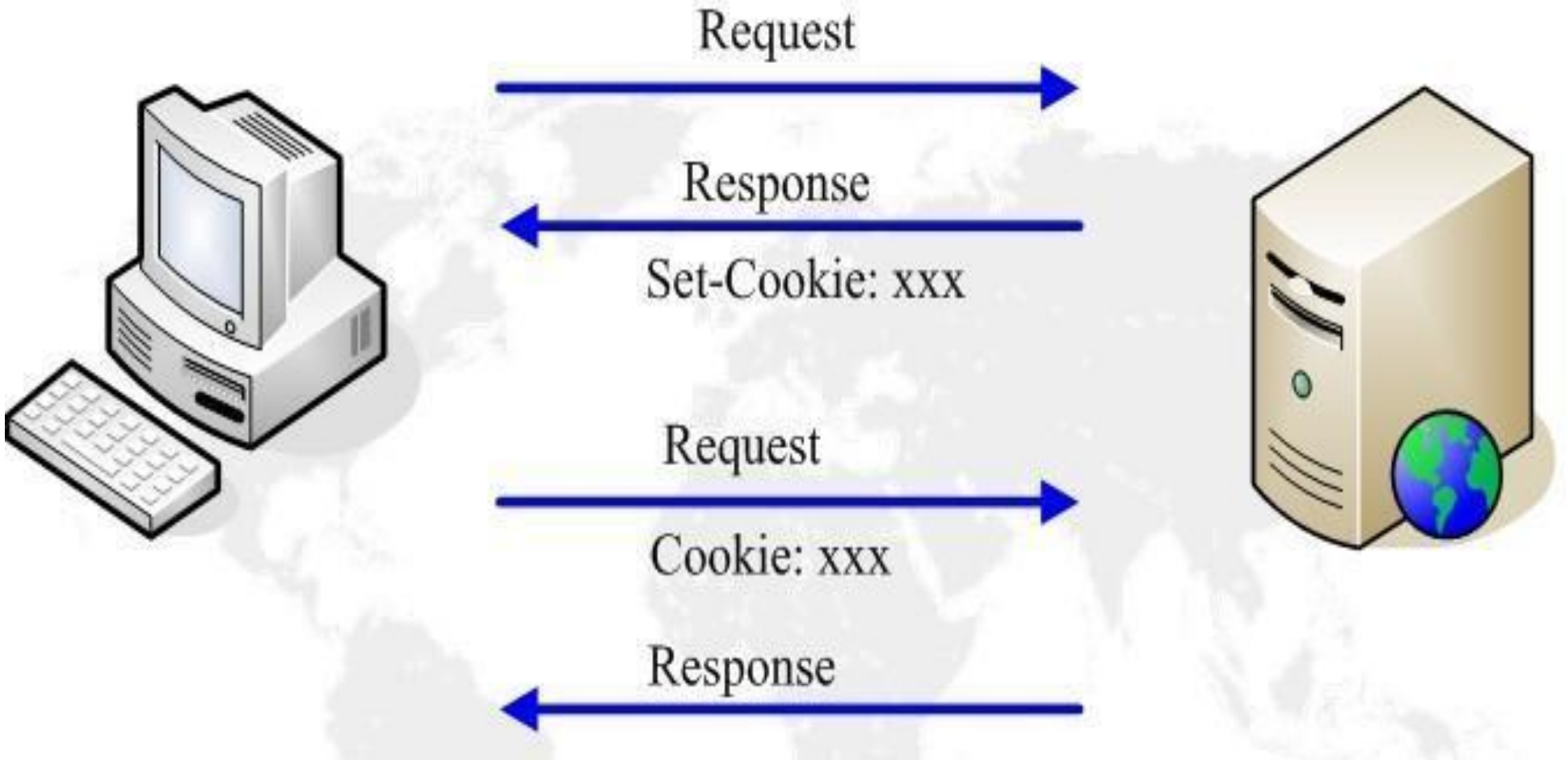
Upgrade-Insecure-Requests:1

User-Agent:Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/56.0.2924.76 Safari/537.36

Cookie

- HTTP 是一种无状态协议。协议自身不对请求和响应之间的通信状态进行保存。也就是说在 HTTP 这个级别，协议对于发送过的请求或响应都不做持久化处理。这是为了更快地处理大量事务，确保协议的可伸缩性，而特意把 HTTP 协议设计成如此简单的。可是随着 Web 的不断发展，很多业务都需要对通信状态进行保存。于是引入了 Cookie 技术。使用 Cookie 的状态管理Cookie 技术通过在请求和响应报文中写入 Cookie 信息来控制客户端的状态。Cookie 会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。服务器端发现客户端发送过来的 Cookie 后，会去检查究竟是从哪一个客户端发来的连接请求，然后对比服务器上的记录，最后得到之前的状态信息

Cookie


Request
Response
Set-Cookie: xxx
Request
Cookie: xxx
Response

Set-Cookie首部字段

- Set-cookie首部字段示例：

  Set-Cookie: status=enable; expires=Fri, 24 Nov 2017 20:30:02 GMT; path=/;

- NAME=VALUE　赋予 Cookie 的名称和其值,此为必需项
- expires=DATE　Cookie 的有效期，若不明确指定则默认为浏览器关闭前为止
- path=PATH　将服务器上的文件目录作为Cookie的适用对象，若不指定则默认为文档所在的文件目录
- domain=域名　作为 Cookie 适用对象的域名，若不指定则默认为创建Cookie的服务器的域名
- Secure　仅在 HTTPS 安全通信时才会发送 Cookie
- HttpOnly　加以限制使 Cookie 不能被 JavaScript 脚本访问

Web相关工具

- links [OPTION]... [URL]...
  - -dump 非交互式模式，显示输出结果
  - -source 打印源码
- wget [OPTION]... [URL]...
  - -q 静默模式
  - -c 断点续传
  - -P /path 保存在指定目录
  - -O filename 保存为指定文件名，filename 为 – 时，发送至标准输出
  - --limit-rate= 指定传输速率，单位K，M等

Web相关工具

- curl工具

curl是基于URL语法在命令行方式下工作的文件传输工具，它支持FTP, FTPS, HTTP, HTTPS, GOPHER, TELNET, DICT, FILE及LDAP等协议。curl支持HTTPS认证，并且支持HTTP的POST、PUT等方法， FTP上传， kerberos认证，HTTP上传，代理服务器，cookies，用户名/密码认证， 下载文件断点续传，上载文件断点续传, http代理服务器管道（ proxy tunneling ），还支持IPv6，socks5代理服务器，通过http代理服务器上传文件到FTP服务器等，功能十分强大

- curl [options] [URL...]

-A/--user-agent <string> 设置用户代理发送给服务器

-e/--referer <URL> 来源网址

--cacert <file> CA证书 (SSL)

-k/--insecure 允许忽略证书进行 SSL 连接

curl工具常用选项

--compressed 要求返回是压缩的格式
-H/--header <line>自定义首部信息传递给服务器
-i　　显示页面内容，包括报文首部信息
-I/--head 只显示响应报文首部信息
-D/--dump-header <file>将url的header信息存放在指定文件中
--basic 使用HTTP基本认证
-u/--user <user[:password]>设置服务器的用户和密码
-L　如果有3xx响应码，重新发请求到新位置
-O　使用URL中默认的文件名保存文件到本地
-o <file>　将网络文件保存为指定的文件中
--limit-rate <rate> 设置传输速度
-0/--http1.0　数字0，使用HTTP 1.0
-v/--verbose　更详细

curl工具常用选项

-C 选项可对文件使用断点续传功能
-c/--cookie-jar <file name> 将url中cookie存放在指定文件中
-x/--proxy <proxyhost[:port]> 指定代理服务器地址
-X/--request <command> 向服务器发送指定请求方法
-U/--proxy-user <user:password> 代理服务器用户和密码
-T 选项可将指定的本地文件上传到FTP服务器上
--data/-d 方式指定使用POST方式传递数据
-b name=data 从服务器响应set-cookie得到值，返回给服务器

httpd自带的工具程序

- httpd自带的工具程序

  htpasswd：basic认证基于文件实现时，用到的账号密码文件生成工具

  apachectl：httpd自带的服务控制脚本，支持start和stop

  rotatelogs：日志滚动工具

  access.log -->

  access.log, access.1.log -->

  access.log, acccess.1.log, access.2.log

httpd的压力测试工具

- httpd的压力测试工具
  - ab, webbench, http_load, seige
  - Jmeter 开源
  - Loadrunner 商业，有相关认证
  - tcpcopy：网易，复制生产环境中的真实请求，并将之保存
  - ab [OPTIONS] URL

    来自httpd-tools包

    -n：总请求数

    -c：模拟的并行数

    -k：以持久连接模式测试

    ulimit –n # 调整能打开的文件数

练习

- 1、建立httpd服务器，要求提供两个基于名称的虚拟主机:

(1)www.X.com，页面文件目录为/web/vhosts/x；错误日志为/var/log/httpd/x.err，访问日志为/var/log/httpd/x.access

(2)www.Y.com，页面文件目录为/web/vhosts/y；错误日志为/var/log/httpd/www2.err，访问日志为/var/log/httpd/y.access

(3)为两个虚拟主机建立各自的主页文件index.html，内容分别为其对应的主机名

(4)通过www.X.com/server-status输出httpd工作状态相关信息

2、为上面的第2个虚拟主机提供https服务，使得用户可以通过https安全的访问此web站点

(1)要求使用证书认证，证书中要求使用的国家(CN)、州(Beijing)、城市(Beijing)和组织(MageEdu)

(2)设置部门为Ops，主机名为www.Y.com，邮件为admin@Y.com

APR

- APR(Apache portable Run-time libraries，Apache可移植运行库) 主要为上层的应用程序提供一个可以跨越多操作系统平台使用的底层支持接口库。在早期的Apache版本中，应用程序本身必须能够处理各种具体操作系统平台的细节，并针对不同的平台调用不同的处理函数
- 随着Apache的进一步开发，Apache组织决定将这些通用的函数独立出来并发展成为一个新的项目。这样，APR的开发就从Apache中独立出来，Apache仅仅是使用 APR而已。目前APR主要还是由Apache使用，由于APR的较好的移植性，因此一些需要进行移植的C程序也开始使用APR，开源项目比如用于服务器压力测试的Flood loader tester，该项目不仅仅适用于Apache，http://httpd.apache.org/test/flood

编译安装httpd-2.4

- 安装httpd-2.4
  - 依赖于apr-1.4+, apr-util-1.4+
  - apr： apache portable runtime，解决跨平台实现
- 安装前准备开发包：
  - 开发环境包：

    相关包： gcc，pcre-devel，openssl-devel，expat-devel
- 下载源代码并解压缩：

  httpd-2.4.39.tar.bz2

  apr-1.7.0.tar.bz2

  apr-util-1.6.1.tar.bz2

编译安装httpd-2.4方法一

- 安装apr

```
cd apr-1.7.0
./configure --prefix=/app/apr
make && make install
```

- 安装apr-util

```
cd ../apr-util-1.6.1
./configure --prefix=/app/apr-util --with-apr=/app/apr/
make -j 2 && make install
```

编译安装httpd-2.4方法一

- 编译安装httpd-2.4

```
cd ../httpd-2.4.39
./configure --prefix=/app/httpd24 \
--enable-so \
--enable-ssl \
--enable-cgi \
--enable-rewrite \
--with-zlib \
--with-pcre \
--with-apr=/app/apr/ \
--with-apr-util=/app/apr-util/ \
--enable-modules=most \
--enable-mpms-shared=all \
--with-mpm=prefork
make -j 4 && make install
```

编译安装httpd-2.4方法二

- mv apr-1.7.0 httpd-2.4.39/srclib/apr
- mv apr-util-1.6.1 httpd-2.4.39/srclib/apr-util
- cd httpd-2.4.39/
- ./configure \
  --prefix=/app/httpd24 \
  --enable-so \
  --enable-ssl \
  --enable-cgi \
  --enable-rewrite \
  --with-zlib \
  --with-pcre \
  --with-included-apr \
  --enable-modules=most \
  --enable-mpms-shared=all \
  --with-mpm=prefork
- make && make install

编译安装httpd-2.4

- Httpd编译过程：/app/httpd24/build/config.nice
- 自带的服务控制脚本：/app/httpd24/bin/apachectl
- 创建用户：useradd -s /sbin/nologin -r apache
- vim /app/httpd24/conf/httpd
  user apache
  group apache
- vim /etc/profile.d/httpd24.sh
  PATH=/app/httpd24/bin:$PATH
- vim /etc/man_db.conf
  MANDATORY_MANPATH /app/httpd24/man
- 设置开机自动启动
  vim /etc/rc.d/rc.local
  /app/httpd24/bin/apachectl start
  chmod +x /etc/rc.d/rc.local

编译安装httpd-2.4

- vim /usr/lib/systemd/system/httpd24.service

```
[Unit]
Description=The Apache HTTP Server
After=network.target remote-fs.target nss-lookup.target
Documentation=man:httpd(8)
Documentation=man:apachectl(8)
[Service]
Type=forking
#EnvironmentFile=/etc/sysconfig/httpd
ExecStart=/app/httpd24/bin/httpd $OPTIONS -k start
ExecReload=/app/httpd24/bin/httpd $OPTIONS -k graceful
ExecStop=/bin/kill -WINCH ${MAINPID}
KillSignal=SIGCONT
PrivateTmp=true
[Install]
WantedBy=multi-user.target
```

练习

- 使用httpd-2.4实现
- 1、建立httpd服务，要求：
  - (1) 提供两个基于名称的虚拟主机：

www.a.com

页面文件目录为/web/vhosts/www1

错误日志为/var/log/httpd/www1/error_log

访问日志为/var/log/httpd/www1/access_log

www.b.com

页面文件目录为/web/vhosts/www2

错误日志为/var/log/httpd/www2/error_log

访问日志为/var/log/httpd/www2/access_log

练习

- (2) 通过www.a.com/server-status输出其状态信息，且要求只允许提供账号的用户访问
- (3) www.a.com不允许192.168.1.0/24网络中的主机访问

◆ 2、为上面的第2个虚拟主机提供https服务，使得用户可以通过https安全的访问此web站点

(1) 要求使用证书认证，证书中要求使用国家（CN），州（Beijing），城市（Beijing），组织为(MageEdu)

(2) 设置部门为Ops, 主机名为www.b.com

关于马哥教育

马哥教育
IT人的高薪职业学院

- 博客：http://mageedu.blog.51cto.com
- 主页：http://www.magedu.com
- QQ：1661815153, 113228115
- QQ群：203585050, 279599283

祝大家学业有成

谢　谢