WWW - 超文本和超媒体

- 网络是一个 ***分布式超媒体*** 支持超媒体文件的交互式访问系统（又名 ***参考资料***）。

- *超媒体*（相对于 *超文本*）资源可能包含：

  - 不同类型的信息，包括：文字，图片，图形，音频等的例子包括：HTML文件，图像文件，查询结果等。

  - *超链接* 其他资源，

- 从网络编程的角度看，这些资源将被视为 ***数据***。

WWW和客户端 - 服务器模式

- 网络的分布式特性意味着，资源/数据都可能跨越多个在互联网上的计算机传播。

- 这很适合于客户端 - 服务器范例如下：

  - ***客户：*** 资源/数据的消费者是谁通常与交互的最终用户 *客户*

    应用程序被称为Web浏览器，

  - ***服务器：*** 资源库中，通常位于远程服务器级的机器，并获得这些资源通常由网络控制 *服务器。*

有待解决的问题

- 然而，这种资源分配还引入了许多潜在的问题：
  - 资源可以被更新，移动或删除，而不通知客户端应用程序，
  - 同样，资源之间的链接可能会更新，移动或删除，而不通知客户端应用程序，
  - 在远程服务器级的机器访问资源对网络带宽的占用影响。
- 这些问题可能会影响最终用户体验和网络。

客户端和服务器的交互 - HTTP

- Web浏览器和服务器之间通过交互的 *超文本传输协议*

  （HTTP）。

- 这是一个 *网络协议* 用于在Web上提供几乎所有的资源

  ：

  - Web浏览器客户端发送 ***HTTP请求***
    消息Web服务器。这些消息通常（但并不总是）包含一个资源
    请求，

  - Web服务器返回 ***HTTP响应*** 消息客户端。这些消息通常包含资源/数据
    （但不总是）。

Web浏览器和服务器的操作

- 为了欣赏潜在的问题，以开发Web浏览器和Web服务器时，要了解他们的操作是很重要的解决。

- Web服务器一遍遍执行简单的任务：
  - 他们接受来自客户端的HTTP请求
  - 他们返回HTTP响应，指示在处理请求的成功或失败的客户。

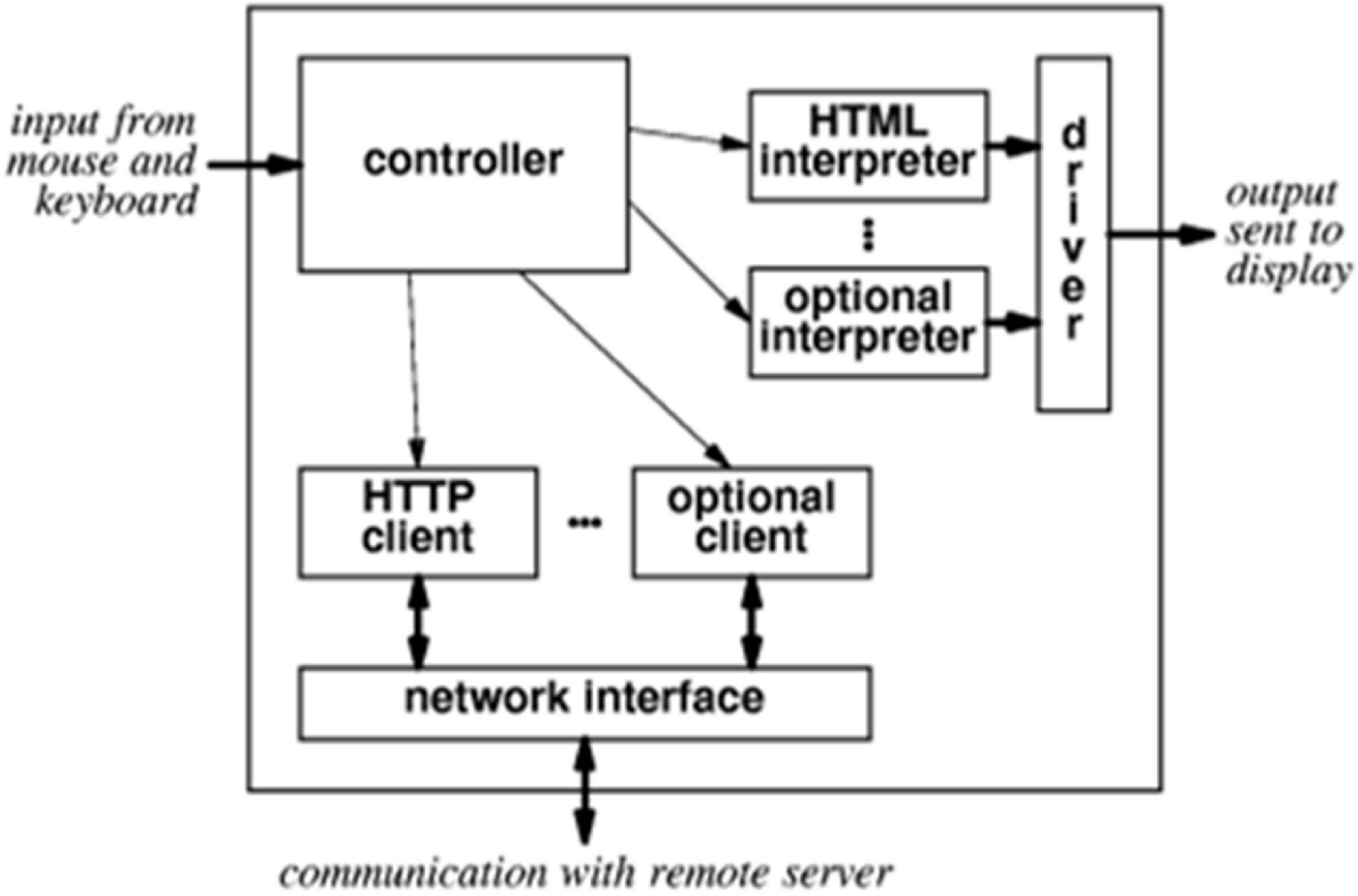
浏览器架构

- Web浏览器在他们的操作要复杂得多。这可以从他们的架构最清楚地看到（见下一页）。

- 浏览器的功能包括：
  - 渲染和向用户显示不同的（不同类型的）资源，
  - 用户交互用户，
  - 启动与Web服务器交互，获取资源，或者在某些情况下上传的资源。

- 该浏览器提供无缝使用许多软件组件的这些服务。

浏览器架构


input from mouse and keyboard
controller
HTML interpreter
optional interpreter
driver
output sent to display
HTTP client
optional client
network interface
communication with remote server

浏览器架构 - 续。

- 具体地，浏览器由以下部分组成：
  - 一套 *客户* 上载/检索资源，
  - 一套 *口译* 显示/渲染资源，
  - 一个 *调节器* 来管理他们所有。控制器负责：通过键盘和鼠标点击解释用户输入 和 在适当的时候调用解释器和客户端组件。

- 所有的浏览器最小包含一个基本的HTTP客户端，一个基本的HTML解释器和控制器。
  - 现代浏览器中包含更多。

解释器和客户端组件

- 一个例子 *翻译员* 是个 ***HTML解释：***
  - 它解析包含标准的HTML代码，文档和呈现内容到本地屏幕，
  - 需要显示的图片，视频，音频等其他解释
- 一个例子 *客户* 是个 ***HTTP客户端：***
  - 它是用来与HTTP服务器进行交互检索/上传资源的目的，
  - 需要其他客户发送/接收电子邮件，使用FTP文件传输等。
  - 1 ST 在目的地URL字段用于确定调用哪些客户端组件。

公文流转和HTTP

- 从网络规划角度来看，我们感兴趣的是HTTP客户端以及它与HTTP服务器的交互。

- 这种相互作用包括HTTP请求和响应的交流：

  - 这些通常被作为英语ASCII编码，即纯文本，

  - 这意味着，与协议分析器诸如Wireshark的观察时，所述应用数据字段可以容易地阅读和理解。

HTTP请求

- HTTP请求从HTTP客户端发起。
- 他们支持多项业务通过一组 ***方法：***

  - GET，HEAD，POST，OPTIONS，PUT，DELETE，TRACE和CONNECT。
  - 该模块的目的，我们将限制自己，GET和HEAD方法，
  - 这两种方法应充分论证的问题加以解决。

HTTP响应

- HTTP响应从HTTP服务器发起。
- 回想先前关于概述的链接断开，重新定位的资源和带宽限制的问题：
  - HTTP包含了很多功能来解决这些问题，
  - 服务器典型地发送附加信息与数据的每个传输，
  - 从这个附加信息的HTTP客户端可以：调用一个解释器来显示/呈现的资源数据，推断一个错误条件等

HTTP交易结构

- 最喜欢的网络协议，HTTP使用的客户端 - 服务器模型

  - HTTP客户端打开一个连接，并发送 *请求* 消息到HTTP服务器; 服务器返回 *响应* 消息，通常包含所请求的资源。
  - 提供响应，服务器后 *关闭* 连接。Web浏览器和Web服务器之间的连接的持续时间很短。这可能会导致在情况下额外的开销，其中一个浏览器必须返回到同一台服务器的多个文档/图片

  - 这也意味着，HTTP是一个 *无国籍* 协议，即不保持事务之间的连接信息。

- 的格式 *请求* 和 *响应* 消息是类似的，和英语为主。双方包括：

初始线，

HTTP消息的结构

- 回想一下，协议层的每个层规定被称为一个“成帧型”结构 *协议数据单元（PDU）：*
  - 实例包括：数据链路帧，一个IP数据报/包，TCP段等
- HTTP也是一个协议：
  - 它存在于应用层，
  - 有迹象表明，存在于应用层的许多其他协议。
- 在谈到应用层协议术语PDU没有实际意义。

HTTP消息的结构

- 这是因为应用层协议典型地遵循 *请求 - 响应* 要么 *命令响应* 互动的模式：
  - 通常，客户端从服务器请求某事或发出命令到服务器，
  - 通常服务器响应到客户端成功或失败的指示，
  - 但是，有时服务器发出请求和命令，但后来更多。
- 应用层协议在以下方面更有用的描述 ***句法*** 和 ***语义。***

语法和语义

- **句法** 描述的结构
  请求 - 响应 消息。
- **语义** 描述了客户端和服务器之间的交互：

  - 本质上，这涉及 序列 请求/响应消息的，

  - 更有用地这可以被描述为“ *谁首次会谈* 的”，即其侧发出的第一条消息。

HTTP消息的语法

- HTTP消息具有特定的结构或格式。

- 的格式 ***要求*** 和 ***回应***
  消息是类似的。双方包括：

  初始行，

  零个或多个标题行，一个空行，并

  一个可选的消息体通常从文件包含资源数据，或查询输出等

HTTP消息的语法

- 具体而言，一个HTTP消息的格式是：

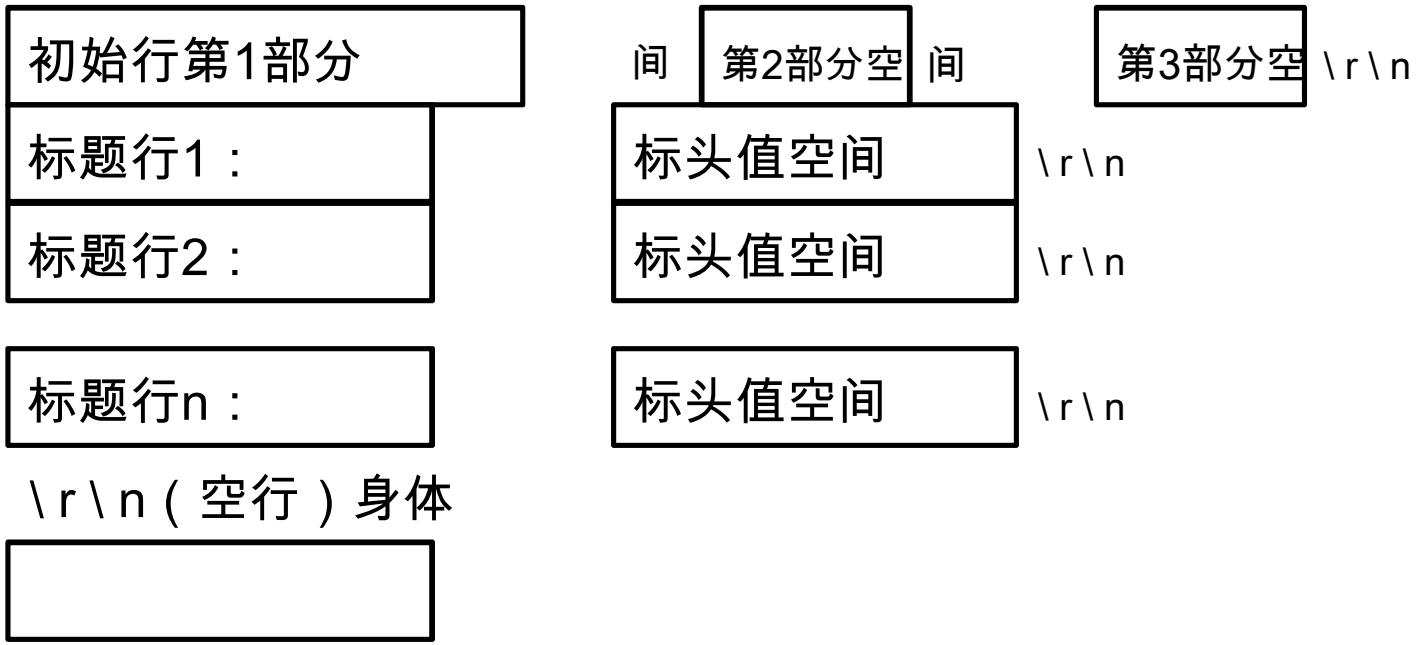
  <初始线，对于不同的 *请求* 和 *响应*>
  头1：数值头2：值2 He
  ader3：VALUE3

  <可选的消息体放在这里，像文件内容或查询数据>

- 初始线和标头中CRLF应该结束
  - 具体而言CR和LF这里平均ASCII值分别13和10。

HTTP消息的语法

- 这种结构可以在一个方面更有效地描述 *协议开箱图：*



初始 *请求* 线

- 用于初始行 请求 线具有三个部分，由空格隔开：
  - 一个方法名
  - 所请求的资源的本地路径
  - 正在使用的HTTP的版本。
- 典型的请求行是：

GET /path/to/file/index.html HTTP / 1.0

初始 *请求* 线

- 笔记：
  - GET是最常见的HTTP *方法*
    - 它说：“给我拿这个资源”
    - *方法* 名总是 大写 。
  - 该 *路径* 是URL的主机名之后的部分
  - 该HTTP *版* 始终把形式“ **HTTP / XX”** ，
    大写。

初始 *响应* 线

- 初始响应线还具有由空格分隔三个部分：
  - HTTP版本，
  - 一个回应 *状态代码* 给出的请求的结果，
  - 一个英国人 *原因短语* 描述状态码。
- 典型状态线路有：

  HTTP / 1.0 200 OK HTTP / 1.0 4

  04未找到

初始 *响应* 线

- 笔记：
  - 该HTTP *版* 是格式“ **HTTP / XX**”。
  - 该 *状态代码* 意在 *计算机可读*
    - 它包括一个三位整数，并且所述第一数字标识响应的普通类
  - 该 *原因短语* 意在 *人力阅读*， 并且可以变化。

标题行

- 标题行提供关于信息
  请求 要么 *响应*，或关于资源发送在消息主体中。

- 标题行是在一个特定的格式
  - 每头一行的形式是“页眉名称：值”，以结束CRLF。
  - 这是用于电子邮件类似的格式，并在RFC 822中定义。
  - 标题名称是 不 区分大小写（虽然值而定）。

标题行

- 有HTTP的两个版本：
  - HTTP 1.0是老年人和定义16个标头，尽管没有要求。
  - HTTP 1.1是较新的，并且限定46个标头，和一（主机:)在请求是必需的。

实施例请求标题行

Header Field Name	Description	Example
Accept	Content-Types that are acceptable for the response	`Accept: text/plain`
Cache-Control	Used to specify directives that *must* be obeyed by all caching mechanisms along the request-response chain	`Cache-Control: no-cache`
Connection	What type of connection the user-agent would prefer	`Connection: keep-alive`
Cookie	An HTTP cookie previously sent by the server with `Set-Cookie` (below)	`Cookie: $Version=1; Skin=new;`
Content-Length	The length of the request body in octets (8-bit bytes)	`Content-Length: 348`
Content-Type	The MIME type of the body of the request (used with POST and PUT requests)	`Content-Type: application/x-www-form-urlencoded`
Date	The date and time that the message was sent (in "HTTP-date" format as defined by RFC 7231)	`Date: Tue, 15 Nov 1994 08:12:31 GMT`
From	The email address of the user making the request	`From: user@example.com`
If-Modified-Since	Allows a *304 Not Modified* to be returned if content is unchanged	`If-Modified-Since: Sat, 29 Oct 1994 19:43:31 GMT`
If-None-Match	Allows a 304 Not Modified to be returned if content is unchanged, see HTTP ETag	`If-None-Match: "737060cd8c284d8af7ad3082f209582d"`
User-Agent	The user agent string of the user agent	`User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:12.0) Gecko/20100101 Firefox/21.0`

例如响应头系

Field name	Description	Example
Age	The age the object has been in a proxy cache in seconds	`Age: 12`
Cache-Control	Tells all caching mechanisms from server to client whether they may cache this object. It is measured in seconds	`Cache-Control: max-age=3600`
Connection	Options that are desired for the connection[20]	`Connection: close`
Content-Encoding	The type of encoding used on the data. See HTTP compression.	`Content-Encoding: gzip`
Content-Length	The length of the response body in octets (8-bit bytes)	`Content-Length: 348`
Content-Location	An alternate location for the returned data	`Content-Location: /index.htm`
Content-Range	Where in a full body message this partial message belongs	`Content-Range: bytes 21010-47021/47022`
Content-Type	The MIME type of this content	`Content-Type: text/html; charset=utf-8`
Date	The date and time that the message was sent (in "HTTP-date" format as defined by RFC 7231)	`Date: Tue, 15 Nov 1994 08:12:31 GMT`
ETag	An identifier for a specific version of a resource, often amessage digest	`ETag: "737060cd8c284d8af7ad3082f209582d"`
Expires	Gives the date/time after which the response is considered stale	`Expires: Thu, 01 Dec 1994 16:00:00 GMT`
Last-Modified	The last modified date for the requested object (in "HTTP-date" format as defined by RFC 7231)	`Last-Modified: Tue, 15 Nov 1994 12:45:26 GMT`
Location	Used in redirection, or when a new resource has been created.	`Location: http://www.w3.org/pub/WWW/People.html`
Server	A name for the server	`Server: Apache/2.4.1 (Unix)`
Set-Cookie	An HTTP cookie	`Set-Cookie: UserID=JohnDoe; Max-Age=3600; Version=1`

标题行 - 净，礼貌

- 考虑在下面的标题
  客户 要求：
  - 一个 **从：** 头给谁做真实的要求，或运行程序这样做的电子邮件地址。（这必须是用户可配置的，对于隐私的关注。）
  - 一个 **用户代理：** 报头识别真实发出请求的程序，其形式为“ **计划 - 名称/ X.XX**” X.XX为程序的（大部分）的字母数字版本。
  - 例如网景公司3.0发送标题“用户代理：Mozilla的/ 3.0Gold”。

标题行 - 净，礼貌

- 考虑在下面的标题
  服务器 对策：
  - 一个 **服务器：** 头。类似于 **用户代理：**
    标题：它标识形式的服务器软件“ **程序名/ X.XX”**。
  - 例如，Apache服务器会返回 “服务器：Apache / 1.2b3-dev的”。
  - 该 **最后修改：** 头给出了真实被返回的资源的修改日期（GMT）。它在高速缓存中。
  - 例如 **最后一次修改：周五，1999年12月31日23:59:59 GMT**

邮件正文

- 一个HTTP *信息* 可以具有标题行之后发送的数据的主体。

- 在一个 *响应：*
  - 这是所请求的资源被返回给客户端（最常见的用法消息体）
  - 或错误情况的一些说明文字。
- 在一个 *请求：*
  - 这是形式的数据或上传的文件被发送到服务器。

邮件正文

- 如果一个HTTP消息包括主体，通常有在该消息中标题行描述该机构。特别是，

  - 该 **内容类型：** 报头给出了MIME型在体内的数据，例如text / html的或图像/ GIF的。

  - 该 **内容长度：** 报头给出了主体的字节数。

其他HTTP方法 - HEAD和POST

- 另外两种常用的方法是HEAD和POST。

- HEAD方法
  - 类似的GET请求，但它要求服务器返回的响应头只，而不是实际的资源（即没有邮件正文）。
  - 有用的检查资源的特性，而无需实际下载。
  - 到HEAD请求的响应绝不能含有 *邮件正文*。

其他HTTP方法 - HEAD和POST

- POST方法
  - POST请求用于将数据发送到服务器以某种方式进行处理，像由CGI脚本。
  - 它不同于在以下方面GET请求：
    - 有与该请求发送的数据块时，在消息主体中。通常有多余的标题来形容这个消息体，如 **内容类型：** 和

      内容长度：。
    - 请求URI是不取回的资源; 它通常是一个程序来处理你要发送的数据。

GET http://www.comp.dit.ie/dbourke/ HTTP / 1.1的Accept-Language
：EN-US，连接; Q = 0.5
接受：文/ XML，应用/ XML，是application / xhtml + X
毫升，text / html的; Q = 0.9，文本/无格式; Q = 0.8，图像/ PNG。
接收字符集：ISO-8859-1，utf-8; Q = 0.7，*; Q = 0.7主机：w
ww.comp.dit.ie接受编码：gzip，放气

用户代理：Mozilla的/ 5.0（窗口; U; Windows NT的
5.1; EN-US; RV：1.8.0.1）的Gecko / 20060111 Firefox的/ 1.5
.0.1
饼干：PHPSESSID = 13ceaac67329048c4保持活动：3
00
代理连接：保持活跃

HTTP / 1.1 200 OK杂注：无缓存缓存控制：无
缓存MicrosoftOfficeWebServer：5.0_Pub的ET
ag：“e21ceefa6e28df”接受范围：字节的Con
tent-Type：text / html的连接：关闭

日期：星期三，2008年10月22日14时20分十二秒格林尼治标准时间
服务器：Microsoft-IIS / 6.0 Content-Location中：

http://www.comp.dit.ie/dbourke/index.htm的Last-Modified：周四
，2008年10月02日9点12分23秒GMT的Content-Length：1837年X-技
术 - 通过：ASP.NET

从松饼获得的数据