计算机网络

自顶向下方法

第一章 计算机网络和因特网

1.1 构成

主机（或端系统），通信链路，分组交换机，路由器，ISP（Internet Service Provider），协议

协议（protocol）定义了在两个或多个通信实体之间交换的报文的格式和顺序，以及报文发送和/或接收一条报文或其他事件所采取的动作。

1.2 网络边缘

端系统也称为主机，因为它们容纳（即运行）应用程序。主机有时进一步被划分为两类：客户机和服务器。

### 1.2.1 接入网

接入网是指将端系统物理连接到其边缘路由器的网络。边缘路由器是端系统到任何其他远程端系统的路径上的第一台路由器。

家庭接入:DSL、电缆、FTTH、拨号和卫星

今天，宽带住宅接入有两种最流行的类型：**数字用户线**（Digital Subscriber Line，DSL）和**电缆**。

住户通常从提供本地电话接入的本地电话公司处获得**DSL因特网接入**。每个用户的DSL调制解调器使用现有的电话线（即双绞铜线）与位于电话公司的本地中心局中的数字用户线接入复用器（DSLAM）交换数据。

**电缆因特网接入**利用了有线电视公司现有的有线电视基础设施。光缆将电缆头端连接到地区枢纽，从这里使用传统的同轴电缆到达各家各户和公寓。因为这个系统中应用了光纤和同轴电缆，因此被称为**混合光纤同轴**（Hybrid Fiber Coax， HFS）系统。

光纤到户（Fiber To The Home，FTTH）速率更高，从本地中心局直接到家庭提供了一条光纤路径。

企业（和家庭）接入：以太网和WiFi

广域无线接入：3G和LTE

### 1.2.2 物理媒体

物理媒体分两种类型：导引型媒体和非导引型媒体。

对于导引型媒体，电波沿着固体媒体前行，如光缆、双绞铜线或同轴电缆。

对于非导引型媒体，电波在空气或外层空间中传播，例如在无线局域网或数字卫星频道中。

双绞铜线

双绞线有两根绝缘的铜线组成，每根大约1mm粗，以规则的螺旋状排列着。这两根线被绞合起来，以减少临近类似的双绞线的电气干扰。

同轴电缆

同轴电缆由两个铜导体组成，但是这两个导体是同心的而不是并行的。借助于这种结构及特殊的绝缘体和保护层，同轴电缆能够达到较高的数据传输速率。同轴电缆能被用作导引型共享媒体，特别是，许多端系统能够直接与该电缆相连，每个端系统能接收由其他端系统发送的内容。

光纤

光纤是一种细而柔软的、能够导引光脉冲的媒体，每个脉冲表示一个比特。

陆地无线电信道

无线电通信承载电磁频谱中的信号。它不需要安装物理线路，并具有穿越墙壁、提供与移动用户的连接以及长距离承载信号的能力。无线电信道极大的依赖于传播的环境和信号传输的距离。

卫星无线电信道

一颗通信卫星连接地球上的两个或多个微波发射器/接收器，它们被称为地面站。该卫星在一个频段上接收传输，使用一个转发器再生信号，并在另一个频率上发射信号。

通信中常使用两类卫星：同步卫星和近地轨道卫星。

## 1.3 网络核心

通过网络链路和交换机移动数据有两种基本方法：电路交换和分组交换

### 1.3.1 分组交换

在各种网络应用中，端系统彼此交互报文。报文能够包含协议设计者需要的任何东西。报文可以执行一种控制功能，也可以包含数据。

为了从源端系统向目的端系统发送一个报文，源将长报文划分为较小的数据块，称之为**分组**。在源和目的地之间，每个分组都通过通信链路和分组交换机传送。（交换机主要有两类：**路由器**和**链路层交换机**。）分组以等于该链路**最大传输速率**的速度传输通过通信链路。因此，如果某源端系统或分组交换机经过一条链路发送一个L比特的分组，链路的传输速率为R比特/秒，则传输该分组的时间为L/R秒。

存储转发传输

**存储转发传输**是指在交换机能够开始向输出链路传输该分组的第一个比特之前，必须接收到整个分组。

排队时延和分组丢失

**输出缓冲**：每台分组交换机有多条链路与之相连。对于每条相连的链路，该分组交换机具有一个**输出缓存**（也称为**输出队列**），它用于存储路由器准备发往那条链路的分组。

**排队时延**：如果到达的分组需要传输到某条链路，但发现该链路正忙于传输其他分组，该到达分组必须在输出缓存中等待，由此产生的时延为排队时延。

**分组丢失**：因为缓存空间的大小有限，一个到达的分组可能发现该缓存已被其他等待传输的分组完全充满了。在此情况下，将出现分组丢失（丢包），到达的分组或已经排队的分组之一将被丢弃。

转发表和路由选择协议

转发表：每台路由器具有一个转发表，用于将目的地址（或目的地址的一部分）映射成为输出链路。

路由选择协议：一个路由选择协议可以决定从每台路由器到每个目的地的最短路径，并使用这些最短路径结果来配置路由器中的转发表。

### 1.3.2 电路交换

在电路交换网络中，在端系统通信会话期间，预留了端系统间沿路径通信所需要的资源（缓存，链路传输速率）。

在分组交换网络中，这些资源则不是预留的；会话的报文按需使用这些资源，其后果是可能不得不等待（即排队）进入通信线路。

在电路交换网络中，必须在发送方和接收方之间建立一条连接。这是一条名副其实的链接，因为在沿着发送方和接收方之间的路径上都将为该连接维护连接状态，即一条**电路**。当网络创建这种电路时，它也在连接期间在该网络链路上预留了恒定的传输速率（表示为每条链路传输容量的一部分）。既然已经为该发送方—接收方连接预留了带宽，则发送方能够以**确保的恒定速率**向接收方传送数据。

电路交换网络中的复用

链路中的电路时通过频分复用（Frequency-Division Multiplexing，FDM）或时分复用（Time-Division Multiplexing，TDM）来实现的。

对于FDM，链路的频谱由跨越链路创建的所有连接共享。特别是，在连接期间链路为每条连接专用一个频段。这个频段的带宽通常为4kHz。该频段的宽度称为带宽。

对于一条TDM链路，时间被划分为固定时间的帧，并且每个帧又被划分为固定数量的时隙。当网络跨越一条链路创建一条连接时，网络在每个帧中为该连接制定一个时隙。这些时隙专门由该连接单独使用，一个时隙（在每个帧内）可用于传输该连接的数据。

分组交换与电路交换的对比

分组交换优缺点（引用他人的观点）：

优点：1）它提供了比电路交换更好的带宽共享；

2）它比电路交换更简单、更有效，实现成本更低。

缺点：分组交换不适合实时服务（例如，电话和视频会议），因为他的端到端时延是可变的和不可预测的（主要是因为排队时延的变动和不可预测所致）。

**作者总结：**

1）分组交换的性能能够优于电路交换的性能。

2）电路交换不考虑需求，而预先分配了传输链路的使用，这使得已分配而并不需要的链路时间并未被利用。另一方面，分组交换按需分配链路使用。链路传输能力将在所有需要在链路上传输分组的用户之间逐步地被共享。

### 1.3.3 网络的网络

今天的因特网是一个网络的网络，其结构复杂，由十多个第一层ISP和数十万个较低层ISP组成。较低层ISP与较高层ISP相连，较高层ISP彼此互联。用户和内容提供商是较低层ISP的客户，较低层ISP是较高层ISP的客户。

## 1.4 分组交换网中的延迟、丢包和吞吐量

### 1.4.1 分组交换网中的时延概述

时延的类型

（1）处理时延

检查分组首部和决定将该分组导向何处所需要的时间是处理时延的一部分。处理时延也可能包括其他因素：如检查比特级别的差错所需要的时间，该差错出现在从上游节点向路由器A传输这些分组比特的过程中。

（2）排队时延

在队列中，当分组在链路上等待传输时，它经受排队时延。一个特定分组的排队时延长度将取决于先期到达的正在排队等待向链路传输的分组数量。

（3）传输时延

假定分组以先到先服务方式传输，仅当所有已经到的分组被传输后，才能传输刚到达的分组。用L比特表示该分组的长度，用R bps表示从路由器A到路由器B的链路传输速率，则传输时延是L/R。这是将所有分组的比特推向链路（即传输）所需要的时间。

（4）传播时延

一旦一个比特被推向链路，该比特需要向路由器B传播。从该链路的起点到路由器B传播所需要的时间是传播时延。该比特以链路的传播速率传播。该传播速率取决于该链路的物理媒体（即光纤、双绞铜线等），其速率范围是2 x 108~3 x 108m/s。

### 1.4.2 排队时延和丢包

流量强度

排队时延大小很大程度取决于流量到达该队列的速率、链路的传输速率和到大流量的性质，即流量是周期性到达还是以突发形式到达。

令a表示分组到达队列的平均速率（单位为分组/秒），传输速率为Rbps，所有分组大小均为L比特。比率La/R称为流量强度，它在估计排队时延的范围方面经常起到重要的作用。如果La/R>1，则比特到达队列的平均速率超过从该队列传输出去的速率。在这种不幸的情况下，该队列趋向于无限增加，并且排队时延将趋向于无穷大！因此：设计系统时流量强度不能大于1。

丢包

随着流量强度接近于1，排队时延并不真正趋向无穷大。

相反，到达的分组将发现一个满的队列。由于没有地方存储这个分组，路由器将丢弃该分组，即该分组会丢失。

### 1.4.3 端到端时延

Traceroute

用于计算端到端时延，其原理如下：

假定在源和目的地之间有N-1台路由器。源将向网络发送N个特殊的分组，其中每个分组地址指向最终目的地。这N个特殊分组标识为从1到N，第一个分组标识为1，第N个分组标识为N。当第N台路由器接收到标识为n的第n个分组时，该理由起路由器不是向它的目的地转发该分组，而是想源回送一个报文。当目的主机接收第N个分组时，它也会向源返回一个报文。该源记录了从它发送一个分组到它接收到对应返回报文所经历的时间；它也记录了返回该报文的路由器（或目的主机）的名字和地址。

### 1.4.4 计算机网络中的吞吐量

瞬时吞吐量：主机在任何时间瞬间接收到文件的速率。

平均吞吐量：主机接收到所有F比特用去T秒，则平均吞吐量为F/T bps。

## 1.5 协议层次及其服务模型

### 1.5.1 分层的体系结构

1.协议分层

各层的所有协议被称为**协议栈**。因特网的协议栈由五个层次组成：物理层、链路层、网络层、运输层和应用层。

（1）应用层

应用层是网络应用程序及它们的应用层协议存留的地方。因特网的应用层协议包括：HTTP、SMTP、FTP和DNS等。

应用层协议分部在多个端系统上，而一个端系统中的应用程序使用协议与另一个端系统中的应用程序交换信息分组。

应用层的信息分组称为**报文**（message）。

（2）运输层

因特网的运输层在应用程序端点之间传送应用层报文。有两种因特网运输协议：TCP和UDP。

TCP向它的应用程序提供了面向连接的服务。这种服务包括了应用层报文向目的地的确保传递和流量控制（即发送方/接收方速率匹配）。TCP也将长报文划分为短报文，并提供拥塞控制机制。

UDP协议向它的应用程序提供无连接服务。这是一种不提供不必要服务的服务，没有可靠性，没有流量控制，也没有拥塞控制。

运输层分组称为报文段（segment）。

（3）网络层

网络层负责将网络层分组—数据报（datagram）从一台主机移动到另一台主机。在一台源主机中的因特网运输层协议（TCP或UDP）向网络层递交运输层报文段和目的地址。

（4）链路层

因特网的网络层通过源和目的地之间的一系列路由器路由数据报。为了将分组从一个节点（主机或路由器）移动到路径上的下一个节点，网络层必须依靠该链路层的服务。特别是在每个节点，网络层将数据报下传给链路层，链路层沿着路径将数据报传递给下一个节点。在该下一个节点，链路层将数据报上传给网络层。

由链路层提供的服务取决于应用于该链路的特定链路层协议。如以太网、WiFi和PPP协议等。

链路层分组称为帧。

（5）物理层

链路层的任务是将整个帧从一个网络元素移动到邻近的网络元素，而物理层的任务是将该帧中的一个个比特从一个节点移动到下一个节点。在这层中的协议任然是链路相关的，并且进一步与该链路（如双绞铜线，单模光纤）的实际传输媒体相关。

2. OSI模型

开发系统互联模型（OSI）包括七层协议：应用层、表示层、会话层、运输层、网络层、链路层和物理层。

表示层的作用是使通信能够解释交换数据的含义。这些服务包括数据压缩和数据加密以及数据描述（这使得应用程序不必担心在个台计算机中表示/存储的内部各式不同的问题）。

会话层提供了数据交换的定界和同步功能，包括了建立检查点和恢复方案的方法。

### 1.5.2 封装

在发送主机端，一个**应用层报文**被发送给运输层。在最简单的情况下，运输层收取到报文并附上附加信息，该首部将被接收端的运输层使用。应用层报文和运输层首部信息一道构成了**运输层报文段**。运输层报文段因此封装了应用层报文。

附加的信息也许包括了下列信息：允许接收端运输层向上向适当的应用程序交付报文的信息；差错检测位信息，该信息让接收方能够判断报文中的比特是否在途中已被改变。

运输层则向网络层传递该报文段，网络层增加了如源和目的端系统地址等网络层首部信息，生成了**网络层数据报**。

该数据报接下来被传递给链路层，链路层增加它自己的链路层首部信息并生成链路层帧。

所以，在每一层，一个分组具有两种类型的字段：**首部字段**和**有效载荷字段**。有效载荷通常是来自上一层的分组。

第二章 应用层

## 2.1 应用层协议原理

### 2.1.1 网络应用程序体系结构

应用程序体系结构由应用程序研发者设计，规定了如何在各种端系统上组织该应用程序。两种主流体系结构：**客户-服务器体系结构**、**对等（P2P）体系结构**。

在**客户-服务体系结构**中，有一个总是打开的主机成为服务器，他服务于来自许多其他称为客户的主机的请求。

在一个P2P体系结构中，对位于数据中心的专用服务器有最小的依赖。相反，应用程序在间断连接的主机对之间使用直接通信，这些主机对被称为**对等方**。

### 2.1.2 进程通信

在同一台主机上的进程间的通信方式有：管道，消息队列，信号量，共享内存

在两个不同端系统上的进程，通过跨越计算机网络交换报文而相互通信。发送进程生成并向网络中发送报文；接收进程接收这些报文并可能通过回送报文进行响应。

1. 客户和服务器进程

在一对进程之间的通信会话场景中，发起通信（即在该会话开始时发起与其他进程的联系）的进程被标识为**客户**，在会话开始时等待联系的进程是**服务器**。

2. 进程与计算机网络之间的接口

进程通过一个称为**套接字**的软件接口向网络发送报文和从网络接收报文。套接字是同一台主机内应用层与传输层之间的接口。由于该套接字是建立网络应用程序的可编程接口，因此套接字也称为应用程序和网络之间的**应用程序编程接口**。

应用程序开发者可以控制套接字在应用层端的一切：①选择运输层协议；②也许能设定几个运输层参数，如最大缓存和最大报文段长度。

3. 进程寻址

标识接收进程，需要定义两种信息：①主机的地址；②在目的主机中指定接收进程的标识符。

在因特网中，主机由其IP地址标识，进程由端口号标识。

常见的应用程序对应的端口号如下：

表2.1 常见服务端口号

|  |  |
| --- | --- |
| 服务名称 | 端口号 |
| FTP | 21 |
| SSH、SCP | 22 |
| Telnet | 23 |
| SMTP | 25 |
| DNS | 53 |
| HTTP | 80 |
| POP3 | 110 |
| HTTPS | 443 |

### 2.1.3 可供应用程序使用的运输服务

1.可靠数据传输

定义：确保应用程序的一端发送的数据正确、完全地交付给该应用程序的另一端。

运输层协议能够潜在地向应用程序提供的一个重要服务是进程到进程的可靠数据传输。当一个运输协议提供这种服务时，发送进程只要将其数据传递进套接字，就可以完全相信该数据将能无差错地到达接收进程。

2. 吞吐量

在沿着一条网络路径上的两个进程之间的通信会话场景中，可用吞吐量就是发送进程能够向接收进程交付比特的速率。因为其他会话将共享沿着该网络路径的带宽，并且因为这些会话将会到达和离开，该可用吞吐量将**随时间波动**。

具有吞吐量要求的应用程序被称为**带宽敏感的应用**。

**弹性应用**能够根据当时可用的带宽或多或少地利用可供使用的吞吐量。

3. 定时

运输层协议也能提供定时保证。一个保证的例子如：发送方注入进套接字中的每个比特到达接收方的套接字不迟于100ms。

4. 安全性

最后，运输协议能够为应用程序提供一种或多种安全性服务。例如，在发送主机中，运输协议能够加密由发送进程传输的所有数据，在接收主机中，运输层协议能够在将数据交付给接收进程之前解密这些数据。

### 2.1.4 因特网提供的运输服务

因特网为应用程序提供两个传输层协议，即TCP和UDP。

1. TCP服务

TCP服务模型包括面向连接服务和可靠数据传输服务。

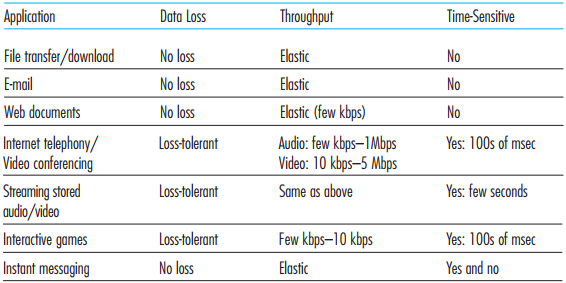


图2-1 选择的网络应用的要求

* **面向连接的服务**：在应用层数据报文开始流动之前，TCP让客户和服务器互相交换运输层控制信息。这个所谓的握手过程提醒客户和服务器，让它们为大量分组的到来做好准备。在握手阶段后，一个TCP连接就在两个进程的套接字之间建立了。这条连接是全双工的，即连接双方的进程可以在此连接上同时进行报文收发。当应用程序结束报文发送时，必须拆除该连接。
* **可靠的数据传输服务**：通信进程能够依靠TCP，无差错、按适当顺序交付所有发送的数据。当应用程序的一端将字节流传进套接字时，它能够依靠TCP将相同的字节流交付给接收方的套接字，而没有字节的丢失和冗余。

2. UDP服务

UDP是一种不提供不必要服务的轻量级运输协议，它仅提供最小服务。UDP是无连接的，因此在两个进程通信前没有我收过程。UDP协议提供一种不可靠数据传输服务，当进程将一个报文发送进UDP套接字时，UDP并不保证该报文将到达接收进程。不仅如此，到达接收进程的报文也可能是乱序到达的。

3. 因特网运输协议所不提供的服务

今天的因特网通常能够为时间敏感应用提供满意的服务，但它不提供任何定时或带宽保证。

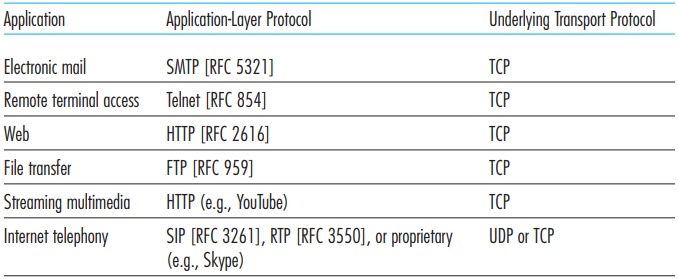


图2-2 流行的网络应用及其应用层协议和支撑的运输协议

### 2.1.5 应用层协议

应用层协议定义了运行在不同端系统上的应用程序进程如何相互传递报文。特别是定义了：

* 交换的报文类型，例如请求报文和响应报文
* 各种报文类型的语法，如报文中的各个字段及这些字段是如何描述的
* 字段的语义，即这些字段中的信息的含义
* 确定一个进程何时以及如何发送报文，对报文进行响应的规则。

## 2.2 Web和HTTP

### 2.2.1 HTTP概况

Web的应用层协议是**超文本传输协议**（HyperText Transfer Protocol， HTTP），它是Web的核心。

Web页面是由对象组成的。一个对象只是一个文件，如一个HTML文件、一个JPEG图形，一个Java小程序这样的文件，切它们可通过一个URL地址寻址。多数Web页面含有一个HTML基本文件以及几个引用对象。

HTTP使用TCP作为它的支撑运输协议。HTTP客户首先发起一个与服务器的TCP连接。一旦连接建立，该浏览器和服务器进程就可以通过套接字接口访问TCP。

**无状态协议**：服务器向客户发送被请求的文件，而不存储任何关于该客户的状态信息。假如某个特定的客户在短短的几秒内两次请求同一个对象，服务器并不会因为刚刚为客户提供了该对象就不再作出反应，而是重新发送该对象。

为什么说HTTP是无状态的协议呢？因为它的每个请求都是完全独立的，每个请求包含了处理这个请求所需的完整的数据（如主机地址，鉴权信息等），发送请求不涉及到状态变更。

### 2.2.2 非持续连接和持续连接

**非持续连接**：每个请求/响应对是经一个单独的TCP连接发送；

**持续连接**：所有的请求及其响应经相同的TCP连接发送。

HTTP默认情况下使用持续连接。

1. 采用非持续连接的HTTP

服务器向客户传送一个Web页面的步骤。假设该页面包含一个HTML基本文件和10个JPEG图形，并且这11个对象位于同一台服务器上。假设HTML文件的URL为：http://www.someSchool.edu/someDept/home.index。

1）HTTP客户进程在端口号80发起一个到服务器www.someSchool.edu的TCP连接，该端口号是HTTP的默认端口。在客户和服务器上分别有一个套接字与该链接相关联。

2）HTTP客户经它的套接字向该服务器发送一个HTTP请求报文。请求报文中包含路径名/someDept/home.index。

3）HTTP服务器进程经它的套接字接收该请求报文，从其存储器检索出对象www.someSchool.edu/someDept/home.index，在一个HTTP响应报文中封装对象，并通过其套接字向客户发送相应报文。

4）HTTP服务器进程通知TCP断开该TCP连接。（但是直到TCP确认客户已经完整地收到响应报文为止，它才会实际终端连接。）

5）HTTP客户接受响应报文，TCP连接关闭。该报文指出封装的对象是一个HTML文件，客户从响应报文中提取出该文件，检查该HTML文件，得到对10个JPEG图形的引用。

6）对每个引用的JPEG图形对象重复前4个步骤。

事实上，用户能够配置现代浏览器来控制连接的并行度。在默认方式下，发部分浏览器打开5~10个并行的TCP连接，每条连接处理一个请求响应事务。

往返时间（Round-Trip Time，RTT）：该时间是指一个短分组从客户到服务器然后再返回客户所花费的时间。RTT包括传播时延、分组在中间路由器和交换机上的排队时延以及分组处理时延。

TCP连接涉及“三次握手”过程，即客户向服务器发送一个小TCP报文段，服务器用一个小TCP报文段作出确认和响应，最后，客户向服务器返回确认。三次握手中前两个部分耗费一个RTT时间。完成了三次握手的前两个部分后，客户结合三次握手的第三部分向该TCP发送一个HTTP请求报文。一旦该请求报文到达服务器，服务器就在该TCP连接上发送HTTP文件。该HTTP请求/响应用去了另一个RTT。

TCP连接细节见第三章。

2. 采用持续连接的HTTP

非持续连接有一些缺点：

1）必须为每一个请求的对象建立和维护一个全新的连接。对于每个这样的连接，在客户和服务器中都要分配TCP的缓冲区和保持TCP变量，这个Web服务器带来了严重的负担。

2）每一个对象经受两倍RTT的交付时延，即一个RTT用于创建TCP，另一个RTT用于请求和接收一个对象。

在采用HTTP1.1持续连接的情况下，服务器在发送响应后保持该TCP连接打开。在相同的客户与服务器之间，后续的请求和响应报文能够通过相同的连接进行传送。一般来说，如果一条连接经过一段时间间隔（一个可配置的时间间隔）仍未被使用，则HTTP服务器就关闭该连接。

HTTP的**默认模式**是使用带**流水线的持续连接**。

### 2.2.3 HTTP报文格式

1. HTTP请求报文

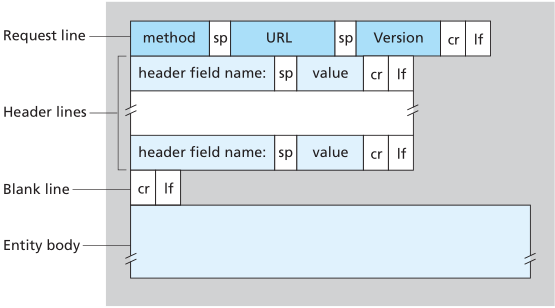


图2-3 HTTP请求报文的通用格式

HTTP请求报文的第一行叫**请求行**，其后继的行叫做**首部行**。

**1）请求行**

请求行有3个字段：方法字段、URL字段和HTTP版本字段。

方法字段的取值包括：GET、POST、HEAD、PUT和DELETE。

**2）首部行**

Host：指明对象所在的主机。

Connection：值close要求服务器在发送完被请求的对象后就关闭这条连接。

User-agent：指明用户代理，即向服务器发送请求的浏览器类型。

Accept-language：表示用户想要得到该对象的法语版本。

**3）实体体**

位于首部行（和附加的回车和换行）后。使用GET方法时实体体为空，而使用POST方法时才使用实体体。

HEAD方法类似于GET方法。当服务器收到一个使用HEAD方法的请求时，将会用一个HTTP报文进行响应，但是并不返回请求对象。应用程序开发者常用HEAD方法进行调试跟踪。

PUT方法常与Web发行工具联合使用，它允许用户上传对象到指定的Web服务器上指定的路径（目录）。

DELETE方法允许用户或者应用程序删除Web服务器上的对象。

2. HTTP响应报文

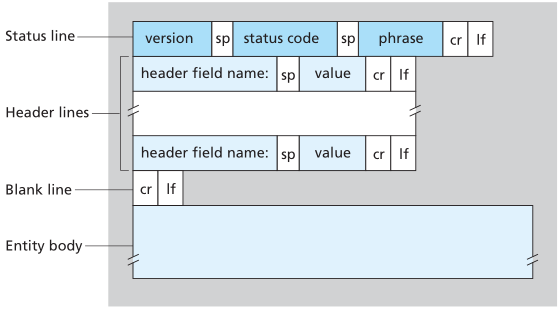


图2-4 HTTP 响应报文的通用格式

响应报文包含三部分：初始状态行、6个首部行、实体体。

**1）初始状态行**

协议版本字段、状态码和响应状态信息。

常见的状态码及含义见[附录1](#_附录1_HTTP常见状态码及含义)

**2）首部行**

Connection：close首部行告诉用户，发送完报文后将关闭该TCP连接。

Date：指示服务器产生并发送该响应报文的日期和时间。这个时间不是指对象创建或者最后修改的时间，而是服务器从它的文件系统中检索到该对象，将该对象插入响应报文并发送该响应报文的时间。

Server：指示发送该报文的服务器类型，类似于请求报文中的User-agent首部行。

Last-modified：指示了对象创建或者最后修改的日期和时间。

Content-Length：指示了被发送对象中的字节数。

Content-Type：指示了实体体中的对象类型。

**3）实体体**

包含了报文的主要部分，即它包含了所请求的对象本身。

### 2.2.4 用户与服务器的交互：cookie

目的：Web站点希望能够识别用户，可能是因为服务器希望限制用户的访问，或者因为它希望把内容与用户身份联系起来。

cookie技术有四个组件：①在HTTP响应报文中的一个cookie首部行；②在HTTP请求报文中的一个cookie首部行；③在用户端系统中保留有一个cookie文件，并由用户的浏览器进行管理；④位于web站点的一个后端数据库。

服务器用一个包含Set-cookie的首部行设置cookie。

### 2.2.5 Web缓存

Web缓存器也叫代理服务器，它是能够代表初始Web服务器来满足HTTP请求的网络实体。Web服务器有自己的磁盘存储空间，并在存储空间中保存最近请求过的对象的副本。

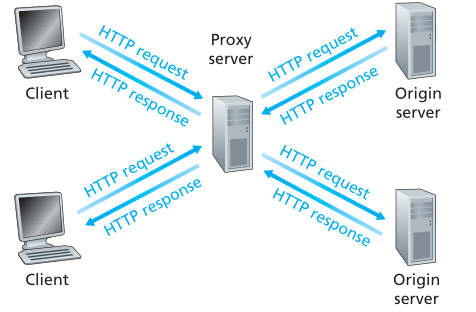


图2-5 客户通过Web缓存器请求对象

**Web缓存器的工作原理**：

假设浏览器请求对象http://www.someschool.edu/campus.gif

1）浏览器创建一个到Web缓存器的TCP连接，并向Web缓存器中的对象发送一个HTTP请求。

2）Web缓存器进行检查，看看本地是否存储了该对象的副本。如果有，Web缓存器就想客户浏览器用HTTP响应报文返回该对象。

3）如果Web缓存器中没有该对象，它就打开一个与该对象的初始服务器的TCP连接。Web缓存器则在这个缓存起到服务器的TCP连接上发送一个对该对象的HTTP请求。在收到该请求后，初始服务器向该Web缓存器发送具有该对象的HTTP响应。

4）当Web缓存器接收到该对象时，它在本地存储空间存储一份副本，并向客户的浏览器用HTTP响应报文发送该副本（通过现有的客户浏览器和Web缓存器之间的TCP连接）。

**在因特网上部署Web缓存器的原因**：

（1）Web缓存器可以大大减少对客户请求的响应时间，特别是当客户与初始服务器的瓶颈带宽远低于客户与Web缓存器之间的瓶颈带宽时更是如此。

（2）Web缓存器能够大大减少一个机构的接入链路到因特网的通信量。通过减少通信量，该机构就不必急于增加带宽，因此降低了费用。

### 2.2.6 条件GET方法

目的：确保客户端接收到的Web缓存器上的对象是最新的。

条件：①请求报文使用GET方法；②请求报文中包含了一个“If-Modified-Since：”首部行。

工作原理：

1）Web缓存器向服务器请求一个对象后，服务器在响应报文中添加**Last-Modified**首部行，表示该对象最后修改日期，Web缓存器将该对象存储在本地存储空间中。

HTTP/1.1 200 OK

Date: Sat, 8 Oct 2011 15:39:29

Server: Apache/1.3.0 (Unix)

Last-Modified: Wed, 7 Sep 2011 09:23:24

Content-Type: image/gif

2）经过一段时间后，另一个用户请求同一个对象。由于在这段时间里Web服务器上的该对象可能已经被修改了，该缓存器通过发送一个条件GET执行最新检查。

GET /fruit/kiwi.gif HTTP/1.1

Host: www.exotiquecuisine.com

If-modified-since: Wed, 7 Sep 2011 09:23:24

If-modified-since首部行的值正好等于此前服务器发送的响应报文中Last-Modified首部行的值。

该条件豹纹告诉服务器，仅当自指定日期之后该对象被修改过，才发送该对象。若该对象在这段时间内没有被修改，则服务器发送一个响应报文：

HTTP/1.1 304 Not Modified

Date: Sat, 15 Oct 2011 15:39:29

Server: Apache/1.3.0 (Unix)

状态行304 Not Modified告诉缓存器可以使用该对象，能向请求的浏览器转发它缓存的该对象副本。

## 2.3 因特网中的电子邮件

### 2.3.1 SMTP

### 2.3.2 与HTTP的对比

### 2.3.3 邮件报文格式

### 2.3.4 邮件访问协议

## 2.4 DNS：因特网的目录服务

### 2.4.1 DNS提供的服务

### 2.4.2 DNS工作原理

### 2.4.3 DNS记录和报文

## 2.5 P2P文件分发

## 2.6 视频流和内容分发网

### 2.6.1 因特网视频

# 附录1 HTTP常见状态码及含义

|  |  |  |  |
| --- | --- | --- | --- |
| 状态码系列 | 状态码 | 状态信息 | 含义 |
| 1XX：通知  1XX系列响应代码仅在与HTTP服务器沟通时使用。 | 100 | Continue | 客户端应重新发送初始请求，并在请求中附上第一次请求时未提供的（可能很大或者包含敏感信息的）表示。客户端这次发送的请求不会被拒绝 |
| 101 | Switching Protocols | 当客户端通过在请求里使用Upgrade报头，以通知服务器它想改用除HTTP协议之外的其他协议时，客户端将获得此响应代码 |
| 2XX: 成功  2XX系列响应代码表明操作成功了 | 200 | OK | 表示服务器成功执行了客户端所请求的动作 |
| 201 | Created | 当服务器依照客户端的请求创建了一个新资源时，发送此响应代码 |
| 202 | Accepted | 客户端的请求无法或将不被实时处理，请求稍后会被处理 |
| 203 | Non-Authoritative Information | 这个响应代码跟200一样，只不过服务器想让客户端知道，有些响应报头并非来自该服务器--他们可能是从客户端先前发送的一个请求里复制的，或者从第三方得到的。 |
| 204 | No Content | 若服务器拒绝对PUT、POST或者DELETE请求返回任何状态信息或表示，那么通常采用此响应代码。服务器也可以对GET请求返回此响应代码，这表明“客户端请求的资源存在，但其表示是空的”。注意与304("Not Modified")的区别。204常常用在Ajax应用里。服务器通过这个响应代码告诉客户端：客户端的输入已被接受，但客户端不应该改变任何UI元素。 |
| 3XX 重定向  3XX系列响应代码表明：客户端需要做些额外工作才能得到所需要的资源。它们通常用于GET请求。他们通常告诉客户端需要向另一个URI发送GET请求，才能得到所需的表示。那个URI就包含在Location响应报头里。 | 301 | Moved Permanently | 服务器知道客户端试图访问的是哪个资源，但它不喜欢客户端用当前URI来请求该资源。它希望客户端记住另一个URI，并在今后的请求中使用那个新的URI。你可以通过这个响应代码来防止由于URI变更而导致老URI失效。 |
| 303 | See Other | 请求已经被处理，但服务器不是直接返回一个响应文档，而是返回一个响应文档的URI。该响应文档可能是一个静态的状态信息，也可能是一个更有趣的资源。对于后一种情况，303是一种令服务器可以“发送一个资源的表示，而不强迫客户端下载其所有数据”的方式。客户端可以向Location报头里的URI发送GET请求，但它不是必须这么做。 |
| 304 | Not Modified | 这个响应代码跟204("No Content")类似：响应实体主体都必须为空。但204用于没有主体数据的情况，而304用于有主体数据，但客户端已拥有该数据，没必要重复发送的情况。 |
| 4XX：客户端错误  这些响应代码表明客户端出现错误。不是认证信息有问题，就是表示格式或HTTP库本身有问题。客户端需要自行改正。 | 400 | Bad Request | 这是一个通用的客户端错误状态，当其他4XX响应代码不适用时，就采用400。此响应代码通常用于“服务器收到客户端通过PUT或者POST请求提交的表示，表示的格式正确，但服务器不懂它什么意思”的情况。 |
| 401 | Unauthorized | 客户端试图对一个受保护的资源进行操作，却又没有提供正确的认证证书。客户端提供了错误的证书，或者根本没有提供证书。这里的证书（credential）可以是一个用户名/密码，也可以是一个API key，或者一个认证令牌。 |
| 403 | Forbidden | 客户端请求的结构正确，但是服务器不想处理它。这跟证书不正确的情况不同--若证书不正确，应该发送响应代码401。该响应代码常用于一个资源只允许在特定时间段内访问，或者允许特定IP地址的用户访问的情况。 |
| 404 | Not Found | 404表明服务器无法把客户端请求的URI转换为一个资源 |
| 405 | Method Not Allowd | 客户端试图使用一个本资源不支持的HTTP方法 |
| 5XX 服务端错误  这些响应代码表明服务器端出现错误。一般来说，这些代码意味着服务器处于不能执行客户端请求的状态，此时客户端应稍后重试。 | 500 | Internal Server Error | 这是一个通用的服务器错误响应。对于大多数web框架，如果在执行请求处理代码时遇到了异常，它们就发送此响应代码。 |
| 501 | Not Implemented | 客户端试图使用一个服务器不支持的HTTP特性。 |
| 502 | Bad Gateway | 只有HTTP代理会发送这个响应代码。它表明代理方面出现问题，或者代理与上行服务器之间出现问题，而不是上行服务器本身有问题。若代理根本无法访问上行服务器，响应代码将是504。 |
| 503 | Service Unavailable | 此响应代码表明HTTP服务器正常，只是下层web服务服务不能正常工作。最可能的原因是资源不足：服务器突然收到太多请求，以至于无法全部处理。由于此问题多半由客户端反复发送请求造成，因此HTTP服务器可以选择拒绝接受客户端请求而不是接受它，并发送503响应代码。 |

1． A类IP地址

一个A类IP地址由1字节的网络地址和3字节主机地址组成，网络地址的最高位必须是“0”， 地址范围从1.0.0.0 到126.0.0.0。可用的A类网络有126个，每个网络能容纳1亿多个主机。

2． B类IP地址

一个B类IP地址由2个字节的网络地址和2个字节的主机地址组成，网络地址的最高位必须是“10”，地址范围从128.0.0.0到191.255.255.255。可用的B类网络有16382个，每个网络能容纳6万多个主机 。

3． C类IP地址

一个C类IP地址由3字节的网络地址和1字节的主机地址组成，网络地址的最高位必须是“110”。范围从192.0.0.0到223.255.255.255。C类网络可达209万余个，每个网络能容纳254个主机。

4． D类地址用于多点广播（Multicast）。

D类IP地址第一个字节以“lll0”开始，它是一个专门保留的地址。它并不指向特定的网络，目前这一类地址被用在多点广播（Multicast）中。多点广播地址用来一次寻址一组计算机，它标识共享同一协议的一组计算机。

5． E类IP地址

以“llll0”开始，为将来使用保留。

在IP地址3种主要类型里，各保留了3个区域作为私有地址，其地址范围如下：

A类地址：10.0.0.0～10.255.255.255

B类地址：172.16.0.0～172.31.255.255

C类地址：192.168.0.0～192.168.255.255

另外IPv6地址长度为128位，是IETF（互联网工程任务组，Internet Engineering Task Force）设计的用于替代现行版本IP协议（IPv4）的下一代IP协议。