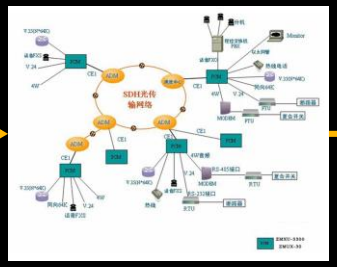
计算机网络基础分享

景东

2017年5月26日

光电转换
光信号传输



电信号
编码

电信号
传输

二极管逻辑电路

01000101 11010100

应用编码

在干吗？

二极管逻辑电路

01000101 11010100

应用解码

在干吗？

发送端

接受端

硬件

操作系统

应用

硬件接口

操作系统接口

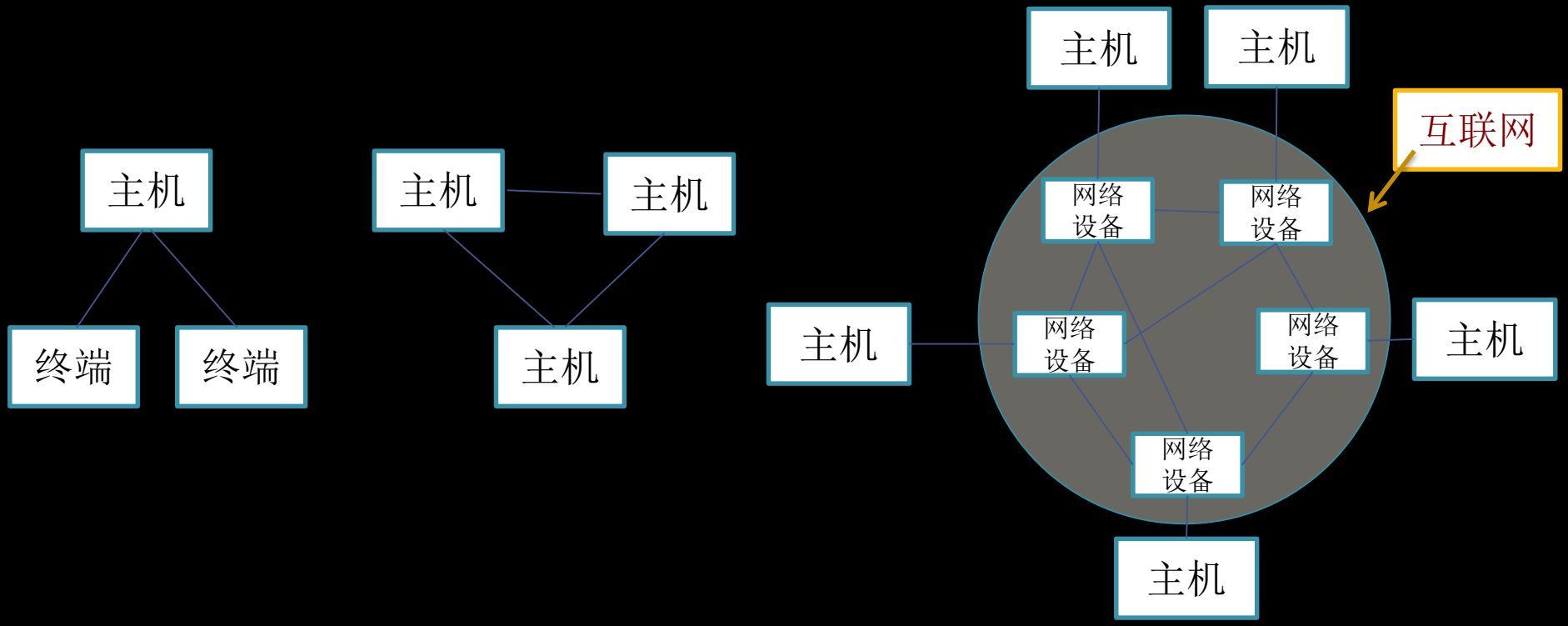
一个典型通信过程

为什么会有计算机网络

- 资源共享
  - 信息资源
  - 计算资源
  - 控制能力

计算机网络的发展

- 主机终端->主机主机->主机与通信网络



互联网的历史

- 美国海陆空军用网络无法互通（DEC、Honeywell、IBM）
- 卡恩 研发ARPANET 来到 UCLA 与 克莱恩洛克 的学生 瑟夫 共同研究开放网络NCP（70年）
- 73年卡恩邀请瑟夫研究普适的协议，74年TCP/IP，83年成为标准

参 TCP/IP协议

为什么要分层

- 分层基于合作共建（分工）
- 标准化，一起玩

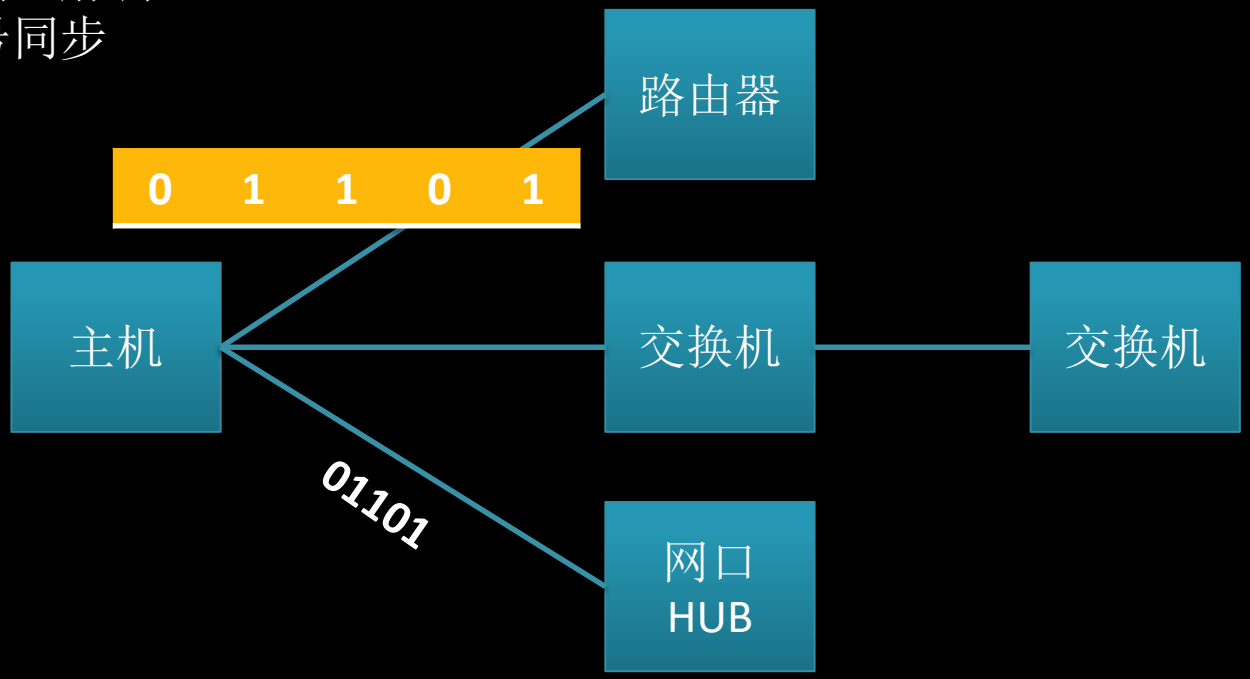
OSI/RM 七层网络模型

应用层	• HTTP
表示层	• GZIP SSL
会话层	• Session
传输层	• 进程，端口
网络层	• 主机间连通，IP
链路层	• 节点间可靠传输，MAC地址
物理层	• 节点间物理信号传输，Bit

物理层

- 节点间物理信号对拷

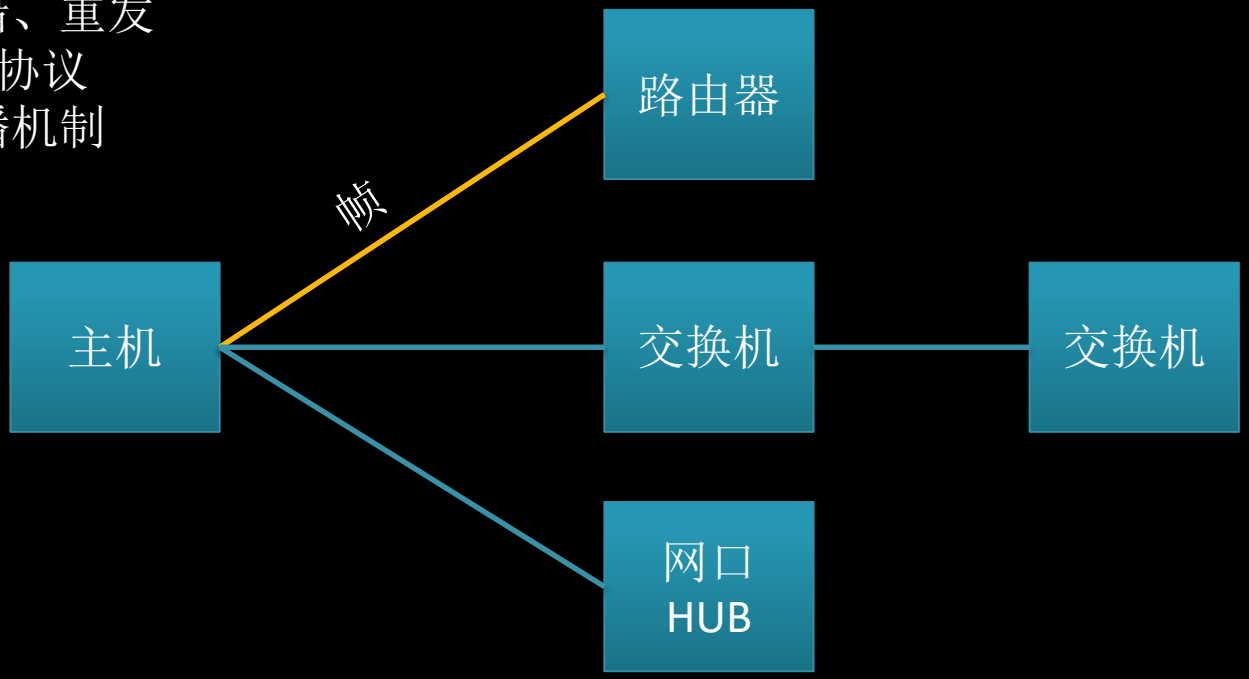
调制、解调
信号同步



数据链路层

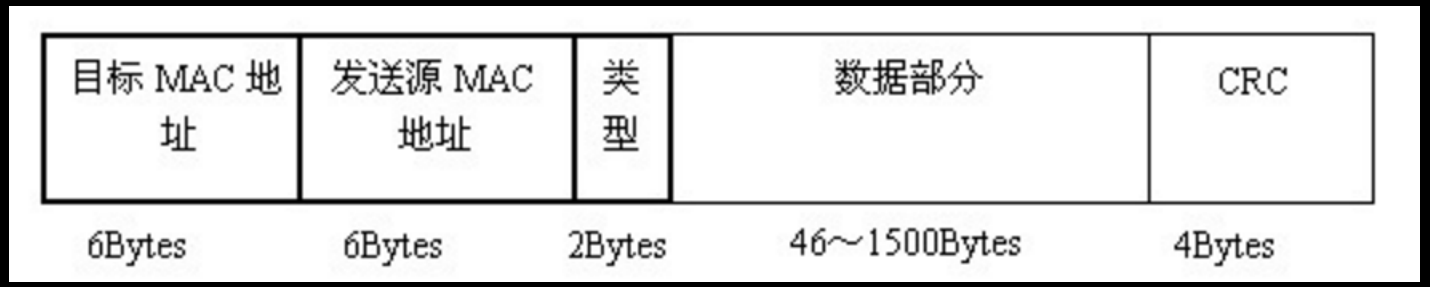
- 节点间可靠传输

MAC地址
容错、重发
ARP协议
广播机制



目的MAC	源MAC	帧数据	FCS

链路帧格式

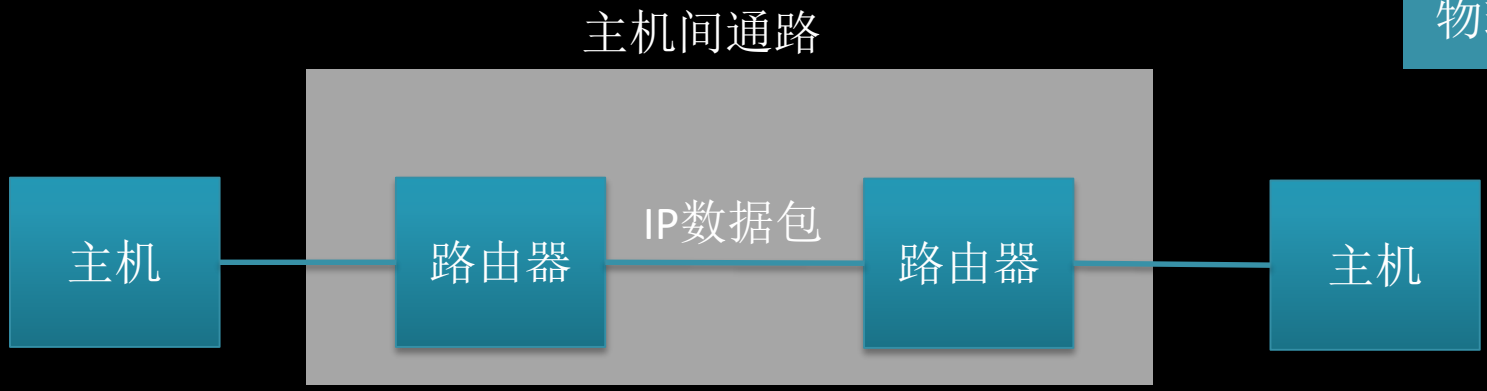


以太网的最大传输单元（MTU）为1500字节

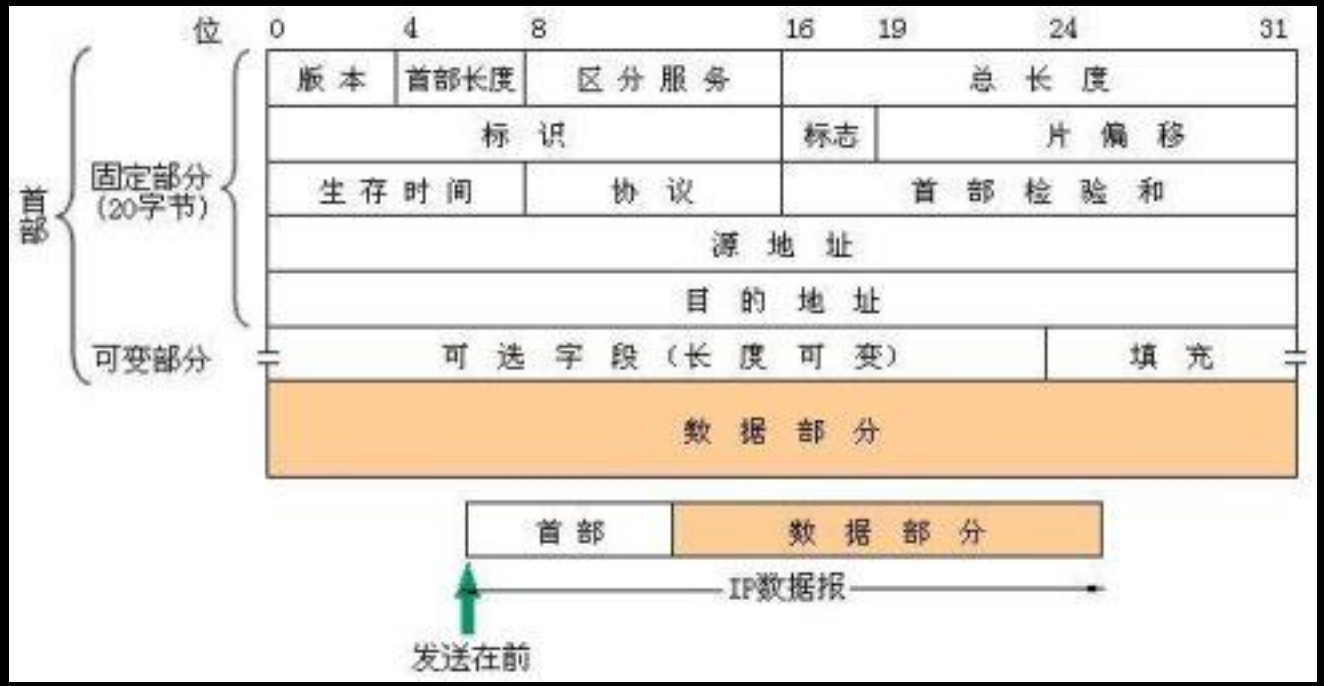
网络层

- 主机间连通，路由寻址、转发

IP地址
路由寻址、转发



IP数据报格式

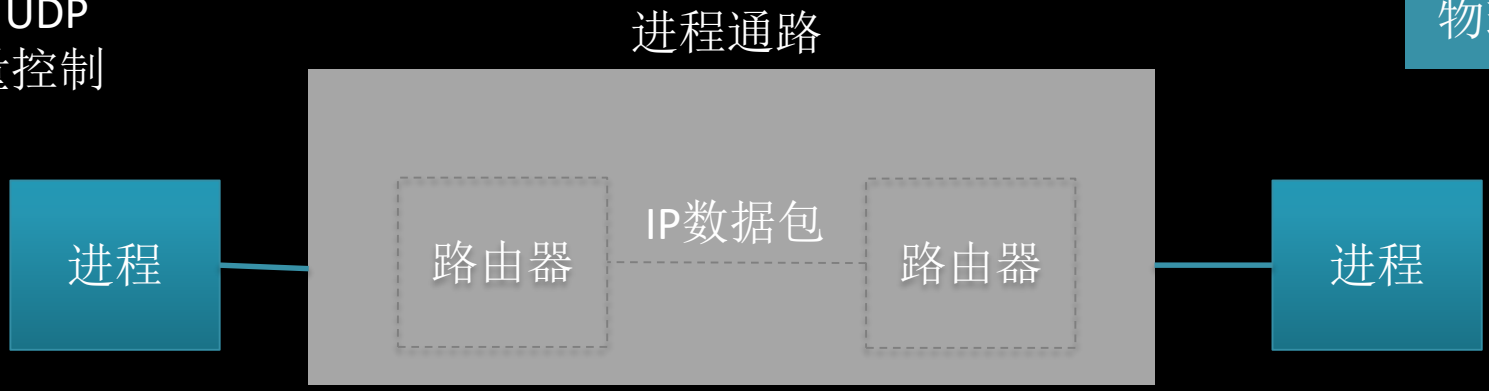


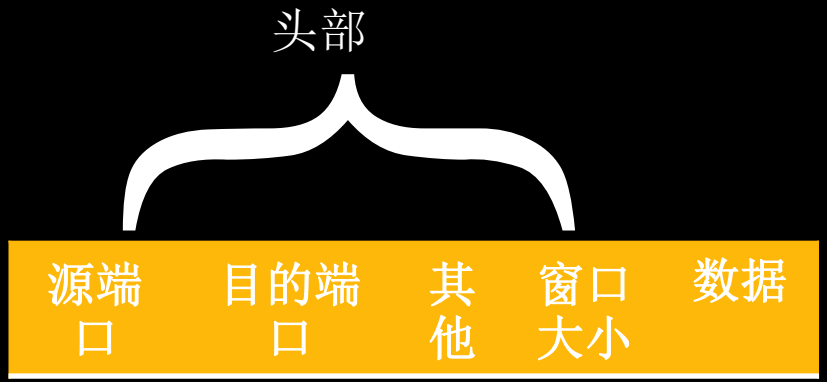
IP报文最大长度65535字节

传输层

- 进程间可靠传输

端口
容错、重发
TCP, UDP
流量控制





TCP段格式

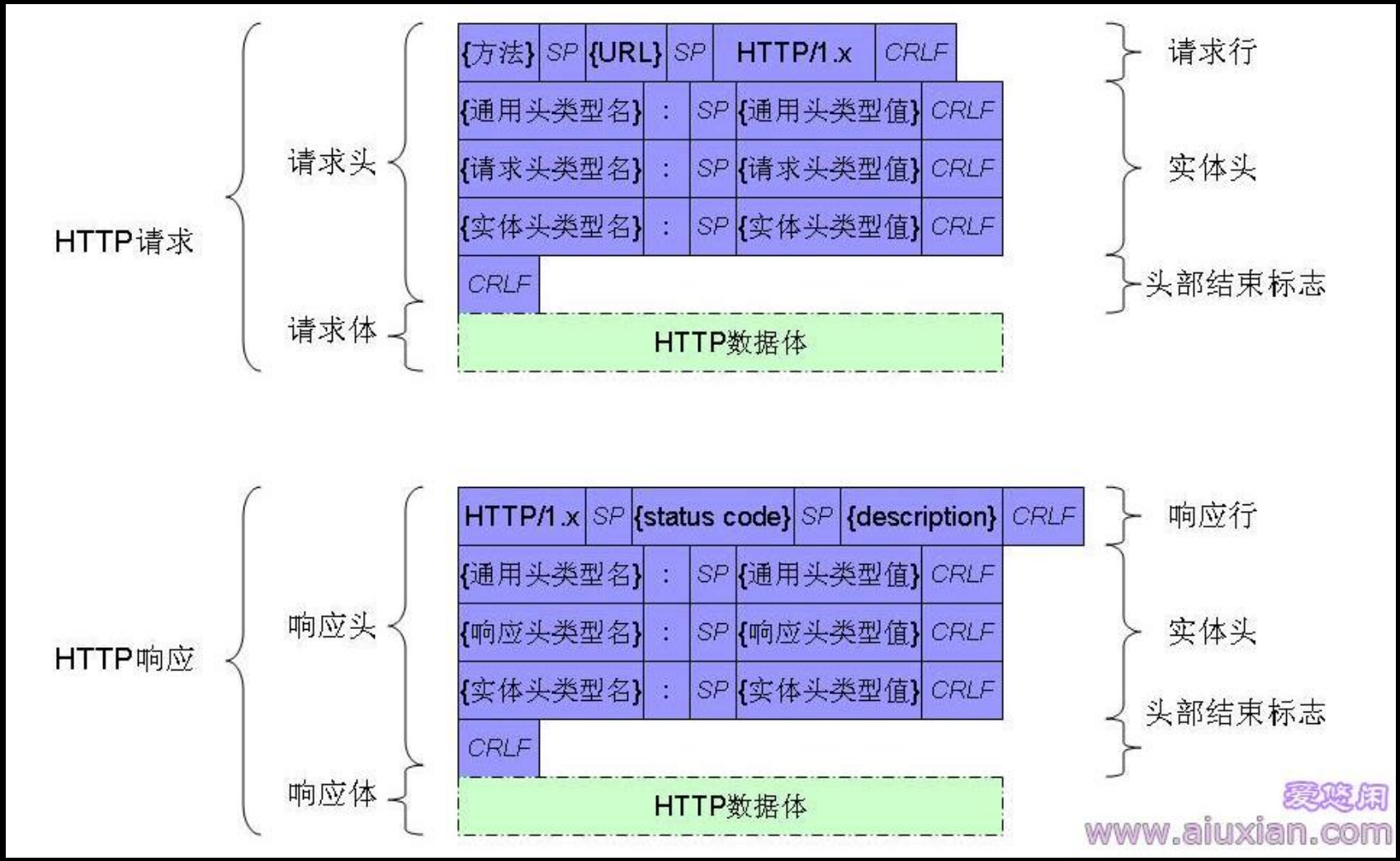

00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 Bit
源端口号 16位
目标端口号 16位
顺序号 32位序列编号
确认号 32位确认编号
数据偏移 4位
6位
头部长度
保留 6位
URG
ACK
PSH
RST
SYN
FIN
窗口大小 16位
校验和 16位
紧急指针 16位
可选项 8的倍数位
数据
20字节

会话层、表示层、应用层

- 程序员控制范围 如：HTTP协议
  - 会话
  - 加解密、压缩解压
  - 应用层数据

参 HTTP协议

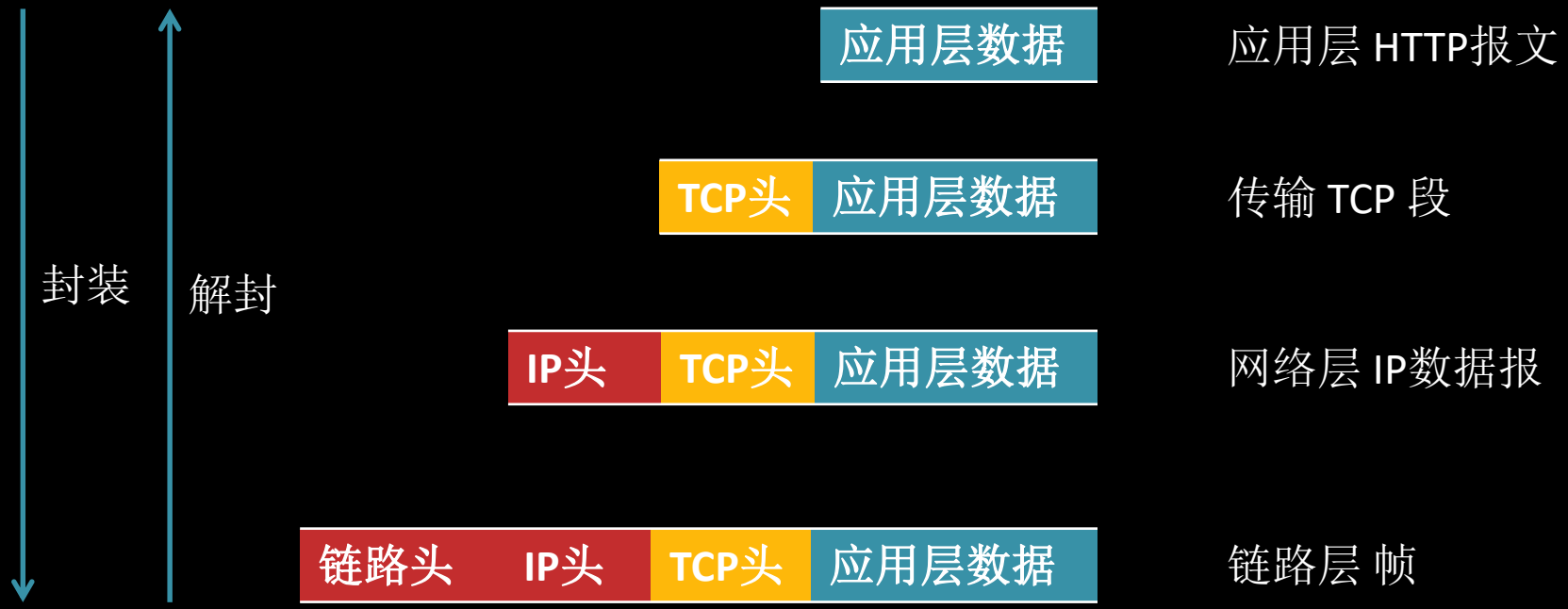
HTTP协议

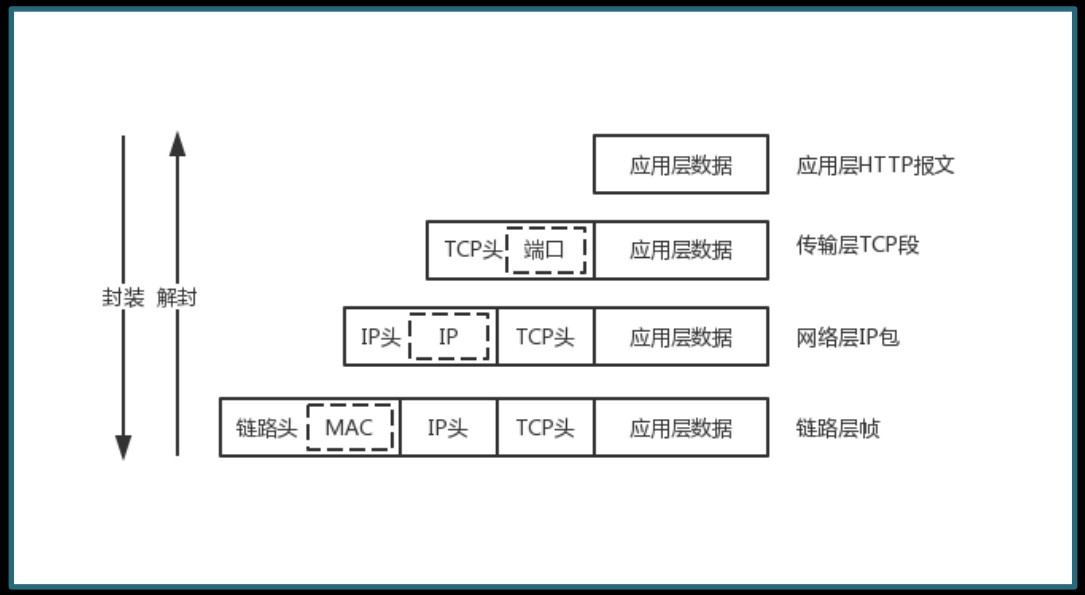


TCP/IP网络模型

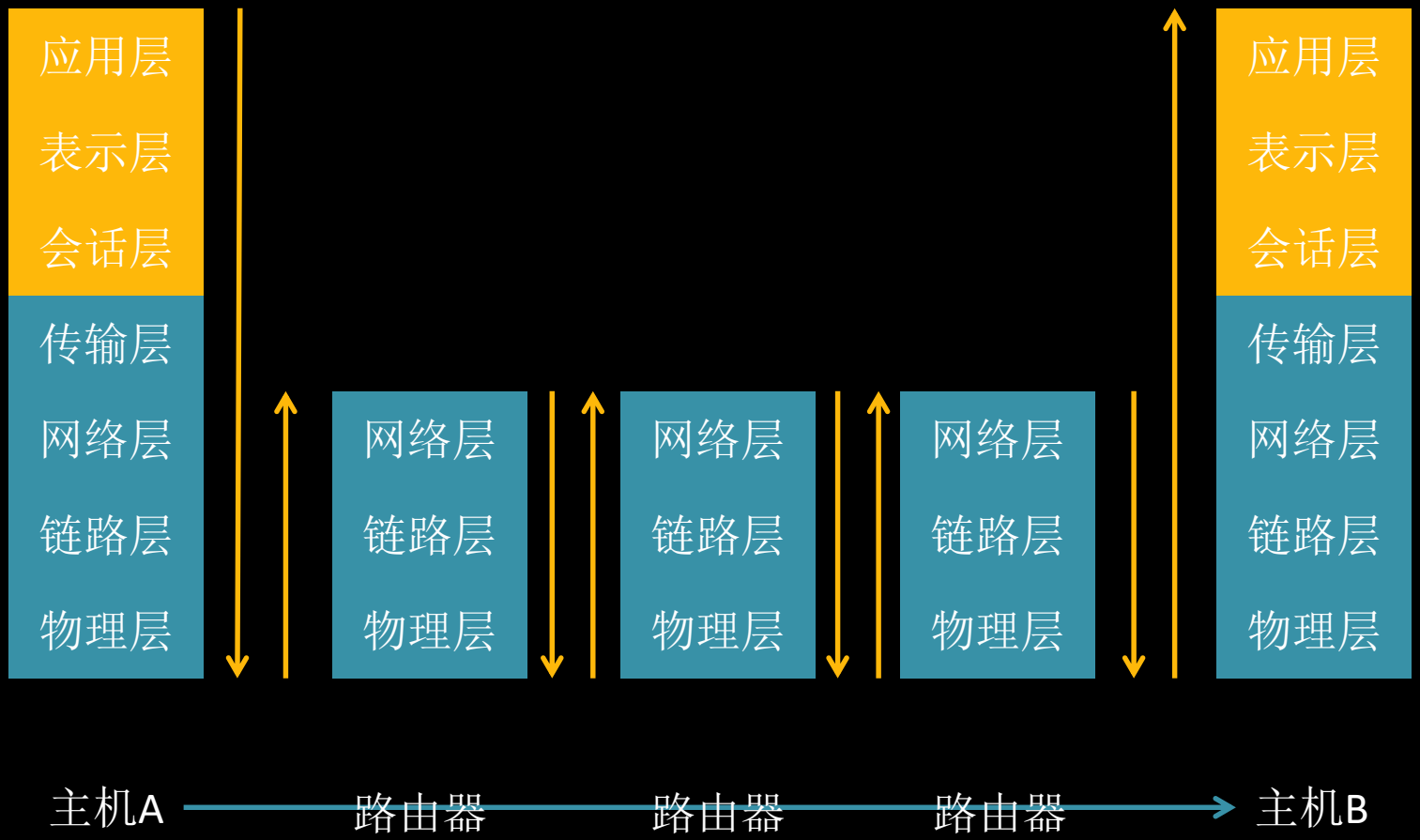

应用层
应用层
• HTTP
表示层
• GZIP SSL
会话层
• Session
传输层
• 进程，端口
网络层
• 主机间连通，IP
链路层
• 节点间可靠传输，MAC地址
网络接口层
物理层
• 节点间物理信号传输，Bit

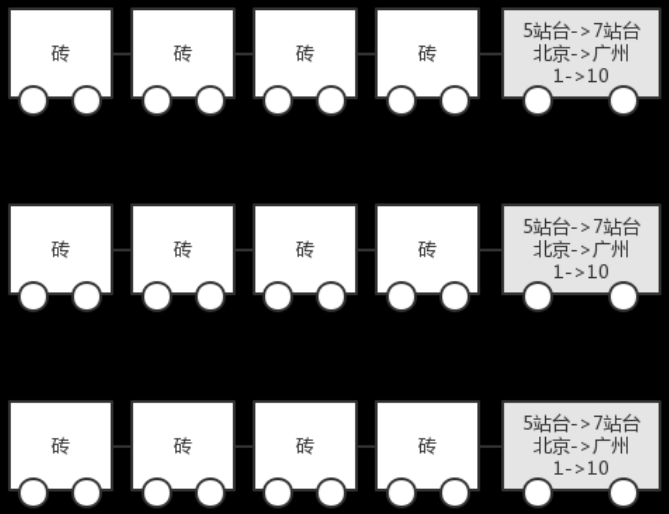
TCP/IP 数据封装


封装
解封
应用层数据
应用层 HTTP报文
TCP头
应用层数据
传输 TCP 段
IP头
TCP头
应用层数据
网络层 IP数据报
链路头
IP头
TCP头
应用层数据
链路层 帧


封装
解封
应用层数据
应用层HTTP报文
TCP头
端口
应用层数据
传输层TCP段
IP头
IP
TCP头
应用层数据
网络层IP包
链路头
MAC
IP头
TCP头
应用层数据
链路层帧

一个网络请求的通信过程



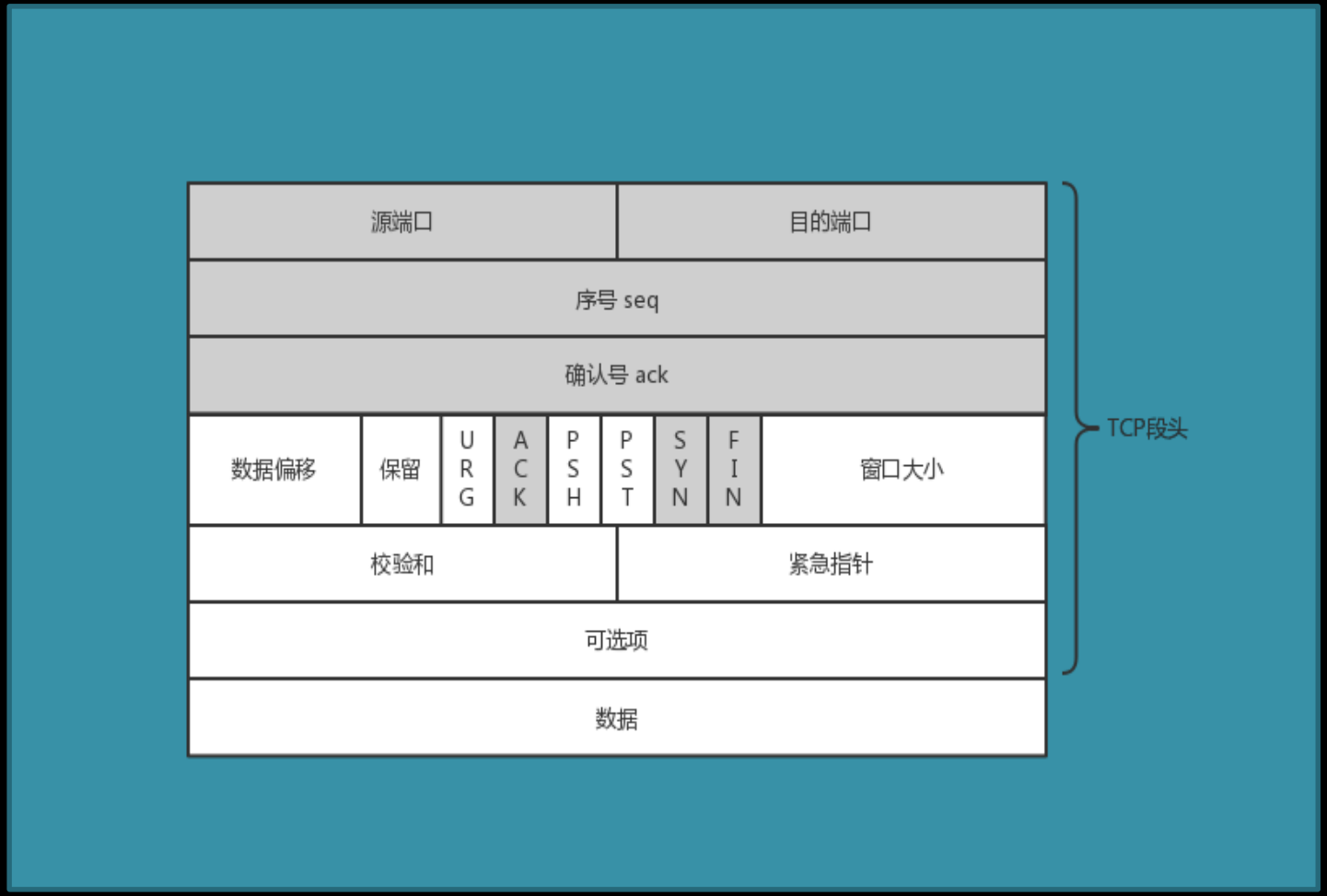

砖
砖
砖
砖
5站台->7站台
北京->广州
1->10
砖
砖
砖
砖
5站台->7站台
北京->广州
1->10
砖
砖
砖
砖
5站台->7站台
北京->广州
1->10

北京 石家庄 ..... 广州

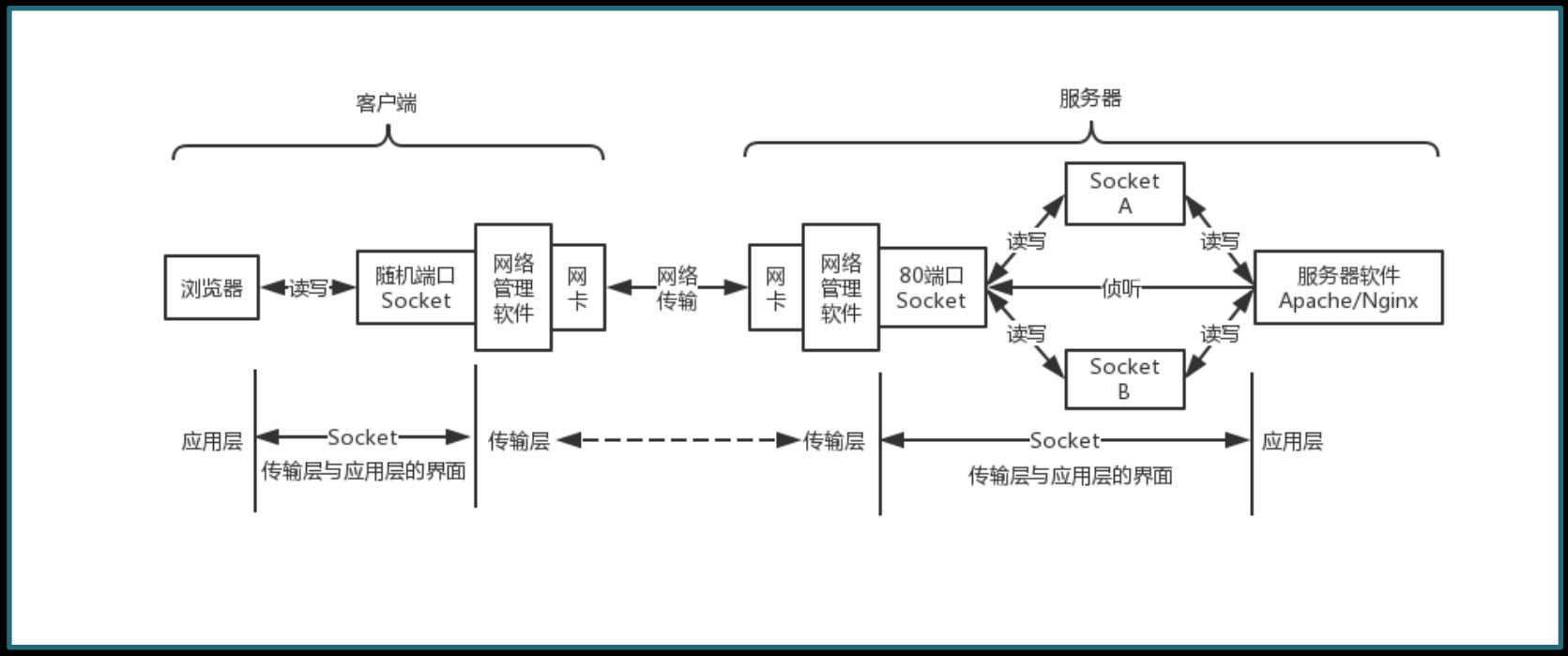
TCP协议

- 从火车运输的例子我们知道了，一列火车是如何从起点运到终点的，但是如果很多站点同时向广州站运输怎么办？
- 如果中途一列火车失踪了怎么办？
- 同样，对于HTTP服务器，同时有很多客户端发起请求，服务器如何知道这些数据属于哪个客户端，又该返回数据给哪个客户端？
- 数据来了如何交给应用层？
- 由于HTTP协议是建立在TCP协议之上的，所以我们有必要进一步了解TCP协议。

TCP段格式

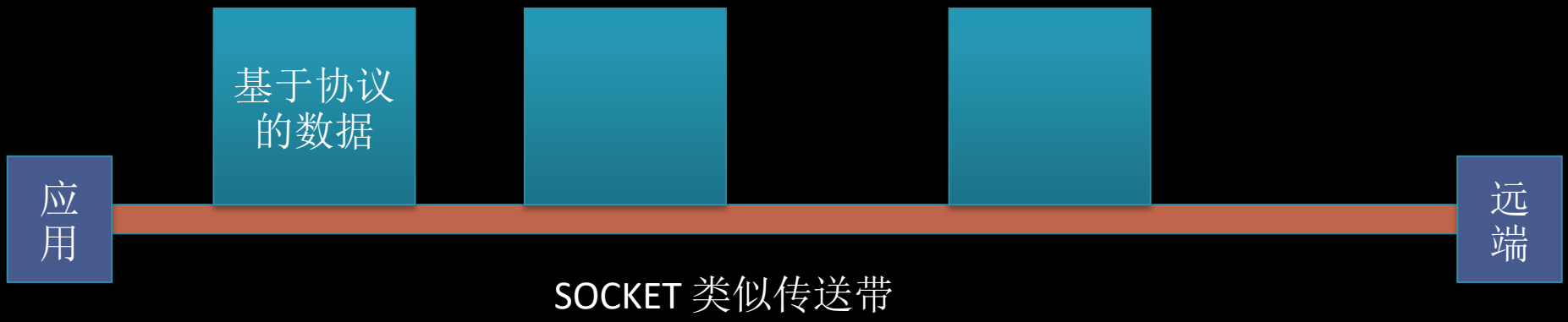

源端口
目的端口
序号 seq
确认号 ack
数据偏移
保留
URG
ACK
PSH
PST
SYN
FIN
窗口大小
校验和
紧急指针
可选项
数据
TCP段头

SOCKET


客户端
服务器
浏览器
读写
随机端口
Socket
网络
管理
软件
网
卡
网络
传输
网
卡
网络
管理
软件
80端口
Socket
Socket
A
Socket
B
读写
读写
读写
读写
侦听
服务器软件
Apache/Nginx
应用层
Socket
传输层与应用层的界面
传输层
传输层
Socket
传输层与应用层的界面
应用层

SOCKET 与 TCP和UDP

- SOCKET套接字，实现两端之间的管道
- SOCKET打通了网络层与应用层，基于TCP和UDP的SOCKET使得应用可以将远程数据当作本地文件一样读写。



SOCKET原语

原语名	功能
SOCKET	创建一个新的socket
BIND	将socket与本地IP和端口绑定
LISTEN	等待连接
ACCEPT	接受连接，接受后会创建一个新的socket
CONNECT	主动请求建立一个连接
SEND	向指定的socket连接发送数据
RECV	从指定的socket读取数据
CLOSE	释放指定的连接

扩展阅读

- 基于TCP/UDP的SOCKET编程，实现从网络层到应用层的连通，由通信学科向软件学科的桥梁。

- 阻塞与非阻塞SOCKET通信 资料

TCP 协议

- 传输层端到端面向连接的可靠的协议
- 应用层报文分段，添加TCP头（端口）
- 每一个段发出去，必须等待目标主机的确认，才能表示成功，否则重发

UDP

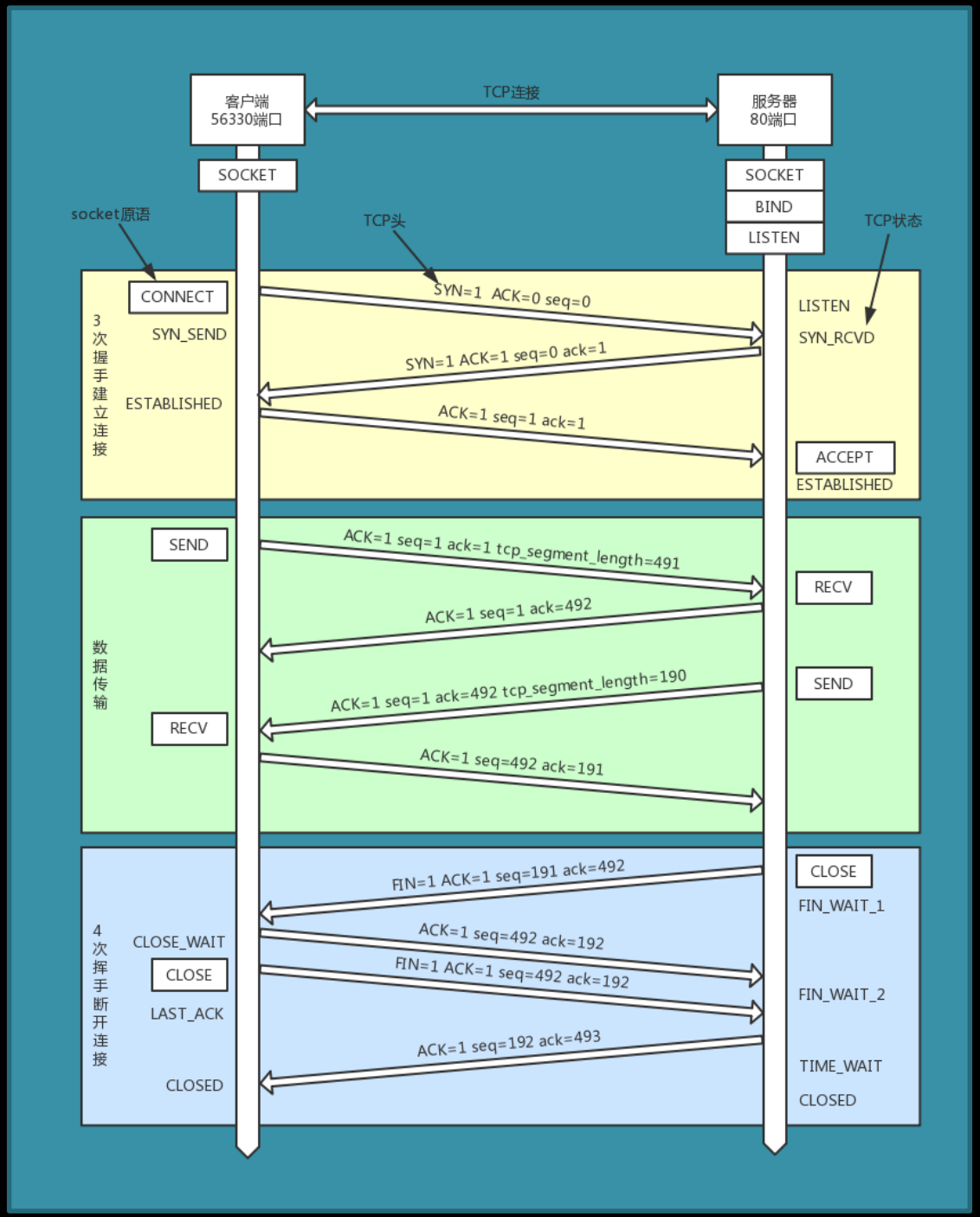
- 无连接不保证数据传输（通常应用层保证）
- 无需对方确认
- 适用于音视频传输
- UDP成为IP协议的傀儡协议，仅仅实现了端口封装，不再进行数据分段。但其最大的作用是UDP复用。

参 UDP协议

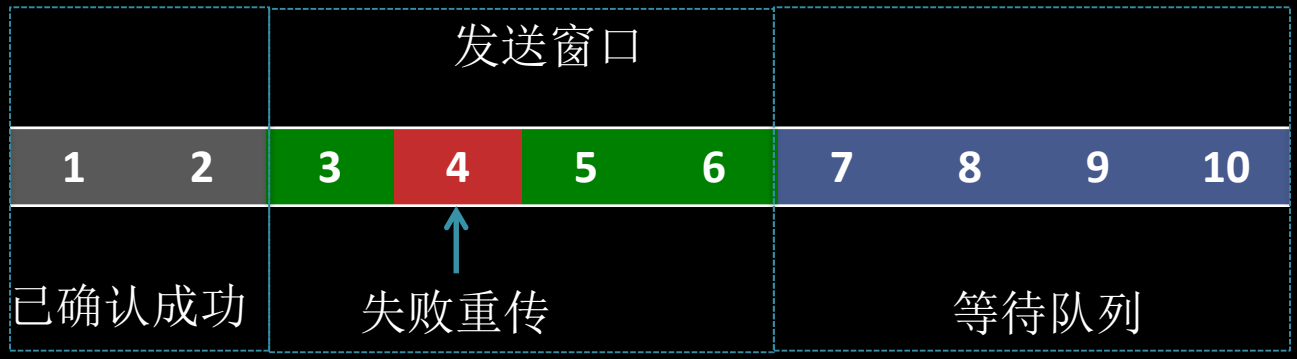
TCP传输

- TCP传输分为三个阶段，建立连接、数据传输和释放连接。建立连接需要三次握手，连接建立后才能传输数据，释放连接需要四次挥手。下图描述了TCP传输的三个阶段，下面通过一个HTTP请求来分别解释这三个阶段。
    - 客户端ip和端口：192.168.1.151:56330
    - 服务器ip和端口：192.168.1.179:80
    - 请求地址：http://192.168.1.179/todo/public/js/todo.js

Tcp连接


客户端
56330端口
TCP连接
服务器
80端口
SOCKET
SOCKET
BIND
LISTEN
socket原语
TCP头
TCP状态
CONNECT
SYN_SEND
ESTABLISHED
SYN=1 ACK=0 seq=0
SYN=1 ACK=1 seq=0 ack=1
ACK=1 seq=1 ack=1
LISTEN
SYN_RCVD
ACCEPT
ESTABLISHED
3次握手建立连接
SEND
RECV
ACK=1 seq=1 ack=1 tcp_segment_length=491
ACK=1 seq=1 ack=492
ACK=1 seq=1 ack=492 tcp_segment_length=190
ACK=1 seq=492 ack=191
RECV
SEND
数据传输
FIN=1 ACK=1 seq=191 ack=492
ACK=1 seq=492 ack=192
FIN=1 ACK=1 seq=492 ack=192
ACK=1 seq=192 ack=493
CLOSE_WAIT
CLOSE
LAST_ACK
CLOSED
CLOSE
FIN_WAIT_1
FIN_WAIT_2
TIME_WAIT
CLOSED
4次挥手断开连接

滑动窗口协议



接受窗口

已读入应用空间

参 滑动窗口协议

滑动窗口 拥塞控制

- 利用滑动窗口协商窗口大小
- 利用慢开始算法尝试网速

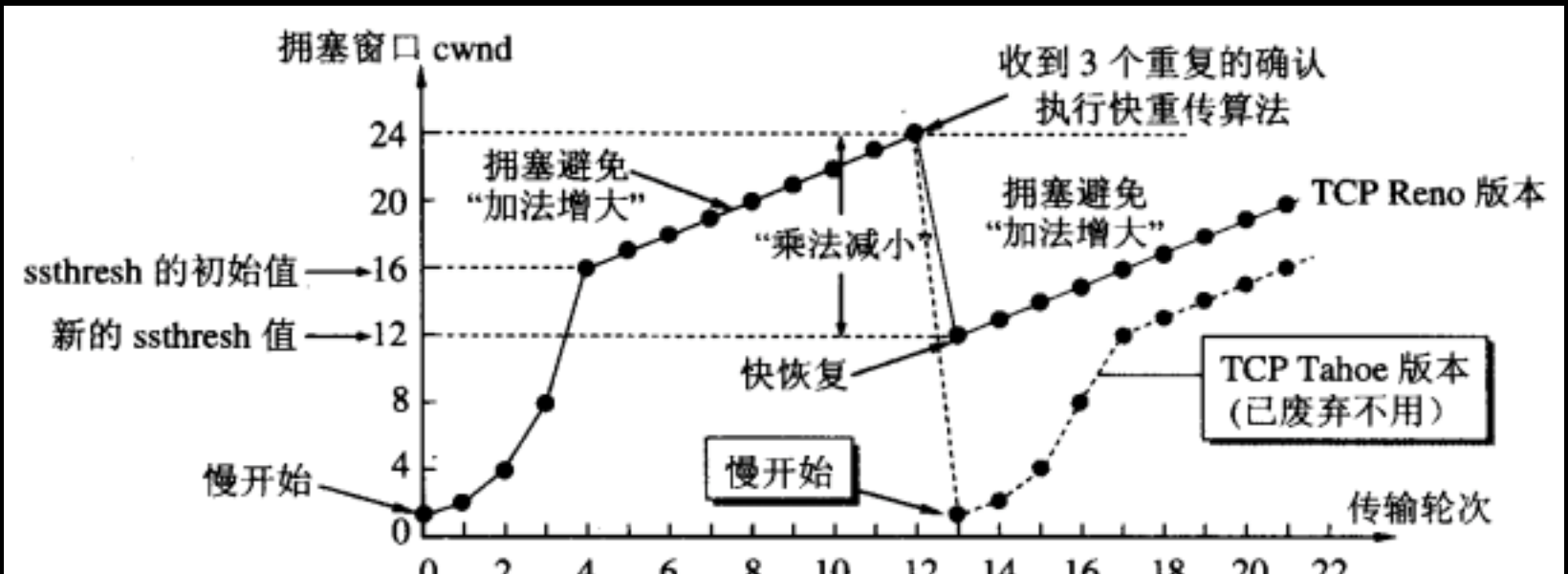
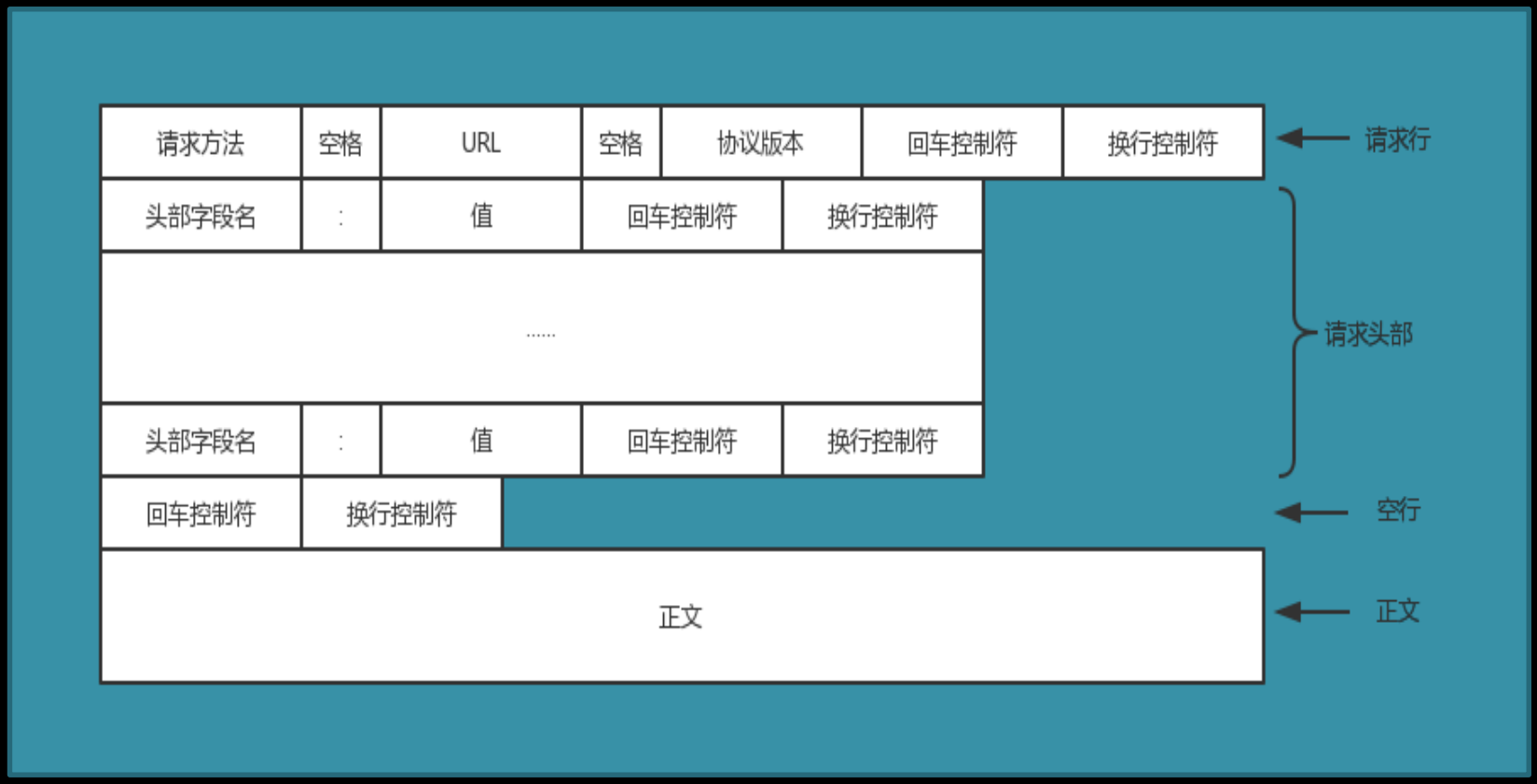
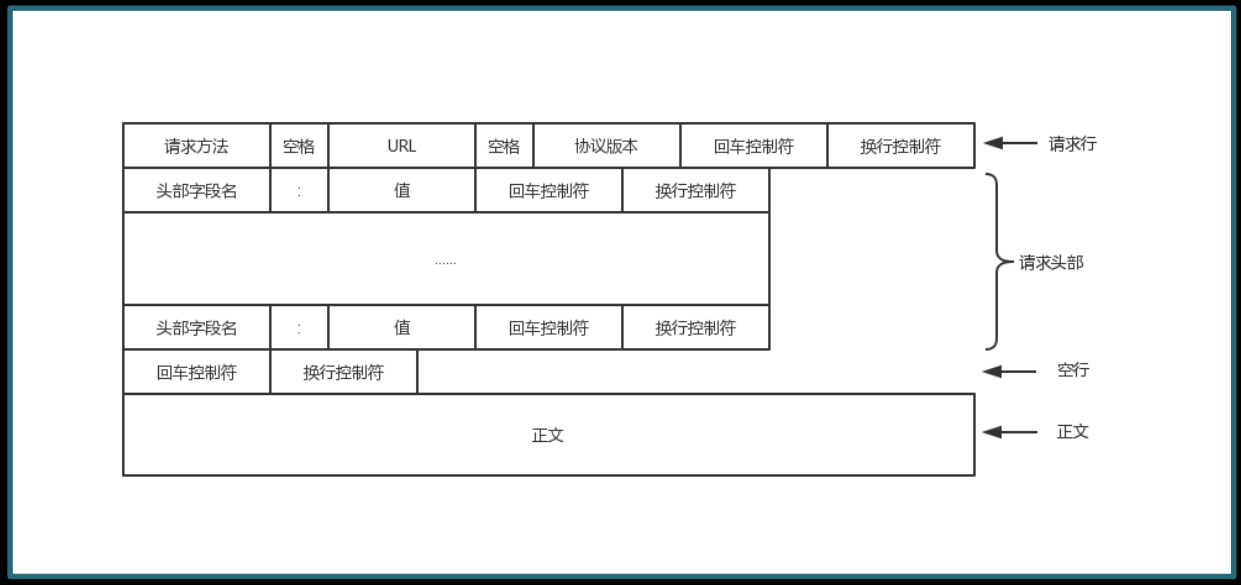


图 5-27　从连续收到三个重复的确认转入拥塞避免

HTTP协议


请求方法
空格
URL
空格
协议版本
回车控制符
换行控制符
请求行
头部字段名
:
值
回车控制符
换行控制符
.....
头部字段名
:
值
回车控制符
换行控制符
请求头部
回车控制符
换行控制符
空行
正文
正文

HTTP请求


请求方法
空格
URL
空格
协议版本
回车控制符
换行控制符
请求行
头部字段名
:
值
回车控制符
换行控制符
.....
头部字段名
:
值
回车控制符
换行控制符
请求头部
回车控制符
换行控制符
空行
正文
正文

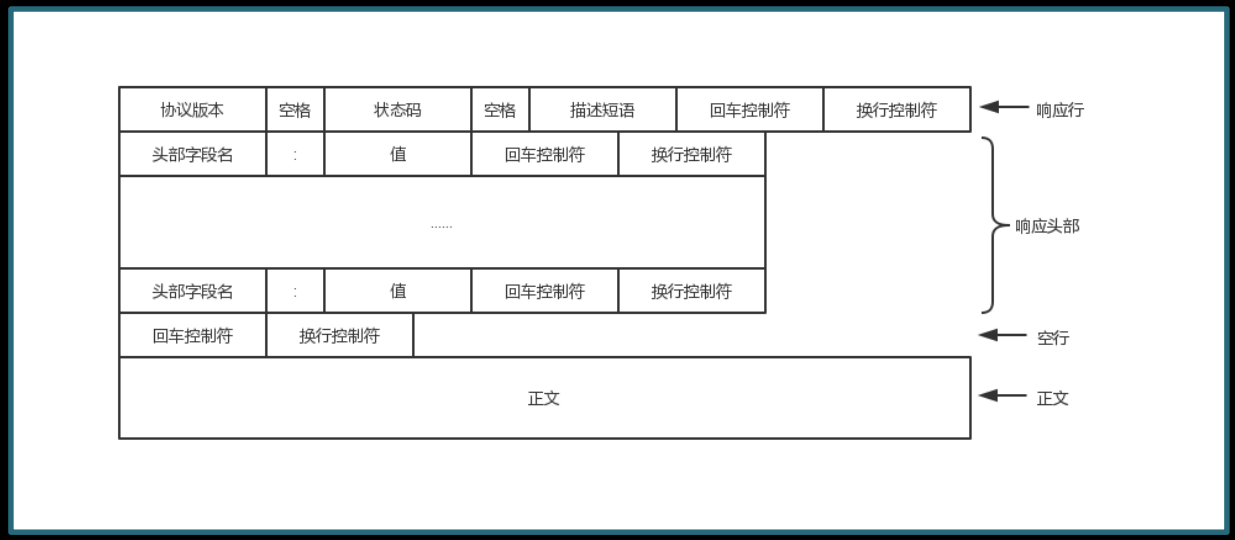
HTTP请求

请求方法	含义
GET	请求指定的资源地址，一般来说GET方法只用于数据的读取，例如在地址栏输入一个地址或者点击一个链接地址。
POST	向指定资源提交数据，如：表单数据提交、文件上传等，请求数据被包含在正文中。
HEAD	HEAD方法与GET方法一样，都是向服务器发出指定资源的请求。但是服务器在响应HEAD请求时只传头部信息，可根据头部的Content-Length判断网页是否更新，或者根据头部的状态码判断身份是否过期等。
PUT	向指定资源位置上传其最新内容，与POST不同的是PUT是幂等的，而幂等只是语义上的差别。关于幂等请进一步查阅资料。
DELETE	请求服务器删除资源。
TRACE	请求服务器回显其收到的请求信息，该方法可用于HTTP请求的测试或诊断。由于此方法回显请求信息，黑客可利用此方法获取Cookie实施跨站漏洞攻击（XST），建议服务器关闭TRACE方法。
CONNECT	HTTP代理服务器连接时使用，与web开发无关
OPTIONS	用于探测针对某个资源应有的约束，例如支持的方法以及自定义的头部。该方法可用'*'来代替资源名称，测试服务器功能是否正常。JavaScript的XMLHttpRequest对象进行CORS跨域资源共享时，就是用OPTIONS方法发送嗅探请求，以判断是否有对指定资源的访问权限。

HTTP请求头

请求头字段	含义
Accept	指定客户端所能接受处理的页面类型，如text/html, application/json
Accept-Charset	指定客户端所能接受处理的字符集，默认支持所有字符集
Accept-Encoding	指定客户端所能接受处理的数据编码方式，如gzip, deflate
Accept-Language	指定客户端所能接受处理的语言类型, 如zh-cn，默认支持所有语言
Authorization	指定客户端身份认证信息
Cookie	存储在Cookie中的信息
Connections	请求采用持久连接的方式，如keep-alive
Date	请求发送的日期和时间
Host	指定请求服务器的域名和端口号
Referer	当前网页上一次请求的网页地址
User-Agent	客户端的信息，包含客户端操作系统、浏览器内核等其他属性
Upgrade	客户端希望切换到其他协议
Cache-Control	缓存指令

HTTP响应


协议版本
空格
状态码
空格
描述短语
回车控制符
换行控制符
响应行
头部字段名
:
值
回车控制符
换行控制符
.....
头部字段名
:
值
回车控制符
换行控制符
响应头部
回车控制符
换行控制符
空行
正文
正文

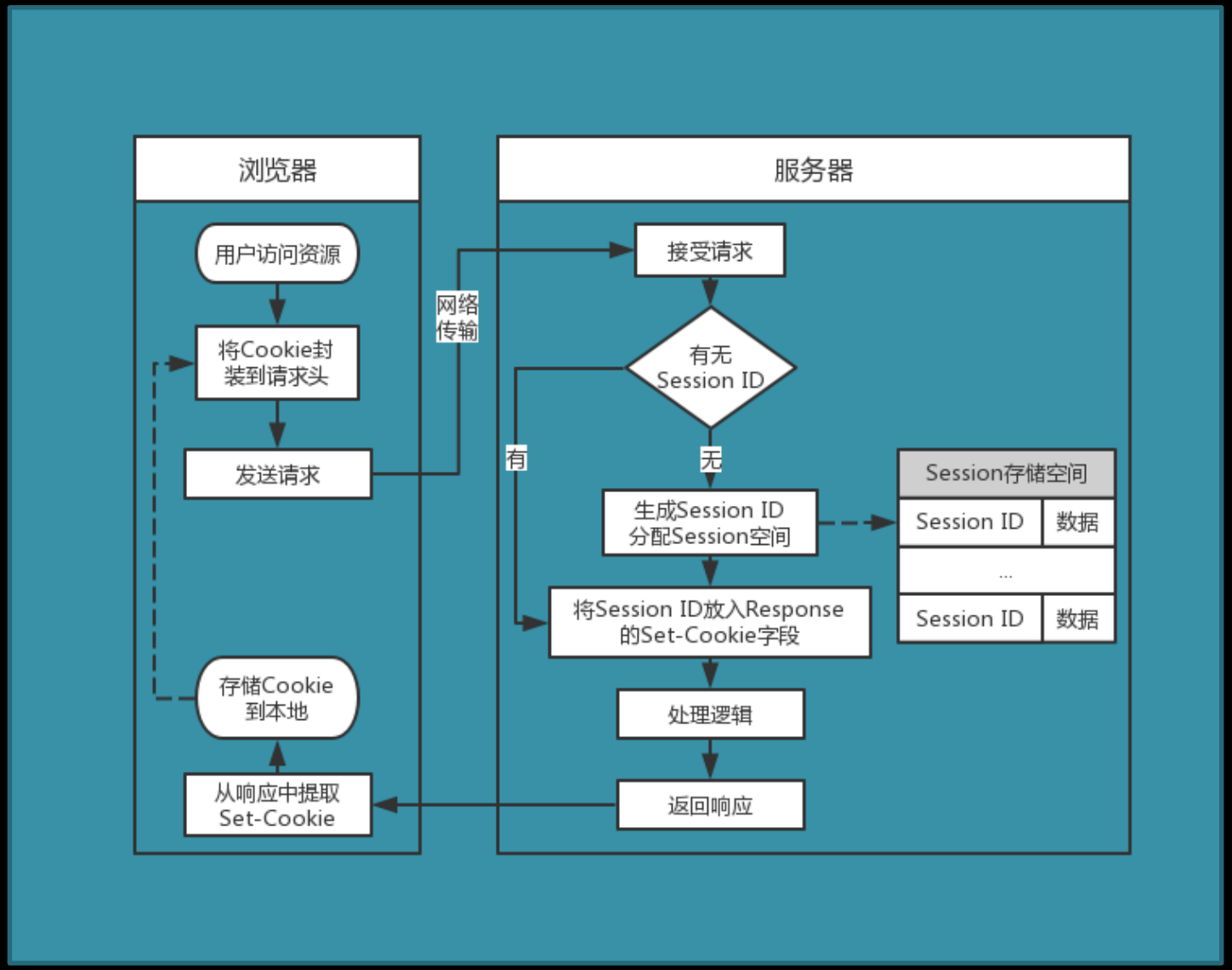
HTTP响应行

状态码类型	含义	示例
1xx	指示类响应，表示请求已被接收，继续处理	如100表示服务器同意处理客户的请求
2xx	成功类响应，表示请求已被成功处理	如200表示请求成功，204表示无内容，但是也请求成功
3xx	重定向类响应，表示请求要完成必须进行下一步的操作	如301表示页面已经被重定向了
4xx	客户端错误类响应，表示客户端请求有语法错误或者请求无法实现	如404表示客户端请求的资源不存在
5xx	服务器错误类响应，表示服务器未能实现所需要的请求处理	如500表示服务器发生了不可预知的错误

HTTP响应头

响应头	含义
Allow	服务器支持哪些请求方法
Server	服务器软件的一些信息
Content-Encoding	请求资源所使用的编码类型
Content-Language	请求资源所使用的语言
Content-Length	响应正文的长度，以字节为单位
Content-Type	请求资源的MIME类型，如text/html
Date	响应时的日期和时间
Last-Modified	请求资源最后被修改的日期和时间
Expires	网页缓存过期时间
Location	重定向
Accept-Range	服务器支持指定字节范围的请求
Refresh	客户端多少秒后刷新网页
Set-Cookie	服务器端设置的Cookie信息，每次写入Cookie都会生成一个Set-Cookie
P3P	用于跨域Cookie设置
Upgrade	显示服务器希望切换到的协议

Cookie 与 Session



HTTPS

小浏：你好，让我们开始一个加密通信吧。
我支持这些加密算法：ECDHE and RSA with 128 AES in GCM mode and SHA256，RSA with 128bit AES in GCM mode and SHA256
我生成了一个随机数：da56awesa

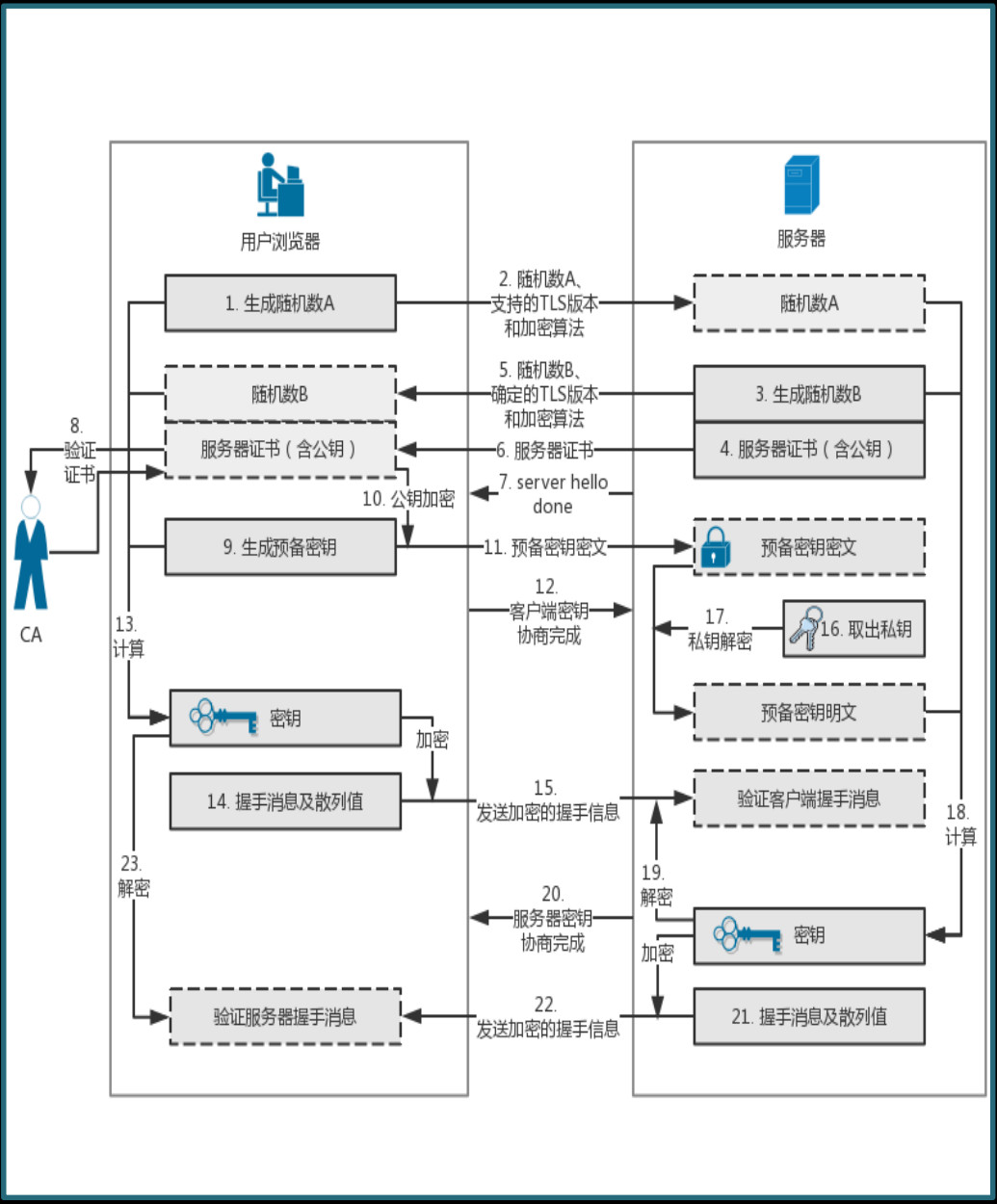
小服：好的，我们使用RSA with 128bit AES in GCM mode and SHA256加密算法。
这是我生成的随机数：556e8wesds4dfa
这是我的证书。

小浏：我到CA验证了下证书是合法的。
我生成了一个预备密钥，用你给我的证书里面的公钥对它进行了加密，发给你。
我们可以根据刚才交换的随机数和预备密钥，用相同的方法算到相同的密钥。

小服：我收到了你用公钥加密的预备密钥，我可以用私钥解密得到预备密钥明文。
我也用相同的方法算到了密钥，下面我可以用密钥加密数据了。

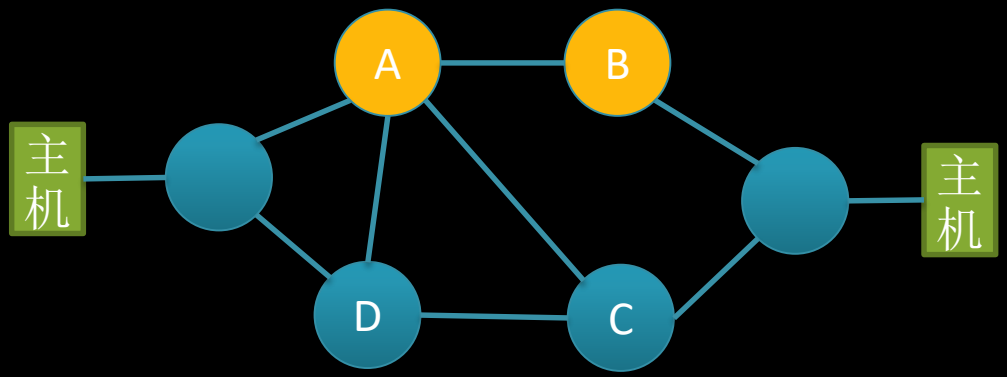
小浏：太好了，终于安全了。
我要查看我的订单。

小服：好的，这是你的订单。



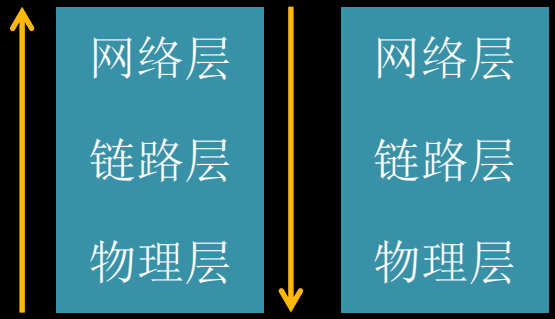
路由的原理

由路由组层的互联网



A 物理层得到数据
A 链路层解包，比较MAC地址
A 网络层解包，找出目标IP
A 检索路由表，寻找目标主机路径

A 修改目的MAC地址为B，链路层打包
A 通过物理层发出数据
B 物理层得到数据，循环直到目的主机



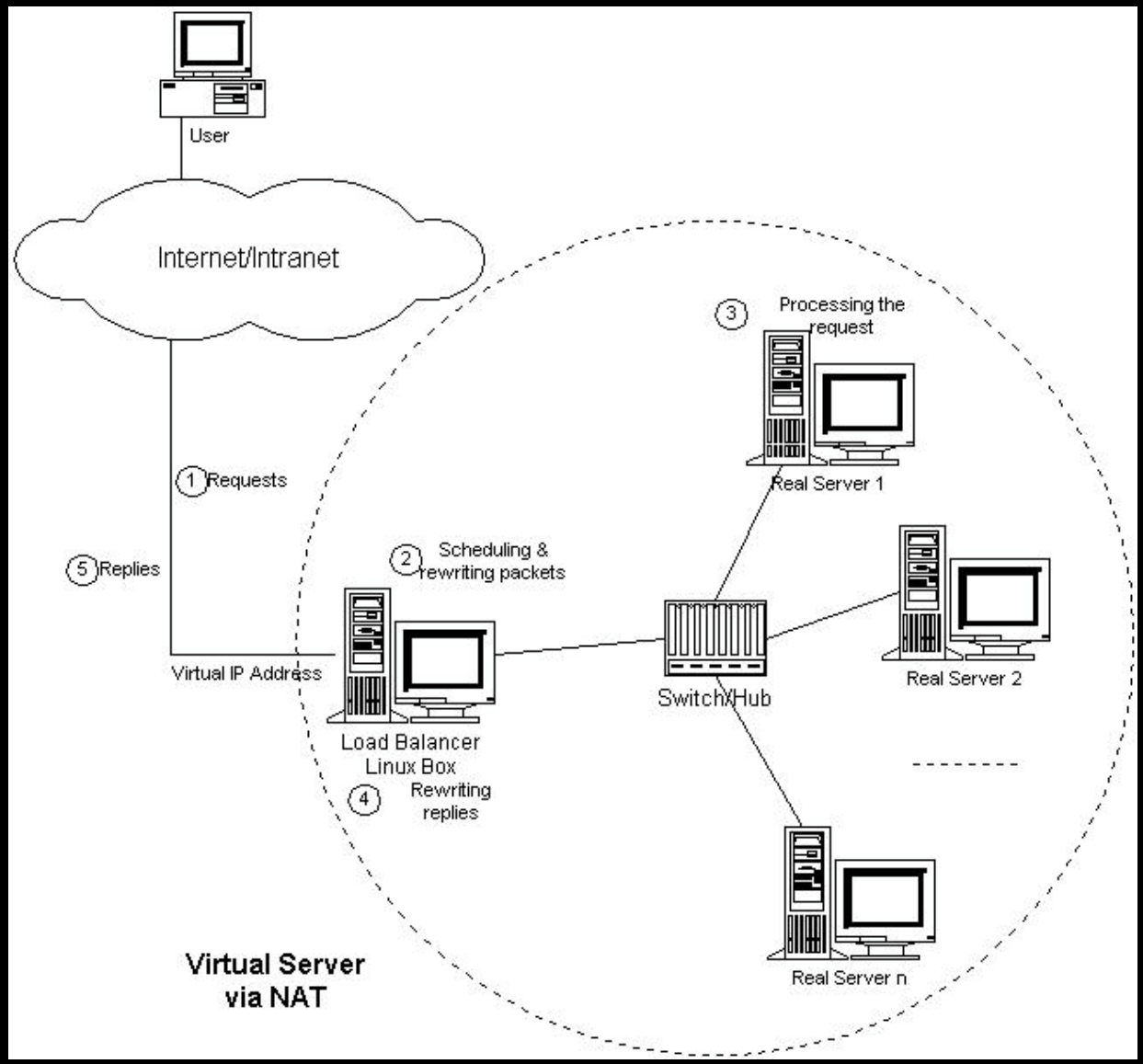
拆包、寻址、封装、转发

思考ISP运营商为什么可以在网页中插入广告？

负载均衡

- Taobao 的LVS四层负载均衡原理
- VS/NAT, VS/TUN, VS/DR

负载均衡VS/NAT



负载均衡器充当路由器的角色

修改IP数据报的IP地址和端口为应用服务器，转发到应用服务器

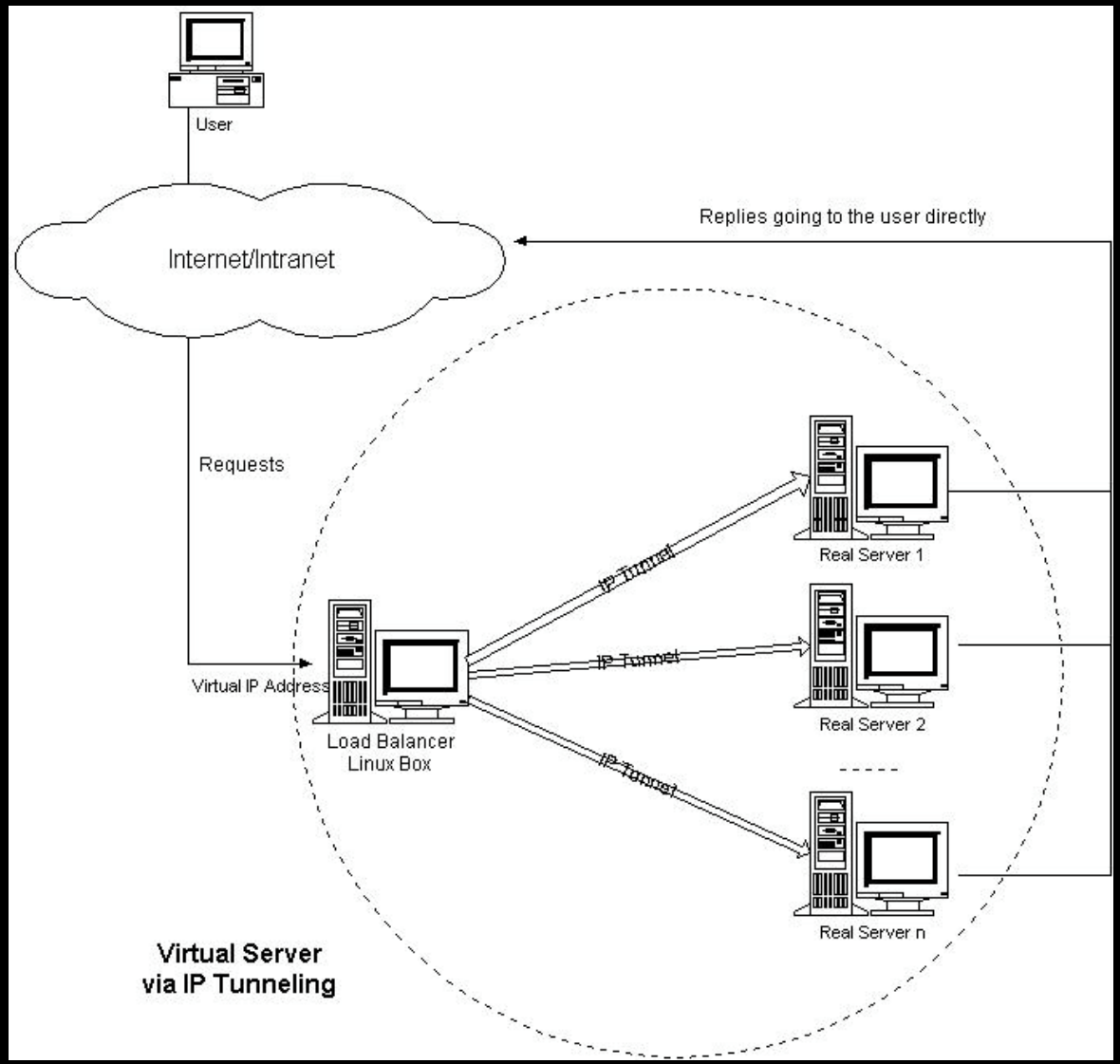
应用服务器处理完成后，返回给负载均衡器

负责均衡器修改源IP和端口为自己的IP和端口

缺点：负载均衡器压力大

参 VS/NAT

负载均衡VS/TUN



负载均衡器与应用服务器之间建立一个私有网络隧道

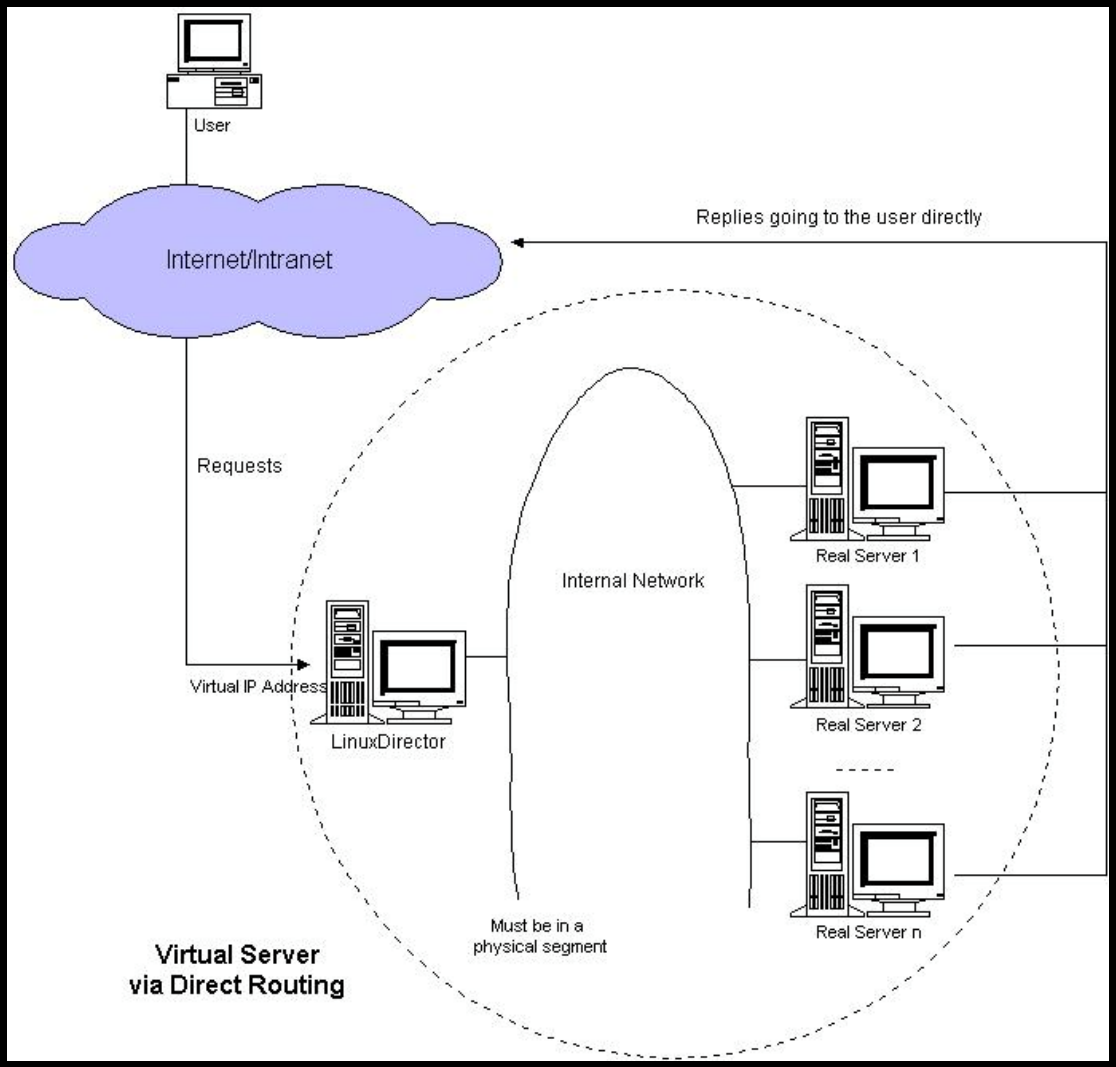
负载均衡器将IP数据报再次打包封装上目标服务器的IP，转发出去

目标服务器收到后处理，返回IP数据报中将源IP地址设置为负载均衡器的IP，直接返回客户端

优点：
远距离负载，跨网络
负载均衡器无压力

参 VS/TUN

负载均衡VS/DR



负载均衡器与应用服务器在一个局域网内

负载均衡器将IP数据报的MAC地址修改为应用服务器的MAC地址后，转发出去

应用服务器接受数据报后处理，将返回的数据报的源IP设置为负载均衡器的IP

优点：
负载均衡器无压力，效率高
缺点：
只能局域网内负载

参 VS/DR

负载均衡七层负载Nginx

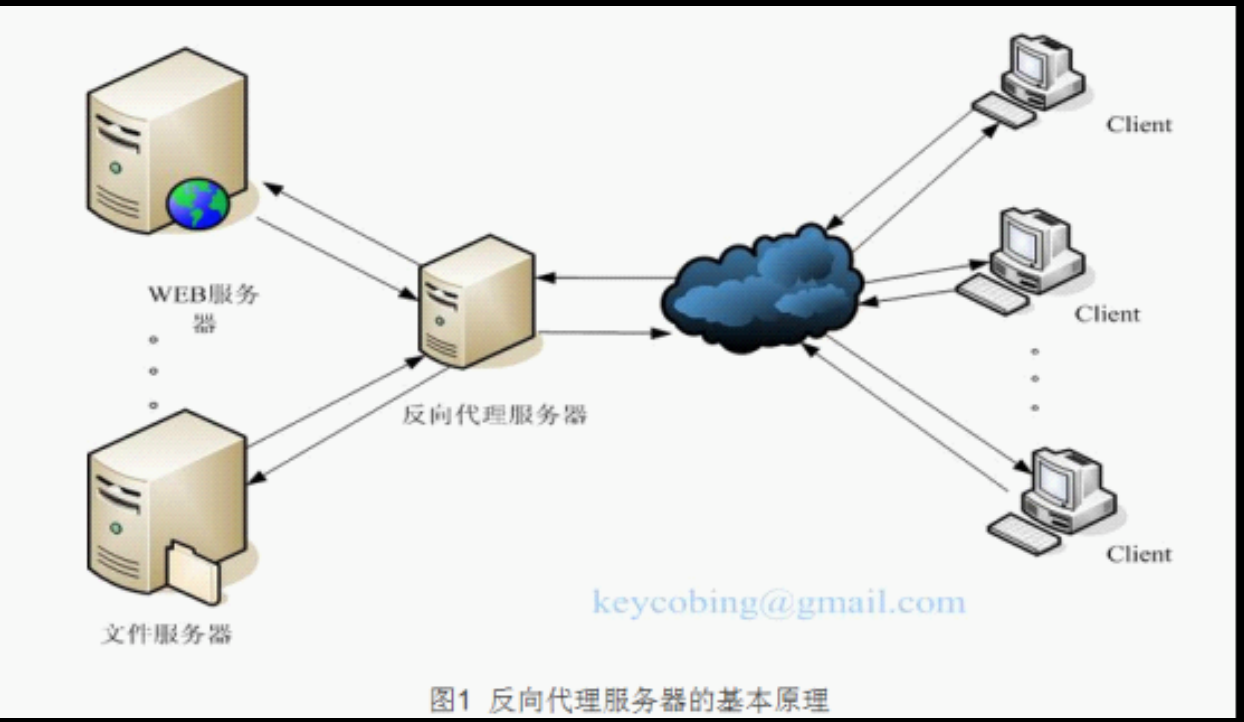


图1 反向代理服务器的基本原理

使用Nginx转发HTTP请求，也称前端代理服务器

Nginx解析请求的HTTP报文，根据URL和负载均衡策略确定应用服务器，向应用服务器发送请求

应用服务器执行并返回给代理服务器，代理服务器缓存，返回客户端

缺点：
Nginx服务器压力大
性能较低
优点：
实现简单，可以缓存

参 反向代理服务器

扩展知识：负载均衡算法

负载均衡DNS智能解析

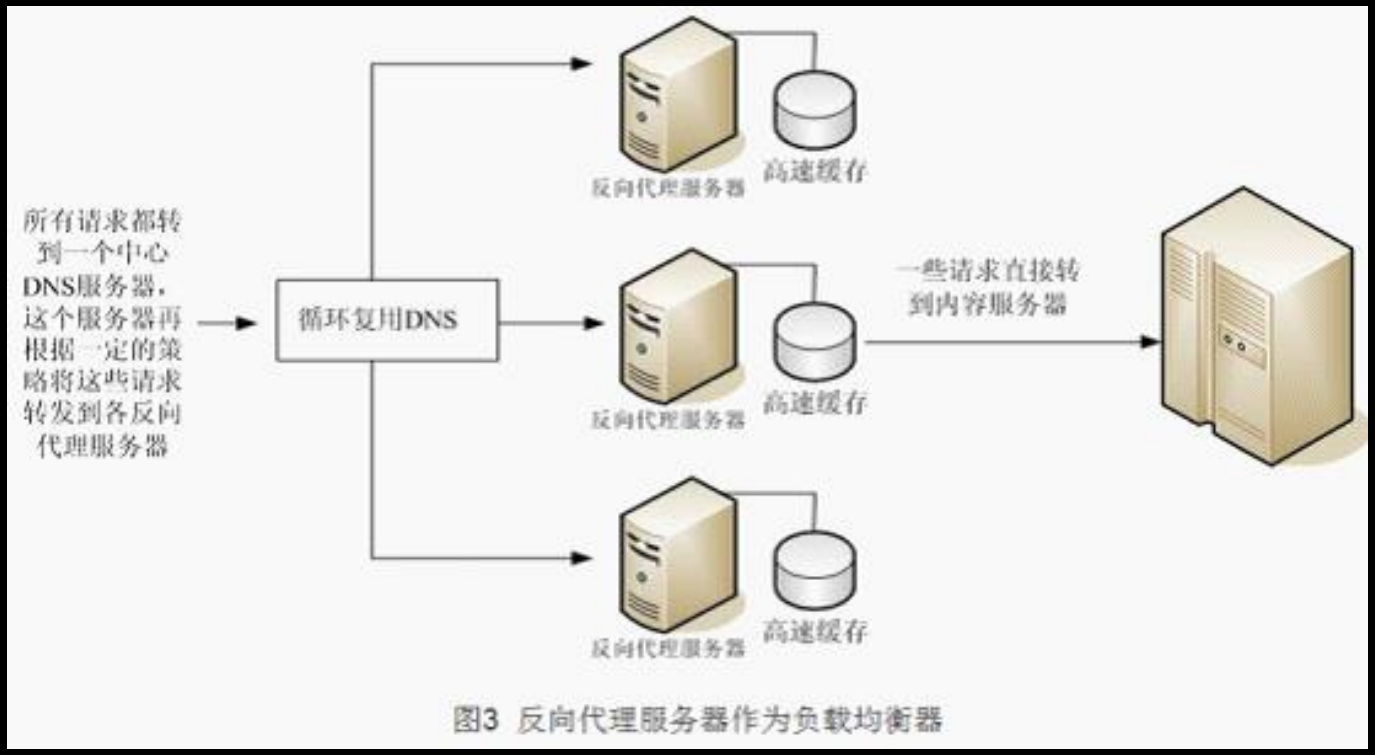
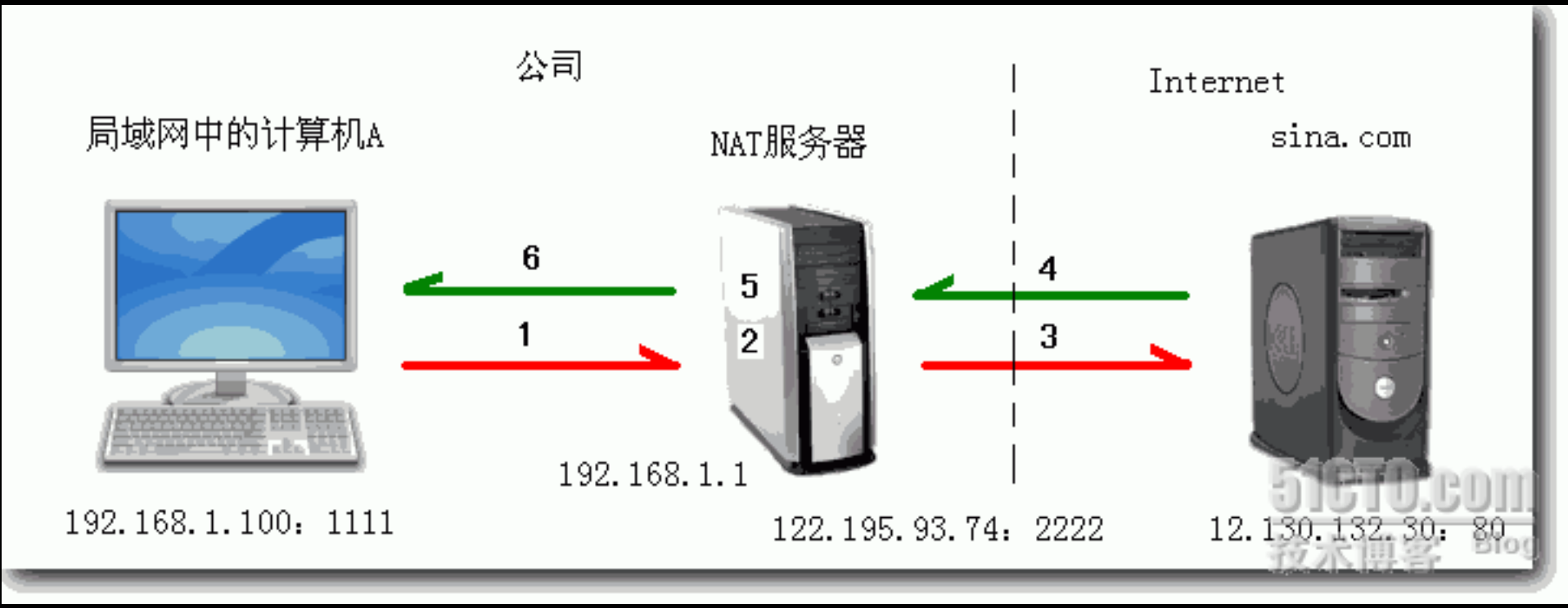


图3 反向代理服务器作为负载均衡器

用户请求域名URL时，首先经过DNS服务器解析，DNS根据用户的网段智能解析到后端服务器。

路由NAT



计算机A:1111端口发起请求到sina的ip:80端口，NAT服务器为计算机A动态分配一个端口2222，将A的数据报中IP地址改为NAT公网IP端口为2222后扔到互联网，sina接到数据报处理后，返回给NAT公网IP:2222，NAT将数据报IP和端口改为A 192.168.1.100:1111，转发给A。

计算机A发起请求时，NAT服务器动态分配了一个端口，并记录Session在NAT的缓存中，此时计算机A在局域网中向sina服务器打洞，只有sina服务器可以走这个洞。

参 QQ通信原理

P2P穿透

- 两个NAT内的主机如何进行点对点通信？

P2P打洞

- ClientA发起Server请求
- Server记下A的SessionAS
- ClientB发起Server请求
- Server告诉B： A的SessionAS
- B向A以SessionAS发起多个探测包
- B告诉Server已经向A发送了探测包
- Server告诉A向B传输数据和B的SessionBS
- A以B的SessionBS向B传输数据

思考

- 理学院329的IP是私有IP，能够访问外网，不能被外网直接访问。
- 如何利用这个IP搭建一个代理请求校内网成绩的服务器？

关键词词典

- 域名、DNS服务器、MAC地址、IP、端口
- 信号
- 路由
- 位流、帧、数据包、段、报文
- 负载均衡
- TCP、UDP、HTTP