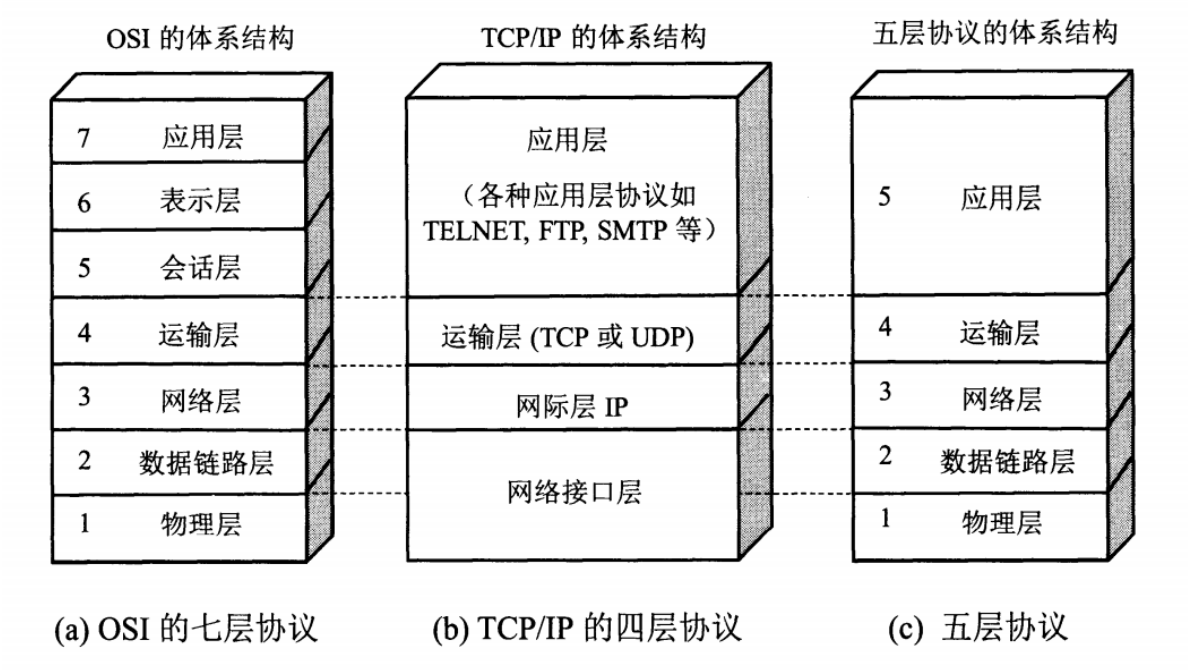
计算机网络

网络体系结构

结构图



各层作用及协议

分层	作用	协议
物理层	通过媒介传输比特，确定机械及电气规范（比特 Bit）	RJ45、CLOCK、IEEE802.3（中继器，集线器）
数据链路层	将比特组装成帧和点到点的传递（帧 Frame）	PPP、FR、HDLC、VLAN、MAC（网桥，交换机）
网络层	负责数据包从源到宿的传递和网际互连（包 Packet）	IP、ICMP、ARP、RARP、OSPF、IPX、RIP、IGRP（路由器）
运输层	提供端到端的可靠报文传递和错误恢复（ 段Segment）	TCP、UDP、SPX
应用层		HTTP DNS DHCP

应用层

常用端口

应用	应用层协议	端口号	传输层协议	备注
域名解析	DNS	53	UDP/TCP	长度超过 512 字节时使用 TCP
动态主机配置协议	DHCP	67/68	UDP	
简单网络管理协议	SNMP	161/162	UDP	
文件传送协议	FTP	20/21	TCP	控制连接 21，数据连接 20
远程终端协议	TELNET	23	TCP	
超文本传送协议	HTTP	80	TCP	
简单邮件传送协议	SMTP	25	TCP	
邮件读取协议	POP3	110	TCP	
网际报文存取协议	IMAP	143	TCP	

页面请求过程

1. DHCP 配置主机信息

- 假设主机最开始没有 IP 地址以及其它信息，那么就需要先使用 DHCP 来获取。
- 主机生成一个 DHCP 请求报文，并将这个报文放入具有目的端口 67 和源端口 68 的 UDP 报文段中。
- 该报文段则被放入在一个具有广播 IP 目的地址(255.255.255.255) 和源 IP 地址 (0.0.0.0) 的 IP 数据报中。
- 该数据报则被放置在 MAC 帧中，该帧具有目的地址 FF:FF:FF:FF:FF:FF，将广播到与交换机连接的所有设备。
- 连接在交换机的 DHCP 服务器收到广播帧之后，不断地向上分解得到 IP 数据报、UDP 报文段、DHCP 请求报文，之后生成 DHCP ACK 报文，该报文包含以下信息：IP 地址、DNS 服务器的 IP 地址、默认网关路由器的 IP 地址和子网掩码。该报文被放入 UDP 报文段中，UDP 报文段有被放入 IP 数据报中，最后放入 MAC 帧中。
- 该帧的目的地址是请求主机的 MAC 地址，因为交换机具有自学习能力，之前主机发送了广播帧之后就记录了 MAC 地址到其转发接口的交换表项，因此现在交换机就可以直接知道应该向哪个接口发送该帧。
- 主机收到该帧后，不断分解得到 DHCP 报文。之后就配置它的 IP 地址、子网掩码和 DNS 服务器的 IP 地址，并在其 IP 转发表中安装默认网关。

2. ARP 解析 MAC 地址

- 主机通过浏览器生成一个 TCP 套接字，套接字向 HTTP 服务器发送 HTTP 请求。为了生成该套接字，主机需要知道网站的域名对应的 IP 地址。
- 主机生成一个 DNS 查询报文，该报文具有 53 号端口，因为 DNS 服务器的端口号是 53。
- 该 DNS 查询报文被放入目的地址为 DNS 服务器 IP 地址的 IP 数据报中。
- 该 IP 数据报被放入一个以太网帧中，该帧将发送到网关路由器。

- DHCP 过程只知道网关路由器的 IP 地址，为了获取网关路由器的 MAC 地址，需要使用 ARP 协议。
- 主机生成一个包含目的地址为网关路由器 IP 地址的 ARP 查询报文，将该 ARP 查询报文放入一个具有广播目的地址（FF:FF:FF:FF:FF:FF）的以太网帧中，并向交换机发送该以太网帧，交换机将该帧转发给所有的连接设备，包括网关路由器。
- 网关路由器接收到该帧后，不断向上分解得到 ARP 报文，发现其中的 IP 地址与其接口的 IP 地址匹配，因此就发送一个 ARP 回答报文，包含了它的 MAC 地址，发回给主机。

3．DNS 解析域名

- 知道了网关路由器的 MAC 地址之后，就可以继续 DNS 的解析过程了。
- 网关路由器接收到包含 DNS 查询报文的以太网帧后，抽取出 IP 数据报，并根据转发表决定该 IP 数据报应该转发的路由器。
- 因为路由器具有内部网关协议（RIP、OSPF）和外部网关协议（BGP）这两种路由选择协议，因此路由表中已经配置了网关路由器到达 DNS 服务器的路由表项。
- 到达 DNS 服务器之后，DNS 服务器抽取出 DNS 查询报文，并在 DNS 数据库中查找待解析的域名。
- 找到 DNS 记录之后，发送 DNS 回答报文，将该回答报文放入 UDP 报文段中，然后放入 IP 数据报中，通过路由器反向转发回网关路由器，并经过以太网交换机到达主机。

4．HTTP 请求页面

- 有了 HTTP 服务器的 IP 地址之后，主机就能够生成 TCP 套接字，该套接字将用于向 Web 服务器发送 HTTP GET 报文。
- 在生成 TCP 套接字之前，必须先与 HTTP 服务器进行三次握手来建立连接。生成一个具有目的端口 80 的 TCP SYN 报文段，并向 HTTP 服务器发送该报文段。
- HTTP 服务器收到该报文段之后，生成 TCP SYN ACK 报文段，发回给主机。
- 连接建立之后，浏览器生成 HTTP GET 报文，并交付给 HTTP 服务器。
- HTTP 服务器从 TCP 套接字读取 HTTP GET 报文，生成一个 HTTP 响应报文，将 Web 页面内容放入报文主体中，发回给主机。
- 浏览器收到 HTTP 响应报文后，抽取出 Web 页面内容，之后进行渲染，显示 Web 页面。

HTTP

HTTP协议工作在应用层，端口号是80。HTTP协议被用于网络中两台计算机间的通信，相比于TCP/IP这些底层协议，HTTP协议更像是高层标记型语言，浏览器根据从服务器得到的HTTP响应体中分别得到报文头，响应头和信息体（HTML正文等），之后将HTML文件解析并呈现在浏览器上。同样，我们在浏览器地址栏输入网址之后，浏览器相当于用户代理帮助我们组织好报文头，请求头和信息体（可选），之后通过网络发送到服务器，，服务器根据请求的内容准备数据。所以如果想要完全弄明白HTTP协议，你需要写一个浏览器 + 一个Web服务器，一侧来生成请求信息，一侧生成响应信息。

从网络分层模型来看，HTTP工作在应用层，其在传输层由TCP协议为其提供服务。所以可以猜到，HTTP请求前，客户机和服务器之间一定已经通过三次握手建立起连接，其中套接字中服务器一侧的端口号为HTTP周知端口80。在请求和传输数据时也是有讲究的，通常一个页面上不只有文本数据，有时会内嵌很多图片，这时候有两种选择可以考虑。一种是对每一个文件都建立一个TCP连接，传送完数据后立马断开，通过多次这样的操作获取引用的所有数据，但是这样一个页面的打开需要建立多次连接，效率会低很多。另一种是对于有多个资源的页面，传送完一个数据后不立即断开连接，在同一次连接下多次传输数据直至传完，但这种情况有可能会长时间占用服务器资源，降低吞吐率。上述两种模式分别是HTTP 1.0和HTTP 1.1版本的默认方式，具体是什么含义会在后面详细解释。

HTTP协议结构

请求报文

对于HTTP请求报文我们可以通过以下两种方式比较直观的看到：一是在浏览器调试模式下（F12）看请求响应信息，二是通过wireshark或者tcpdump抓包实现。通过前者看到的数据更加清晰直观，通过后者抓到的数据更真实。但无论是用哪种方式查看，得到的请求报文主题体信息都是相同的，对于请求报文，主要包含以下四个部分，每一行数据必须通过"\r\n"分割，这里可以理解为行末标识符。

- 报文头（只有一行）

  结构：method uri version

  - method

    HTTP的请求方法，一共有9中，但GET和POST占了99%以上的使用频次。GET表示向特定资源发起请求，当然也能提交部分数据，不过提交的数据以明文方式出现在URL中。POST通常用于向指定资源提交数据进行处理，提交的数据被包含在请求体中，相对而言比较安全些。

  - uri

    用来指代请求的文件，≠URL。

  - version

    HTTP协议的版本，该字段有HTTP/1.0和HTTP/1.1两种。

- 请求头（多行）

  在HTTP/1.1中，请求头除了Host都是可选的。包含的头五花八门，这里只介绍部分。

  - Host：指定请求资源的主机和端口号。端口号默认80。
  - Connection：值为keep-alive和close。keep-alive使客户端到服务器的连接持续有效，不需要每次重连，此功能为HTTP/1.1预设功能。
  - Accept：浏览器可接收的MIME类型。假设为text/html表示接收服务器回发的数据类型为text/html，如果服务器无法返回这种类型，返回406错误。
  - Cache-control：缓存控制，Public内容可以被任何缓存所缓存，Private内容只能被缓存到私有缓存，non-cache指所有内容都不会被缓存。
  - Cookie：将存储在本地的Cookie值发送给服务器，实现无状态的HTTP协议的会话跟踪。
  - Content-Length：请求消息正文长度。

  另有User-Agent、Accept-Encoding、Accept-Language、Accept-Charset、Content-Type等请求头这里不一一罗列。由此可见，请求报文是告知服务器请求的内容，而请求头是为了提供服务器一些关于客户机浏览器的基本信息，包括编码、是否缓存等。

- 空行（一行）
- 可选消息体（多行）

响应报文

响应报文是服务器对请求资源的响应，通过上面提到的方式同样可以看到，同样地，数据也是以"\r\n"来分割。

- 报文头（一行）

  结构：version status_code status_message

  - version

    描述所遵循的HTTP版本。

  - status_code

    状态码，指明对请求处理的状态，常见的如下。

    - 200：成功。
    - 301：内容已经移动。
    - 400：请求不能被服务器理解。
    - 403：无权访问该文件。
    - 404：不能找到请求文件。

        - 500：服务器内部错误。
        - 501：服务器不支持请求的方法。
        - 505：服务器不支持请求的版本。
    - status_message

      显示和状态码等价英文描述。

- 响应头（多行）

  这里只罗列部分。

    - Date：表示信息发送的时间。
    - Server：Web服务器用来处理请求的软件信息。
    - Content-Encoding：Web服务器表明了自己用什么压缩方法压缩对象。
    - Content-Length：服务器告知浏览器自己响应的对象长度。
    - Content-Type：告知浏览器响应对象类型。
- 空行（一行）
- 信息体（多行）

  实际有效数据，通常是HTML格式的文件，该文件被浏览器获取到之后解析呈现在浏览器中。

会话机制

HTTP作为无状态协议，必然需要在某种方式保持连接状态。这里简要介绍一下Cookie和Session。

Cookie

Cookie是**客户端**保持状态的方法。

Cookie简单的理解就是存储由服务器发至客户端并由客户端保存的一段字符串。为了保持会话，服务器可以在响应客户端请求时将Cookie字符串放在Set-Cookie下，客户机收到Cookie之后保存这段字符串，之后再请求时候带上Cookie就可以被识别。

除了上面提到的这些，Cookie在客户端的保存形式可以有两种，一种是会话Cookie一种是持久Cookie，会话Cookie就是将服务器返回的Cookie字符串保持在内存中，关闭浏览器之后自动销毁，持久Cookie则是存储在客户端磁盘上，其有效时间在服务器响应头中被指定，在有效期内，客户端再次请求服务器时都可以直接从本地取出。需要说明的是，存储在磁盘中的Cookie是可以被多个浏览器代理所共享的。

Session

Session是**服务器**保持状态的方法。

首先需要明确的是，Session保存在服务器上，可以保存在数据库、文件或内存中，每个用户有独立的Session用户在客户端上记录用户的操作。我们可以理解为每个用户有一个独一无二的Session ID作为Session文件的Hash键，通过这个值可以锁定具体的Session结构的数据，这个Session结构中存储了用户操作行为。

综合

当服务器需要识别客户端时就需要结合Cookie了。每次HTTP请求的时候，客户端都会发送相应的Cookie信息到服务端。实际上大多数的应用都是用Cookie来实现Session跟踪的，第一次创建Session的时候，服务端会在HTTP协议中告诉客户端，需要在Cookie里面记录一个Session ID，以后每次请求把这个会话ID发送到服务器，我就知道你是谁了。如果客户端的浏览器禁用了Cookie，会使用一种叫做URL重写的技术来进行会话跟踪，即每次HTTP交互，URL后面都会被附加上一个诸如sid=xxxxx这样的参数，服务端据此来识别用户，这样就可以帮用户完成诸如用户名等信息自动填入的操作了。