Contents

[HTTP 1](#_Toc456462848)

[<图解HTTP> 1](#_Toc456462849)

[基本概念 1](#_Toc456462850)

[TCP/IP的分层架构模式： 2](#_Toc456462851)

[HTTP是无状态协议 2](#_Toc456462852)

[TCP协议的分割和三次握手 2](#_Toc456462853)

[持久链接 HTTP keep-alive 3](#_Toc456462854)

[Cookie的状态管理 3](#_Toc456462855)

[编码提升传输速率 4](#_Toc456462856)

[发送多种数据的多部分对象集合(multipart) 4](#_Toc456462857)

[恢复下载中断处的下载？ 4](#_Toc456462858)

[内容协商返回最合适的内容 4](#_Toc456462859)

[客户端请求页面流程 5](#_Toc456462860)

[URI格式： 5](#_Toc456462861)

[HTTP报文：多行数据构成的字符串文本 5](#_Toc456462862)

[HTTP状态码 6](#_Toc456462863)

[与HTTP协作的web服务器 6](#_Toc456462864)

[HTTP报文首部 7](#_Toc456462865)

[HTTPS 10](#_Toc456462866)

[确认访问用户身份的认证 11](#_Toc456462867)

[基于http的功能追加协议 12](#_Toc456462868)

[构建Web内容的技术 13](#_Toc456462869)

# HTTP

## <图解HTTP>

### 基本概念

协议：计算机与网络设备要相互通信，双方就必须基于相同的方法。如探测到通信目标、由哪一边先发起通信、使用哪种语言进行通信、怎样结束通信等规则都需要事先确定

DNS协议：提供通过域名查找IP地址，或逆向从IP地址反查域名的服务

IP地址：节点被分配到的地址

MAC地址：网卡所属的固定地址

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的地址。

路由选择：有点像快递公司的送货过程。

### TCP/IP的分层架构模式：

应用层(HTTP, FTP, …)

传输层(TCP/UDP): 提供处于网络连接中的两台计算机之间的数据传输

网络层(IP): 处理在网络上流动的数据包

链路层：处理连接网络的硬件部分

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，为了传输方便，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

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是无状态协议

HTTP协议对发送过的请求或响应都不做持久化处理， 使用HTTP协议，每当有新的请求发送时，就会有对应的新响应产生。协议本身燕不保留之前一切的请求或响应报文的信息。这是因为更快地处理大量事务，确保协议的保伸缩性，而特意把HTTP协议设计成如此简单

用户场景需要状态，如何处理？

比如用户登录到一家购物网站，即使他跳转到该站的其他页面后，也需要能继续保持登录状态

HTTP/1.1虽然是无状态协议，但为了实现期望的保持状态功能，于是引入了Cookie技术，有了Cookie再用HTTP协议通信。就可以管理状态

### TCP协议的分割和三次握手

如何更容易传送大数据？

将大块数据分割成以报文段为单位的数据包进行管理。

如何确保数据能到达目标？

TCP协议采用三次握手策略

发送端首先发送一个带SYN标志的数据包给对方。

接收端收到后，回传一个带有SYN/ACK标志的数据包以示传达确认信息

最后，发送端再回传一个带ACK标志的数据包，代表“握手”结束

（若在握手过程中某个阶段莫名中断，TCP协议会再次以相同的顺序发送相同的数据包）

### 持久链接 HTTP keep-alive

每进行一次HTTP通信就要断开一次TCP连接

建立TCP连接 -> HTTP请求/响应 -> 断开TCP连接

若html里含js, css, jpg等，在发送请求访问HTML页面资源的同时，也会请求该HTML页面里包含的其他资源。因此每次的请求都会造成无谓的TCP连接建立和断开，增加通信量的开销

解决办法：采用持久连接，HTTP keep-alive。只要任意一端没有明确提出断开连接，则保持TCP连接状态。在HTTP/1.1中，所有的连接默认都是持久连接

建立TCP连接 -> HTTP请求/响应 -> HTTP请求/响应 … -> 断开TCP连接

持久连接使得多数请求以管线化pipelining方式发送成为可能。也就是说从前发送请求后需要等待并收到响应，才能发送下一个请求。管线化是指能够做到同时并行发送多个请求，而不需要一个接一个地等待响应

建立TCP连接 -> HTTP请求1, HTTP请求2, … -> HTTP响应 … -> 断开TCP连接

### Cookie的状态管理

假设要求登录认证的web页面本身无法进行状态的管理(不记录已登录的状态)，那么每次跳转新页面不是要再次登录，就是要在请求报文中附加参数来管理登录状态

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，然后对比服务端的记录，最后得到之前的状态信息

第一次请求：

客户端：没有Cookie信息状态

服务端：保存请求，生成Cookie，记住是向谁发送的，在响应中添加Cookie后返回

HTTP/1.1 200 OK

Date: ..

Server: Apache

<Set-Cookie: sid=…>

第二次请求：

客户端：存有Cookie信息状态，请求中添加Cookie后发送

GET /img/ HTTP/1.1

Host: hackr.jp

Cookie: sid=…

服务端：检查Cookie,发回响应

### 编码提升传输速率

压缩传输的内容编码，常用的内容编码：gzip, compress, deflate, indentity

### 发送多种数据的多部分对象集合(multipart)

发送的一份报文主体内可含有多类型实体。通常是在图片或文本文件等上传时使用

在web表单文件上传时使用

Content-Type: multipart/form-data; boundary=Aa

--Aa

…

--Aa--

状态码206(Partial Content部分内容)响应报文包含了多少范围的内容时使用

HTTP/1.1 206 Partial Content

Content-Type: multipart/byteranges; boundary=THIS\_STRING\_SEP

-- THIS\_STRING\_SEP

…

-- THIS\_STRING\_SEP—

### 恢复下载中断处的下载？

获取部分内容的范围请求

Range: byes=5001-10000

### 内容协商返回最合适的内容

同一个Web网站有可能存在着多份相同内容的页面，比如英文版，中文版等。当浏览器的默认语言为英语或中文，访问相同URI的web页面时，则会显示对应的web页面。这样的机制称为内容协商(content negotiation)

内容协商机制：是指客户端和服务器端就响应的资源内容进行交涉，然后提供给客户端最为适合的资源。内容协商会以响应资源的语言、字符集、编码方式等作为判断的基准

Accept

Accept-Charset

Accept-Encoding

Accept-Language

Content-Language

### 客户端请求页面流程

客户端

HTTP协议：生成针对目标web服务器的HTTP请求报文

TCP协议：为了方便通信，将HTTP请求报文分割成报文段，按序号分为多个报文段，把每个报文段可靠地传给对方（通过三次握手）

IP协议：搜索对方的地址，一边中转，一边传送

服务端

TCP协议：从客户端接收到报文段，重组报文段，按序号以原来的顺序重组请求报文

HTTP协议：对Web服务器请求的内容的处理

请求的处理结果也同样利用TCP/IP通信协议向用户进行回传

### URI格式：

<http://user:pass@www.example.jp:80/dir/index.htm?uid=1#ch1>

协议方案名://登录信息（认证）@服务器地址：端口号/带层次的文件路径?查询字符串#片段标识符

### HTTP报文：多行数据构成的字符串文本

请求报文：请求方法，请求URI, 协议版本、可选的请求首部字段和内容实体

POST /form/entry HTTP/1.1

Host: hackr.jp

Connection: keep-alive

Content-type: application/x-www-form-urlencoded

Content-length: 16

name=qzlin&age=35

响应报文： 协议版本、状态码、用以解释状态码的原因短语、可选的响应首部字段以及实体构成

HTTP/1.1. 200 OK

Date: Tue, 10 Jul 2012

Content-Length: 362

Content-type: text/html

<html>

…

### HTTP状态码

* 1XX Informational信息性状态码 接收的请求正在处理
* 2XX Success成功状态码 请求正常处理完毕

200 OK

204 No Content 一般在只需要从客户端往服务器发送信息，而对客户端不需要发送新信息内容的情况下使用

206 Partial Content 响应报文中包含由Content-Range指定范围的实体内容

* 3XX Redirection重定向状态码 需要进行附加操作以完成请求

301 Moved Permanently 永久性重定向

302 Found 临时重定向

和301 Moved Permanently区别：302状态码代表的资源不是被永久移动，只是临时性质的。换句话说，已移动的资源对应的URI将来还有可能发生改变

303 See Other

当使用POST方法访问CGI程序，其执行后的处理结果是希望客户端能以GET方法重定向到另一个URI上去时，返回303状态码。

304 Not Modified

该状态码表示客户端发送附带条件的请求时，服务端允许请求访问资源，但未满足条件的情况。304状态码返回时，不包含任何响应的主体部分

307 Temporary Redirect

* 4XX Client Error客户端错误状态码 服务器无法处理请求

400 Bad Request

401 Unauthorized

当浏览器初次接收到401响应，会弹出认证用的对话窗口。表示需要通过HTTP认证

403 Forbidden 未获得文件系统的访问授权

404 Not Found

* 5XX Server Error服务器错误状态码 服务器处理请求出错

500 Internal Server Error

该状态码表明服务器端在执行请求时发生了错误。也有可能是web应用存在的bug或某些临时的故障

503 Service Unavailable

该状态码表明服务器暂时处于超负载或正在进行停机维护，现在无法处理请求

重点：状态码和状况可能不一致，因为不少返回的状态码响应都是错误的，但是用户可能察觉不到这点。比如Web应用程序内部发生错误，状态码依然返回200 OK

### 与HTTP协作的web服务器

HTTP/1.1规范允许一台http服务器搭建多个web站点，是利用了虚拟主机的功能

在相同的ip地址下，由于虚拟主机可以寄存多个不同主机名和域名的web网站，因此在发送http请求时，必须在host首部内完事指定主机名或域名的uri

在客户端和服务器之间，可能存在通信数据转发程序：代理、网关、隧道

代理：接收由客户端发送的请求并转发给服务器，同时也接收服务器返回的响应并转发给客户端。每次通过代理服务器转发请求或响应时，会追加写入via首部信息

使用代理服务器的理由：利用缓存技术减少网络带宽的流量，组织内部针对特定网站的访问控制，以获取访问日志为主要目的，等等

缓存是指代理服务器或客户端本地磁盘内保存的资源副本

即使存在缓存，也会因为客户端的要求、缓存的有效期等因素，向源服务器确认资源的有效性

网关：接收从客户端发送来的请求时，它就像自己拥有资源的源服务器一样对请求进行处理。利用网关能提高通信的安全性，因为可以在客户端与网关之间的通信线路上加密以确保连接的安全

隧道：客户端和服务器之间的中转，并保持双方通信连接的应用程序。一般使用SSL等加密手段进行通信

### HTTP报文首部

首部内容为客户端和服务器分别处理请求和响应提供所需要的信息

使用首部字段是为了给浏览器和服务器提供报文主要大小、所使用的语文、认证信息等内容

HTTP请求报文

= 方法、URI、HTTP版本、HTTP首部字段（请求首部，通用首部+实体首部）

HTTP响应报文

= HTTP版本、状态码、HTTP首部字段（响应首部，通用首部+实体首部）

#### 通用首部

Cache-Control: private, max-age=0, no-cache

控制缓存的行为

Connection: Keep-Alive

http/1.1版本的默认连接都是持久连接，客户端会在持久连接上连续发送请求。当服务器端想明确断开连接时，则指定Connection首部字段的值为Close

Date：Tue, …

创建报文的时间

Transfer-Encoding: chunked

传输报文主体时采用的编码方式

Via: 1.0 gw.hackr.jp

为了追踪传输路径

#### 请求首部字段

Accept: type/subtype

通知服务器用户代理能够处理的媒体类型及媒体类型的相对优先级

文本文件：text/html, text/plain, text/css, … application/xhtml+xml, …

图片文件：image/jpeg, image/gif, …

视频文件：video/mpeg, video/quicktime, …

应用程序使用的二进制文件：application/octet-stream, application/zip…

Accept-Charset: Unicode

通知服务器用户代理支持的字符集及字符集的相对优先顺序

Accept-Encoding: gzip, deflate

通知服务器用户代理支持的内容编码及内容编码的优先级顺序

Accept-Language: zh-cn, zh;q=0.7, en-us, en; q=0.3

通知服务器用户代理能够处理的自然语言集及优先级顺序

Authorization: Basic ddeeedd

通知服务器用户代理的认证信息（证书值）

Expect: 100-continue

From: [info@hackr.jp](mailto:info@hackr.jp)

Host: [www.hackr.jp](http://www.hackr.jp)

因为虚拟主机运行在同一个ip上，因此使用首部字段Host加以区分

请求被发送至服务器时，请求中的主机名会用IP地址直接替换解决。但如果这时，相同的IP地址下部署运行着多个域名，那么服务器就会无法理解究竟是哪个域名对应的请求。因此，就需要使用首部字段Host来明确指出请求的主机名

条件请求

If-Match 若If-Match的值与服务器资源的ETag匹配，则执行请求

If-None-Match

If-Modified-Since

If-Range 若If-Range的值与ETag匹配，则配合Range执行范围请求处理

If-Unmodified-Since

Max-Forwards: 10

使用HTTP协议通信时，请求可能会经过代理等多台服务器，服务器在往下一个服务器转发请求之前，会将Max-Forwards的值减1后重新赋值。当服务器接收到Max-Forwards值为0的请求时，则不再进行转发，而是直接返回响应

Proxy-Authorization: Basic …

认证行为发生在客户端 与代理之间

Range: bytes=5001-10000

会在处理请求之后返回状态码为206 Partial Content的响应。无法处理该范围请求时，则会返回状态码200 OK的响应及全部资源

User-Agent: Mozilla/5.0 …

将创建请求的浏览器和用户代理名称等信息传达给服务器

#### 响应首部字段

Accept-Ranges: bytes

告知客户端服务器是否能处理范围请求，以指定获取服务器端某个部分的资源

Age: 600

告知客户端，源服务器在多久前创建了响应，单位为秒

ETag: “12222…”

告知客户端实体标识。它是一种可将资源以字符串形式做唯一性标识的方式。服务器会为每份资源分配对应的ETag值

Location: <http://www>...

几乎所有的浏览器在接收到包含首部字段Location的响应后，都会强制性地尝试对已提示的重定向资源的访问

Proxy-Authenticate: Basic realm=”…”

将代理服务器所要求的认证信息发送给客户端

WWW-Authenticate: Basic realm=”…”

告知客户端访问请求URI指定资源需要的认证方案Basic, Digest, …

状态码401 Unauthorized响应中，肯定带有首部字段WWW-Authenticate

Server: Apache/2.2 (unix) PHP/5.

#### 实体首部字段

Allow: GET, HEAD

通知客户端，服务器支持的HTTP方法

Content-Encoding: gzip

实体主体部分选用的内容编码方式

Content-Language: zh-CN

Content-Length: 15000 实体主体的大小，单位是字节

Content-Location: <http://www>...

Content-MD5: dfasfdd…

客户端会对接收的报文主体执行相同的MD5算法获得的128位二进制数，然后经Base64编码，最后与首部字段Content-MD5的字段值比较

Content-Range: bytes 5001-

Content-Type: type/subtype

Expires: Wed, …

告诉客户端该资源失效的日期，若同时存在Cache-Controll的max-age指令时，会优先处理max-age指令

Last-Modified: Wed, …

服务端发送的Set-Cookie

Set-Cookie: status=enable; expires=Tue, …; path=/; domain=.hackr.jp; secure HttpOnly

当expires省略时，其有效期仅限于维持浏览器会话session时间段内，这通常限于浏览器应用程序被关闭之前

Secure属性用地限制web页面伏在https安全连接时才可以发送Cookie

Httponly属性使用cookie不能被javascript脚本访问

客户端带的Cookie

Cookie:status=enable

当客户端想获得http状态管理支持时，就会在请求中包含从服务器接收到的cookie

### HTTPS

HTTP缺点：

通信使用明文，内容可能会被窃听

不验证通信方的身份，因此有可能遭遇伪装

无法证明报文的完整性，所以有可能已遭篡改

HTTP报文使用明文方式发送，而TCP/IP协议族的工作机制，通信内容在所有的通信线路上都有可能遭到窥视

窃听相同段上的通信并非难事，只需要收集在互联网上流动的数据包就行了。对于收集来的数据包的解析工作，可交给那些抓包或嗅探器工具

两种加密方式

内容的加密

由于http协议中没有加密机制，那么就对http协议传输的内容本身加密，即把http报文里所含的内容进行加密处理

为了做到有效的内容加密，前提是要求客户端和服务器同时具备加密和解密机制

通信的加密

http, ftp, telnet -> SSL/TLS -> tcp/ip

https = http + 加密 + 认证 + 完整性保护

经常会在web的登录页面和购物结算界面等使用https通信

加密算法是公开的，密钥保密

共享密钥加密（对称密钥加密）

加密和解密同用一个密钥

在互联网转发密钥，如果通信被监听，那么密钥不可能会落入攻击者之手

公开密钥加密（非对称密钥加密）

公开密钥发布，所有人都可以知道；私有密钥只有解密者知道

关键：使用公开密钥加密，使用私有密钥解密

发送密文的一方使用对方的公开密钥进行加密处理，对方收到被加密的信息后，再使用自己的私有密钥进行解密。

利用这种方式，不需要发送用解密的私有密钥。

要想根据密文和公开密钥，恢复信息原文，目前技术上不太现实，（运算量太大）

HTTPS采用混合加密机制，为什么？

公开密钥加密与共享密钥加密相比，其处理速度更慢

所以组合，在交换密钥环节使用公开密钥加密方式，之后的建立通信交换报文阶段则使用共享密钥加密方式

如何证明公开密钥？

服务器向CA（certificate authority数据证书认证机构）提出公开密钥的申请，CA会对申请的公开密钥做数字签名，然后将公开密钥放入公钥证书后绑定在一起

服务器将公钥证书发送给客户端，公钥证书也叫做数字证书或证书

客户端使用CA的公开密钥，对公钥证书上的数字签名进行验证 （这里注意到：CA的公开密钥如何安全地给客户端？多数浏览器开发商发布版本时，会事先在内部植入常用认证机关的公开密钥）

确认客户端的客户端证书

https还可以使用客户端证书。以客户端证书进行客户端认证，证明服务器正在通信的对方始终是预料之内的客户端，其作用跟服务器证书如出一辙

例如：银行的网上银行就采用了客户端证书。在登录网银时不仅要求用户确认输入ID和密码，还会要求用户的客户端证书，以确认用户是否从特定的终端访问网银

由自认证机构颁发的证书称为自签名证书

如果使用OpenSSL这套开源程序，每个人都可以构建一套属于自己的认证机构，从而自己给自己颂发服务器证书

独立构建的认证机构叫自认证机构，由自认证机构颁发的“无用”证书也被戏称为自签名证书

### 确认访问用户身份的认证

核对信息：

密码：只有本人才会知道的字符串信息

动态令牌：仅限本人持有的设备内显示的一次性密码

数字证书：仅限本人（终端）持有的信息

生物认证：指纹和虹膜等本人的生理信息

IC卡等：仅限本人持有的信息

HTTP认证方式：

Basic 基本认证

Digest 摘要认证

SSL 客户端认证

FormBase 基于表单认证

发送请求

GET /private/ HTTP/1.1

Host:…

返回状态码401，需要认证

HTTP/1.1 401 Authorization Required

WWW-Authenticate: Basic realm=”input your ID and password”

浏览器弹出框，要求输入用户ID and password，然后浏览器会组合ID:password字符串，经过Base64编码，把这串字符串写入首部字段Authorization后，发送请求

GET /private/ HTTP/1.1

Host:…

Authorization: Basic dafdfsa….

认证成功返回状态码200,若认证失败返回状态码401

SSL客户端认证采用双因素认证

第一个认证因素是SSL客户端证书用来认证客户端计算机

第二个认证因素是密码用来确定这是用户本人的行为

基本表单认证

通过服务器端的web应用，将客户端发送过来的用户ID和密码与之前登录过的信息做匹配来进行认证

但HTTP是无状态协议，之前已认证成功的用户状态无法通过协议层面保存下来，即无法实现状态管理，因此即使当该用户下一次继续访问，也无法区分他与其他的用户。所以采用Cookie来管理Session,以弥补http协议中不存在的状态管理功能

步骤：

1. 客户端把用户ID和密码等登录信息放入报文的实体部分，通常是以POST方法把请求发送给服务器。一般使用https通信来进行html表单画面的显示和用户输入数据的发送
2. 服务器验证客户端发送过来的登录信息进行身份认证，然后把用户的认证状态与Session ID绑定后记录在服务器端.向客户端返回响应时，会在首部字段Set-Cookie内写入Session ID。

（可以把Session ID想象成区分用户的等位号，如果Session ID被第三方盗走，对方就可以伪装成你的身份进行恶意操作，所以一般服务器端也需要进行有效期的管理，保证其安全性）

1. 客户端接收到服务器端发来的Session ID后，会将其作为Cookie保存在本地。下次向服务器发送请求时，浏览器会自动发送Cookie,所以Session ID也随之发送到服务器。服务器端可通过验证接收到的Session ID识别用户和其认证状态

### 基于http的功能追加协议

为了尽可能地实时地显示这些更新的内容，服务器上一有内容更新，就需要直接把那些内容反馈到客户端的界面上。

使用http协议探知服务器上是否有内容更新，就必须频繁地从客户端到服务器端进行确认。如果服务器上没有内容更新，那么就会产生徒劳的通信

WebSocket

使用浏览器进行全双工通信

建立在http基础上的协议，发起方仍是客户端，一旦确立websocket通信连接，不论服务器还是客户端，任何一方都可以直接向对方发送报文

主要特点：

推送功能：服务器可直接发送数据，而不必等待客户端的请求

减少通信量：只要建立起websocket连接，就一直保持连接状态

为了实现websocket通信，在http连接建立之后，需要完成一次握手的步骤：

握手请求：

GET /chat HTTP/1.1

Upgrade: websocket

Connection: Upgrade

Sec-WebSocket-Key: …

Sec-WebSocket-Protocol: chat, superchat

握手响应：

HTTP/1.1 101 Switching Protocols

Upgrade: websocket

Connection: Upgrade

Sec-WeSocket-Accept: …

Sec-WebSocket-Protocol: chat

成功握手确立WebSocket连接之后，通信时不再使用http的数据帧，而采用WebSocket独立的数据帧

### 构建Web内容的技术

CGI的应用场景

请求 -> Web服务器 -> CGI程序 ->响应

CGI: Common Gateway Interface通用网关接口，指Web服务器在接收到客户端发送过来的请求后转发给程序的一组机制

每次接到请求，程序都要跟着启动一次，因此一旦访问量过大，Web服务器要承担相当大的负载

Servlet的应用场景

多个请求 -> Web服务器，Web容器Servlet, Servlet -> 响应

Servlet与Web服务器相同的进程中，负载小

TCP/IP

<图解TCP/IP>

分组交换协议

分组交换是指将大数据分割为一个个叫做包的较小单位传输的方法

计算机通信也会在每一个分组中附加上源主机地址和目标主机地址送给通信线路。这些发送端地址、接收端地址以及分组序号写入的部分称为‘报文首部’

一个较大的数据被分为多个分组时，为了标明是原始数据中的哪一部分，就有必要将分组的序号写入包中。接收端会根据这个序号，再将每个分组按照序号重新装配为原始数据

通信协议中，通常会规定报文首部应该写入了哪些信息、应该如何处理这些信息。相互通信的每一台计算机则根据协议构造报文首部、读取首部内容等。

电路交换与分组交换的区别？

电路交换中，两台计算机是可以独占线路进行数据传输的。其他计算机只能等待这台计算机处理结束以后才有机会使用这条电路收发数据

分组交换可以使多个用户同一时间共享一条通信线路进行通信，从而提高了线路的使用效率，也降低了搭建线路的成本

分组交换，让连接到通信电路的计算机将所要发送的数据分成多个数据包，按照一定的顺序排列之后分别发送。

有了分组交换，数据被细分后，所有的计算机就可以一齐收发数据，这样也就提高了通信线路的利用率。由于在分组的过程中，已经在每个分组的首部写入了发送端和接收端的地址，所以即使同一线路同时为多个用户提供服务，也可以明确区分每个分组数据发往的目的地，以及它是与哪台计算机进行的通信

在分组交换中，由分组交换机（路由器）连接通信线路。发送端计算机将数据分组发送给路由器，路由器收到这些分组数据以后，缓存到自己的缓冲区，然后再转发给目标计算机

在分组交换中，通信线路的速度可能会有所不同。根据网络拥堵的情况，数据达到目标地址的时间有长有短。另外，路由器的缓存饱和或溢出时，甚至可能会发生分组数据丢失、无法发送到对端的情况

单播：早先的固定电话

广播通信：电视播放

多播：电视会议

任播：DNS根域名解析服务器

协议的分层

OSI参考模型，每个分层都接收由它下一层所提供的特定服务，并且负责为自己的上一层提供特定的服务。上下层之间进行交互时所遵循的约定叫做接口。同一层之间的交互所遵循的约定叫做协议。

分层优点：易于单独实现每个分层的协议，并界定种个分层的具体责任和义务

分层缺点：过分模块化、使处理变得更加沉重以及每个模块都不得不实现相似的处理逻辑等问题

|  |  |  |  |
| --- | --- | --- | --- |
| 分层名称 | 功能 | 概览 |  |
| 应用层 | 针对特定应用的协议 | smtp, telnet, ftp, ssh, http | 应用程序 |
| 表示层 | 各种数据格式 -> 网络标准数据格式 | 接收不同表现形式的信息，如文字流、图像、声音等 |
| 会话层 | 负责建立和断开通信连接 | 如何建立连接，何时断开连接以及保持多久的连接 |
| 传输层 | 负责可靠传输 | Tcp/udp | 操作系统 |
| 网络层 | 寻址与路由选择 | Ip |
| 数据链路层 | 互连设备之间传送和识别数据帧 | 数据帧 -> 比特流 | 设备驱动程序与网络接口 |
| 物理层 |  | 比特流 -> 电子信号 |

每个分层上，在处理由上一层传过来的数据时可以附加上当前分层的协议所必须的首部信息。然后接收端对收到的数据进行数据“首部”与“内容”的分离，再转发给上一分层，并最终将发送端的数据恢复为原状

主机A 应用层->表示层->会话层->传输层->网络层->数据链路层->物理层

路由器 网络层<-数据链路层<-物理层

网络层->数据链路层->物理层

主机B 应用层<-表示层<-会话层<-传输层<-网络层<-数据链路层<-物理层

由于数据被转换为通用标准的格式后进行处理，使得异构的机型之间也能保持数据的一致性。这也正是表示层的作用所在。即表示层是进行“统一的网络数据格式”与“某一台计算机或某一款软件特有的数据格式”之间相互转换的分层

TCP/IP分层中的网络层与传输层的功能通常由操作系统提供

连接互联网的所有主机跟路由器必须都实现IP的功能

IP不具有重发机制。即使分组数据包未能到达对端主机也不会重发。因此，属于非可靠性传输协议

ICMP: IP数据包在发送途中一旦发生异常导致无法到达对端目标地址时，需要给发送端发送一个发生异常的通知

ARP: 从分组数据包的IP地址中解析出物理地址（MAC地址）的一种协议

计算机内部，通常同一时间运行着多个程序。为此，必须分清是哪些程序与哪些程序在进行通信。识别这些应用程序的是端口号

TCP： 面向有连接的传输层协议。它可以保证两端通信主机之间的通信可达。能够正确处理在传输过程中丢包、传输顺序乱掉等异常情况。为了建立与断开连接，有时它需要至少7次的发包收包，导致网络流量的浪费。此外。为了提高网络的利用率，TCP协议中定义了各种各样复杂的规范，因此不利用视频会议（音频、视频的数据量既定）等场合使用

UDP：面向无连接的传输层协议。UDP不会关注对端是否真的收到了传送过去的数据，如果需要检查对端是否收到分组数据包，或者对端是否连接到网络，则需要在应用程序中实现。UDP常用于分组数据较少或多播、广播通信以及视频通信等多媒体领域

SNMP: simple network management protocol网络管理协议， 使用SNMP管理的主机、网桥、路由器等称为SNMP代理，进行管理的那一端叫SNMP管理器。在SNMP代理端，保存着网络接口的信息、通信数据量、异常数据量以及设备温度等信息。

帧：表示数据链路层中包的单位。

数据报：是IP和UDP等网络层以上的分层中包的单位

段：表示TCP数据流中的信息

消息：应用协议中数据的单位

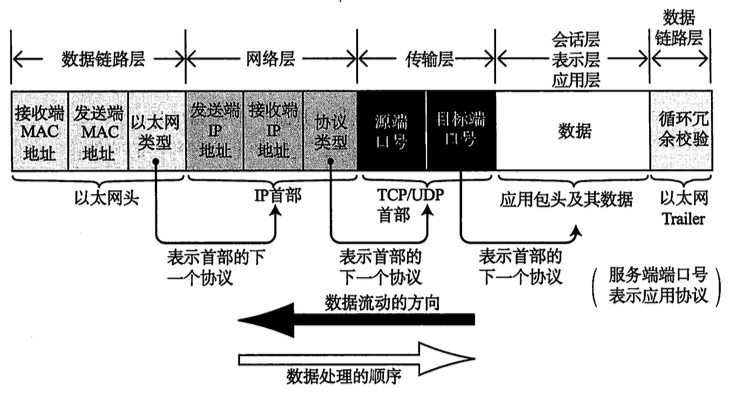
在数据包的首部，明确标明了协议应该如何读取数据。反过来说，看到首部，也就能够了解该协议必要的信息以及所要处理内容

TCP首部中包含源端口号，目标端口号（用以识别发送主机跟接收主机上的应用），序号（用以发送的包中哪部分是数据）以及校验和（用以判断数据是否被损坏）

IP首部中包含接收端IP地址以及发送端IP地址，以及TCP/UDP类型信息

以太网首部中包含接收端MAC地址、发送端MAC地址以及标志以太网类型的以太网数据的协议。

经过每个协议分层时，都必须有识别包发送端和接收端的信息。以太网会用MAC地址，IP会用IP地址，而TCP/UDP会用端口号作为识别两端主机的地址，这些地址信息都在每个包经由各个分层时，附加到协议对应的包首部里边



数据包接收处理流程

网络接口（以太网驱动）的处理

主机收到以太网包以后，首先从以太网的包首部找到MAC地址判断是否为发给自己的包，如果是，查找以太网包首部中的类型域从而确定以太网协议所传送过来的数据类型

若数据类型是IP包，就将数据传给处理IP的子程序

若数据类型是ARP协议，就将数据传给ARP处理

IP模块的处理

应用层写入的数据

会经过表示层格式化编码

再由会话层标记发送顺序后才被发送出去

传输层为确保所传输的数据到达目标地址，会在通信两端的计算机之间进行确认，如果数据没有到达，它会负责进行重发

网络层与数据链路层都是基于目标地址将数据发送给接收端的，但是网络层负责将整个数据发送给最终目标地址，而数据链路层则只负责发送一个分段内的数据。

TCP是面向连接型，好比打电话，输入完对方电话号码拨出去之后，只有对端拿起电话才能真正通话，通话结束后将电话机扣上就如同切断电源

UDP是面向无连接型，如同去邮局寄包裹一样。负责处理邮递业务的营业员，不需要确认收件人的详细地址是否真的存在，也不需要确认收件人是否能收到包裹，只要发件人有一个寄件地址就可以办理邮寄包裹的业务

在实际的网络通信当中，每一层的协议所使用的址都不尽相同。TCP/IP通信中使用MAC地址、IP地址、端口号等信息作为地址标识。甚至在应用层中，可以将电子邮件地址作为网络通信的地址

地址特点：唯一性，层次性

MAC地址由设备的制造厂商针对每块网卡进行分别指定。人们可以通过制造商识别号、制造商内部产品编号以及产品通用编号确保MAC地址的唯一性。

IP地址由网络号和主机号两部分组成。通常，同处一个网段的主机也都层于同一个部门或集团组织。网络传输中，每个节点会根据分组数据的地址信息，来判断该报文应该由哪个网卡发送出去。为此，各个地址会参考一个发出接口列表。在这一点上MAC寻址与IP寻址是一样的。只不过MAC寻址中所参考的这张表叫做地址转发表，而IP寻址中所参考的叫做路由控制表

网卡

使计算机连网的设备，也称为Network Interface、网络适配器、LAN卡。一般会被集成到计算机的主板

中继器Repeater:

从物理层上延长网络的设备。只负责将电信号转换为光信号，因此不能传输速度不同的媒介之间转发。

网桥Bridge/2层交换机

识别数据链路层中的数据帧，重构数据帧转发。丢弃错误的数据帧

从数据链路层上延长网络的设备。能够识别数据链路层中的数据帧，并将这些数据帧临时存储于内存，再重新生成信号作为一个全新的帧转发给相连的另一个网段。

网桥会根据地址自学机制来判断是否需要转发数据帧，会记住曾经通过自己转发的所有数据帧的MAC地址，并保存到自己的内存表中。

路由器Router/3层交换机

判断网络路径，转发至目标地址

通过网络层转发分组数据的设备

网桥是根据物理地址MAC地址进行处理，而路由器则是根据IP地址进行处理

路由器可以连接不同的数据链路，如连接两个以太网，或连接一个以太网与一个FDDI

4-7层交换机：处理传输层以上各层网络传输的设备

负责传输层以上的数据转发

应用场景

负载均衡器：为了并发访问，服务器前端访问的入口地址通常只有一个（为了使用者的方便，只会向最终用户开放一个统一的访问URL）。为了通过同一个URL将前端访问分发到后台多个服务器上，可以在这些服务器的前端加上一个负载均衡器。这种负载均衡就是4-7层交换机的一种。还可以通过DNS实现负载均衡，通过对多个IP地址配置同一个名字，每次查询到这个名字的客户得到其中的某一个地址，从而使不同客户访问不同的服务器。该方法也称为循环复用DNS技术

带宽控制，广域网加速器、特殊应用访问加速以及防火墙等

网关(Gateway): 转换协议的设备

负责传输层以上的数据转发及其协议的转换

同4-7层交换机一样，负责处理传输层及以上的数据。负责将从传输层到应用层的数据进行转换和转发的设备。网关不仅转发数据还负责对数据进行转换，它通常会使用一个表示层或应用层网关，在两个不能进行直接通信的协议之间进行翻译，最终实现两者之间的通信

应用场景

为什么连到互联网的电脑与手机之间能够互发电子邮件呢？

互联网与手机之间设置了一道网关。网关负责读取完各种不同的协议后，对它们逐一进行合理的转换，再将相应的数据转发出去。这样一来即使应用的是不同电子邮件的协议，计算机与手机之间也能互相发送邮件

代理服务器：为了控制网络流量以及出于安全的考虑，也是网关的一种，称为应用网关。有了代理服务器，客户端与服务器之间无需在网络层上直接通信，而是从传输层到应用层对数据和访问进行各种控制和处理

防火墙就是一款通过网关通信，针对不同应用提高安全性的产品

带宽：低速数据链路就如同边道较少，高速数据链路就相当于有多个车道。传输速率又称为带宽

吞吐量：主机之间实际的传输速率，吞吐量这个词不仅衡量带宽，同时也衡量主机的CPU处理能力、网络的拥堵程度、报文中数据字段的占有份额等信息

托管主机

数据中心：高并发访问量，托管主机服务，为了减少访问延迟，会集合多个存储于一起，通过连接高速网络，以期提高响应速度。

虚拟化：抽奖网站等对网络资源的需求时刻都在发生变化。虚拟化技术是指当一个网站需要调整运营所使用的资源时，并不增减服务器、存储设备、网络等实际的物理设备，而是利用软件将这些物理设备虚拟化，在有必要增减资源的时候，通过软件按量增减的一种机制。通过此机制实现按需分配、按比例分配，对外提供可靠的服务

利用虚拟化技术，根据使用者的情况动态调整必要资源的机制被人们称作“云”