网 络 新 技 术

Network New Technologies

实用双语教程

贺思德 申浩如 李海燕 编著

云南大学出版社

内 容 简 介

本教材采用英汉注解双语教学的形式，介绍互联网发展的一些最新应用技术。参考了美国、加拿大和欧洲等著名大学的有关教学大纲，内容包括互联网应用协议及网络协议分析工具，xDSL 数字用户环路，移动通信和 CDMA 的技术原理，计算机外设接口通用串行总线 USB 的工作原理，对等网络与无线局域网的应用，计算机网络的维护与测试方法，高清晰电视、IPTV 和 MP3 的基本知识，网络搜索引擎 Google 的关键技术，博客的理念与技术，计算机网络病毒的原理及防范方法，防火墙的种类和网络安全的配置方法。

教材内容均为当前互联网技术发展的应用热点问题，对复杂技术的介绍基本概念清晰易懂，深入浅出，知识面较广，强调学用结合，学员可根据课程内容进行网络技术实验。可供研究生、高年级本科生和网络工程技术管理人员等作为英汉双语专业教材和参考读物。

前　　言

国家教育部于 1999 年颁布的《大学英语教学大纲》（修订本）规定："学生在完成基础阶段的学习任务，达到四级或六级后，都必须修读专业英语。"这说明学生对英语的学习应当与专业知识的学习结合起来，才能达到学以致用和提高专业教学质量的目的。教育部在 2001 年颁发了《关于加强高等学校本科教学工作提高教学质量的若干意见》明确提出："本科教育要创造条件使用英语等外语进行公共课和专业课教学。高新技术领域的生物技术、信息技术等专业，更要先行一步，力争在三年内，外语教学课程达到所开课程的 5%～10%。"因而高校开设了专业英语课程以及专业课的双语教学。这两类课程的教学目的和方法有所不同，前者是学习与专业相关的英语，着重于有关专业的英语词汇、语法和文化背景等，而后者是采用英语和英文教材进行专业课学习，着重于专业知识方面的系统性和实效性。在学习这两门课程之前学生应当已经掌握了大学英语和初步的专业基础知识，因此对某些专业课程可以将这两类课程的功能合一，以充分利用有限的教学课时。

高校专业课程双语教学的目的在于外语与专业知识的全面提高，使学生真正领会与世界同步的现代科学知识，又为学生提供用英语进行学术交流和学习专业外语词汇的机会。采用国外优秀的原版专业教材与采用国内教材相比，可以使学生接触到该专业方向的当前国际上的先进理念和先进技术，同时原版英文书写的文章使学生更为直接、准确地理解课程内容。可避免因翻译出版国外教材导致的时间滞后，以及翻译中出现的中文表达贻误等不利因素。很多新的科技专业英语词汇如何用中文准确表达，这需要译者具有丰富的专业背景知识并反复推敲，个别首次出现的专业词汇的汉语表达，甚至需要一段时间之后在业内才能形成共识。因此阅读科技原著是与国际科技发展接轨和同步的重要方式。

在专业英文教材的选用和引进过程中，存在几个不容忽视的差异：不同的国家，教育体制不同，各阶段的教育目标也各异，因而在内容上、深度上不可避免地存在差异。对于理工科，由于各国的工程标准、技术规范不尽相同，会影响学生在以后的工作和学习中对专业知识的正确运用。例如：美国的广播电视体制为 NTSC 制、无线电频率规范不同、长度重量等计量单位为英制等，很多标准都与我国的体制不同。如果学生的专业知识学习使用英制计量单位的原版专业教材，而今后工作时要使用公制的计量单位，这就会造成概念的混乱。美国有一次火星登陆飞船项目的失败，其原因就是在控制软件的系统集成时，其中一个模块的计量单位"公里"被误认为"英里"。因此至少需要在技术规范方面对原版教材进行删改或补充说明，有时很难找到与国内专业教学大纲相对应的合适的原版教材。国外很少有统一的教材，一本教材在很大程度上反映的只是某个人或某些人对该学科的认识与看法，特色鲜明，但不能完全适应当前我国教育模式的需要。因而采用原版教材并不能完全解决国内理工科专业课程

的双语教学问题。本教材就是基于这样的情况，在编者近年来对高年级本科生和研究生的课程讲义基础上编写的，它可以作为计算机网络技术的专业课双语教材或专业英语教材，也可供工程技术人员参考。

编者多年从事网络通信与计算机领域的本科生和研究生的专业课以及专业英语教学工作，使用过不少英文原版教材，曾在国外大学研究进修多年，具有丰富的工程实践经验，深切体会到办好专业课程的双语教学，在提高学生的专业素质和掌握新知识方面是可以大有作为的。本教材的重点在于让学生建立起对网络系统的清晰概念和对最新技术的掌握。部分内容介绍了如何对互联网和局域网进行测试、故障分析、维护管理、数据包的捕获与协议分析等，学生可以按照课文的介绍进行网络技术实验，解决网络应用中的技术问题。所介绍的网络协议分析软件 Ethereal 等可从互联网下载。

本教材的编写着重于新颖性（介绍网络的最新技术）、趣味性（讨论实际应用中的技术问题）、广泛性（系统的总体概念，专业知识面广）、实践性（学生可以动手进行网络实验和网络协议分析）。本书分为十个单元，第一单元通过分析几个简单的互联网应用实例（电子邮件、Web 浏览、域名查询）的工作过程，具体介绍了互联网中客户机、服务器、网关、远程访问和 Web 是如何进行工作的，并介绍了网络协议分析软件工具和测试的实例。第二单元介绍了宽带接入网中的 xDSL 的基本原理、GSM 移动通信和 CDMA 技术。第三单元介绍了日益普及的 USB 通用串行总线的工作原理。第四单元介绍了 IPv4 与 IPv6 的对比，无线局域网以及对等网络的概念。第五单元介绍了进行网络维护的诊断、包捕获与协议分析的几个有用的软件工具。第六单元介绍了当前广泛应用的 J2EE 与.NET 的分析对比，以及数据库的基本知识。第七单元介绍了多媒体通信中的常用协议、MP3、HDTV 和 IPTV（网络电视）。第八单元介绍了 Google 搜索引擎和 Blogging （博客）的工作原理。第九单元介绍了恶意软件类型和病毒对抗的工作原理。第十单元介绍了防火墙的种类和网络系统安全配置原理。

本书在版式安排上采用旁注生词、专业词组解释和难句翻译的方式，每篇课文附有简短的中文内容提要，这有利于方便读者能更好地理解课文内容和提高学习兴趣。对课文中的某些背景知识介绍，则放到课文的后面。每单元附有习题，供学员加深对课文知识的理解。教师可以根据需要选择部分内容作为重点讲解，而其余课文供学员课外阅读。教材另配有中英对照参考译文和详细的技术背景知识介绍，供教师和学员参考。

编著者

sdhe@ynu.edu.cn

2006 年 2 月 5 日

目　　录

UNIT 1 Internet Protocols & Analyzer

互联网协议及分析工具

SUMMARY（内容提要）

本单元详细介绍了 INTERNET 互联网的最常用业务的工作原理，以及网络协议的实验分析。包含三个方面的内容：

一、 **网络通信系统的业务与分层结构**；

二、 **TCP/IP 网络协议及分析工具**；

三、 **应用层协议分析和 TCP/IP 常用工具**。

第一节：网络通信系统的业务与分层结构。一个十分复杂的网络通信系统可以分解成由一系列功能较为单一的模块或层来组成。课文首先介绍大家熟悉的互联网的最常用业务－－Web 浏览、域名查询和电子邮件的详细工作过程，由此来说明通信双方系统中的各对等层协议之间是如何协调工作的，以及该层如何利用下层协议所提供的服务。协议、业务和分层的例子：HTTP 网页浏览，DNS 域名查询和 SMTP 电子邮件；传输层的服务 TCP 和 UDP，对等网络的文件共享，OSI 开放系统互联参考模型，单一路由进程的包交换网络，异构网络的互联模型，TCP/IP 网络的数据报构成。

第二节：TCP/IP 网络协议及分析工具。本节**内容包括：**1、TCP/IP 的网络结构；2、TCP/IP 各层之间是如何工作的，以分析一个简单的互联网络的实例来说明网络设备的 IP 地址和物理地址的作用，网络通信的双方如何发送和接收 IP 数据报，路由选择的概念；3、物理层、互联网层、传输层和应用层之间是如何协调工作的；4、如何使用网络协议分析工具 Ethereal 来捕获与分析网络上的数据报，由此可以直观地理解基于 TCP/IP 协议的客户机/服务器之间的通信工作过程。

第三节：应用层协议和 TCP/IP 常用工具。1、远程登录 Telnet 协议，用户可以通过基于字符的网络虚拟终端来远程登录服务器；2、文件传输协议 FTP，它可以在不同的网络操作系统和不同的主机之间高效地传输文件，FTP 支持的文件类型有：ASCII、EBCDIC、图像数据和字节数据流；3、超文本传输协议 HTTP 和网维网，它定义了客户机如何通过 Web 浏览器访问和获取 Web 服务器的信息，详细分析了 HTTP 数据包的结构，介绍了 HTTP 代理服务器和高速缓存的应用，Cookies 和 Web 会话的应用；4、简要介绍了 IP 常用工具：PING，TRACEROUTE，IPCONFIG，NETSTAT，TCPDUMP 等等。

OBJECTIVES（学习目标）

- 了解互联网常用业务：Web 浏览、电子邮件、域名查询和 FTP 文件传输的工作原理；
- 互联网的局域网和 ppp 拨号网络的客户机、服务器和路由器的工作方式，它们之间如何利用物理地址、IP 地址和域名进行路由寻址通信，套接字和端口号的原理；
- 利用网络协议分析器 Ethereal 进行网络实验，数据包的捕获与内容协议分析。

UNIT 1 Internet Protocols & Analyzer

（互联网协议及分析工具）

Section A

Network Applications and Layered Architectures

（网络应用与分层结构）

be called upon	被用于
critical function [ˈkritikəl ˈfʌŋkʃən]	重要的功能
transfer of funds [trænsˈfəː]	资金转移
coherent [kəuˈhiərənt] adj.	一致的，协调的
proprietary network architectures [prəˈpraiətəri] [ˈɑːkitektʃə]	具有自主知识产权的网络构架
routing and forwarding [ˈruːtiŋ]	路由与转发
hops in a network	网络的跳段

Communication networks can be ***called upon*** to support an extremely wide range of services. We routinely use networks to talk to people, to send e-mail, to transfer files, and to retrieve information. Business and industry use networks to carry out ***critical functions***, such as the ***transfer of funds***, and the automated processing of transactions, to query or update database information. Increasingly, the Internet is also being used to provide "broadcast" services along the lines of traditional radio and television. It is clear then that the network must be designed so that it has the flexibility to provide support for current services and to accommodate future services. To achieve this flexibility, an overall network architecture or plan is necessary.

The overall process of enabling two or more devices to communicate effectively across a network is extremely complex. Of course, we've identified the many elements of a network that are required to enable effective communication. Early network designers recognized the need to develop architectures that would provide a structure to organize these functions into a ***coherent*** form. As a result, in the early 1970s various computer companies developed ***proprietary network architectures***. A common feature to all of these was the grouping of the communication functions into related and manageable sets called layers. We saw that communication functions can be grouped according to the following tasks:

- The transport across a network of data from a process in one machine to the ***process*** at another machine.
- The ***routing and forwarding*** of packets across multiple ***hops in a network***.
- The transfer of a frame of data from one physical interface to another.

These layers of functions build on top of each other to enable communications. We use the term network architecture to refer to a set of protocols that specify how every layer is to function.

The ***decomposition*** of the overall communications problem into a set of layers is a first step to simplifying the design of the overall network. In addition the interaction between layers needs to be defined precisely. This is done through the definition of the service provided by each layer to the layer above, and through the definition of the interface between layers through which a service is requested and through which results are conveyed. A clearly defined service and interface allows a layer to ***invoke*** a service from the layer below ***without regard to*** how the service is implemented by any of the layers below. ***As long as*** the service is provided as specified, the implementation of the underlying layers can be changed. Also, new services that build on existing services can be introduced at any time, and in turn enable other new services at layers above. This provides flexibility in modifying and evolving the network. In contrast, a ***monolithic*** network design that uses a single large body of hardware and software to meet all the network requirements can quickly become ***obsolete*** and also is extremely difficult and expensive to modify. The layered approach accommodates ***incremental*** changes much more readily.

In this section we develop the notion of a layered architecture, and we provide examples from TCP/IP, the most important current network architecture. The discussion is organized as follows:

1. Web-browsing and e-mail applications are used to demonstrate the operation of a protocol within a layer and how it makes use of the communication services of the layer below. We introduce the HTTP, DNS, and SMTP application layer protocols in these examples.

2. The ***Open Systems Interconnection (OSI) reference model*** is discussed to show how the overall communication process can be organized into functions that are carried out in seven layers.

3. The TCP/IP architecture is introduced and compared to the OSI reference model. We present a detailed end-to-end example in a typical TCP/IP Internet. We use a network protocol analyzer to show the exchange of messages and packets in real networks. This section is key to seeing the big picture because it shows how all the layers work together.

Two optional sections present material that is useful in developing lab exercises and experiments involving TCP/IP:

4. We introduce Berkeley ***sockets,*** which allow the student to write applications that use the services provided by the TCP/IP protocols. We develop example programs that show the use of UDP and TCP sockets.

5. We introduce several important TCP/IP application layer protocols: Telnet, FTP, and HTTP. We also introduce several ***utilities*** and a ***network protocol analyzer*** that can be used as tools to study the operation of the Internet.

1. Examples of Protocols, Services and Layering

A protocol is a set of rules that ***governs*** how two or more communicating parties are to interact. When dealing with networks we run into a ***multiplicity*** of protocols, such as HTTP, FTP, and TCP. The purpose of a protocol is to provide some type of communication service. For example, the HTTP protocol enables the ***retrieval*** of web pages, and the TCP protocol

decomposition [ˌdi:kɔmpəˈziʃən]	**n. 分解**
invoke [inˈvəuk]	**v. 调用**
without regard to	**不考虑**
as long as	**只要**
monolithic [ˌmɔnəˈliθik]	**n. 单块（层）**
obsolete [ˈɔbsəli:t]	**adj. 陈旧的**
incremental [inkriˈmentəl]	**adj. 增加的**
Open Systems Interconnection reference model	**开放系统互联参考模型 OSI**
sockets [ˈsɔkit]	**套接字**
utilities	**工具，设施**
network protocol analyzer [ˈprəutəkɔl][ˈænəlaizə]	**网络协议分析器**
multiplicity	**n. 多样性**

enables the reliable transfer of streams of information between computers. In this chapter, we will see that the overall communications process can be organized into ***a stack of layers***. Each layer carries out a specific set of communication functions using its own protocol, and each layer builds on the services of the layer below it.

This section uses ***concrete*** examples to illustrate what is meant by a protocol and to show how two ***adjacent*** layers interact. Together the examples also show the advantages of layering. The examples use two familiar applications, namely, e-mail and Web browsing. We present a simplified discussion of the associated protocols.

1.1 HTTP, DNS, and SMTP

All the examples discussed in this section involve a ***client/server*** application. A server process in a computer waits for incoming requests by ***listening to a port***. A port is an address that identifies which process is to receive a message that is delivered to a given machine. Widely used applications have well-known port numbers assigned to their servers, so that client processes in other computers can readily make requests as required. The servers provide responses to those requests. The server software usually runs in the background and is referred to as a ***daemon***. For example, ***httpd*** refers to the server daemon for HTTP.

● Example—HTTP and Web Browsing

Let us consider an example of browsing through the World Wide Web (WWW). The WWW consists of a framework for accessing documents that are located in computers connected to the Internet. These documents are prepared using the Hyper Text Markup Language (HTML) and may consist of text, graphics, and other media and are interconnected by links that appear within the documents. The WWW is accessed through a browser program that displays the documents and allows the user to access other documents by clicking one of these links. Each link provides the browser with a ***uniform resource locator (URL)*** that specifies the name of the machine where the document is located as well as the name of the file that contains the requested document.

The ***Hyper Text Transfer Protocol (HTTP)*** specifies rules by which the client and server interact so as to retrieve a document. The rules also specify how the request and response are phrased. The protocol assumes that the client and server can exchange messages directly. In general, the client software needs to set up a two-way connection prior to the HTTP request.

Figure 1 and Table 1 show ***the sequence of events*** and messages that are involved in retrieving a document. In step 1 a user selects a document by clicking on its corresponding link. For example, the browser may extract the URL associated with the following link:

http://www.comm.utoronto.ca/comm.html

retrieval [ri'tri:vəl] **n. 取回；回传**

a stack of layers 层的堆叠

concrete ['kɔnkri:t] **adj. 具体的**

adjacent [ə'dʒeisənt] **adj. 邻近的**

client/server 客户机/服务器

listening to a port 倾听端口

daemon ['di:mən] **n. Internet 中的后台程序**

httpd n. 万维网服务器软件

uniform resource locator (URL)统一资源定位符

Hyper Text Transfer Protocol (HTTP) 超文本传输协议

the sequence of events 事件的顺序

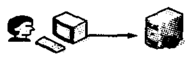
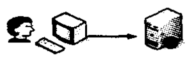
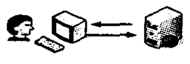
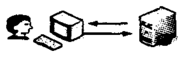
Step 1		The user clicks on a link to indicate which document is to be retrieved. The browser must determine the Internet address of the machine that contains the document. To do so, the browser sends a query to its local name server.
Step 2		Once the address is known, the browser establishes a connection to the server process in the specified machine, usually a TCP connection. For the connection to be successful, the specified machine must be ready to accept TCP connections.
Step 3		The browser runs a client version of HTTP, which issues a request specifying both the name of the document and the possible document formats it can handle.
Step 4-6		The machine that contains the requested document runs a server version of HTTP. It reacts to the HTTP request by sending an HTTP response which contains the desired document in the appropriate format.
Step 7-8		The user may start to view the document. The TCP connection is closed after a certain timeout period.

Figure 1 Retrieving a document from the web

Step	Event	Message Content
1	User selects document.	
2	Network software of client locates the server host and establishes a two-way connection.	
3	HTTP client sends message requesting document.	GET /comm.html HTTP/1.1
4	HTTP daemon listening on TCP port 80 interprets message.	
5	HTTP daemon sends a result code and a description of the information that the client will receive.	HTTP /1.1 200 OK Date: Mon, 06 Jan 2003 23:56:44 GMT Server: Apache/1.3.23 (Unix) Last Modified: 03 Sep 2002 02:58:36 GMT Content-Length: 8218 Content-Type: text/html <html>
6	HTTP daemon reads the file and sends requested file through the TCP port.	<head><title></title> ... <font face="Arial" >What is Communications?</font>
7	Text is displayed by client browser, which interprets the HTML format.	
8	HTTP daemon disconnects the connection after the connection is idle for some timeout period.	

Table 1 Retrieving a document from the web: HTTP message exchange

Domain Name System query [dəu'mein] ['kwiəri]域名系统查询	ephemeral port number ['femərəl] 临时端口号
duration n. 持续时间	server daemon 服务器后台程序
idle ['aidl] adj. 空闲的	timeout period 规定的时限
two peer processes [piə]两个对等的进程	two-way connection 双向连接

①因此 HTTP 客户机与服务器之间的信息传输路径实际上是虚连接，间接地通过 TCP 进行传输。

The client software must usually carry out a ***Domain Name System (DNS) query*** to determine the IP address corresponding to the host name, www.comm.utoronto.ca. (We discuss how this query is done in the next example.) The client software then sets up a TCP connection with the WWW server (the default is port 80) at the given IP address (step 2). The client end identifies itself by an ***ephemeral port number*** that is used only for the ***duration*** of the connection. The TCP protocol provides a reliable byte-stream transfer service that can be used to transmit files across the Internet.

After the connection is established, the client uses HTTP to request a document (step 3). The request message specifies the method or command (GET), the document (comm.html), and the protocol version that the browser is using (HTTP/1.1). The ***server daemon*** identifies the three components of the message and attempts to locate the file (step 4).

In step 5 the daemon sends a status line and a description of the information that it will send. Result code 200 indicates that the client request was successful and that the document is to follow. The message also contains information about the server software, the length of the document (8218 bytes), and the content type of the document (text/html). The request was for an image, the type might be image/gif. If the request is not successful, the server sends a different result code, which usually indicates the type of failure, for example, 404 when a document is not found.

In step 6 the HTTP daemon sends the file over the TCP connection. In the meantime, the client receives the file and displays it (step 7). The server maintains the TCP connection open so it can accept additional requests from the client. The server closes the TCP connection, it remains ***idle*** for some ***timeout period*** (step 8).

The HTTP example clearly indicates that a protocol is solely concerned with the interaction between the ***two peer processes***, that is, the client and the server. The protocol assumes that the message exchange between peer processes occurs directly as shown in Figure 2.2.

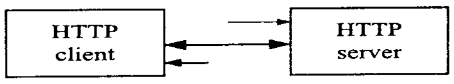


Figure 2 HTTP client/server interaction

Because the client and server machines are not usually connected directly, a connection needs to be set up between them. In the case of HTTP, we require a ***two-way connection*** that transfers a stream of bytes in correct sequential order and without errors. The TCP protocol provides this type of communication service between two processes in two machines connected to a network. Each HTTP process inserts its messages into a buffer, and TCP transmits the contents of the buffer to the other TCP in blocks of information called segments, as shown in Figure 3. Each segment contains port number information in addition to the HTTP message information. HTTP is said to use the service provided by TCP in the layer below. **Thus the transfer of messages between HTTP client and server in fact is virtual and occurs indirectly via the TCP connection**① as shown in Figure 3. Later you will see that TCP, in turn, uses the service provided by IP.

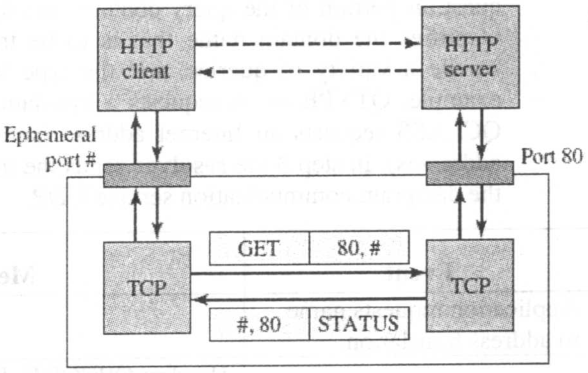


Figure 3 TCP provides a pipe between the HTTP client and HTTP server

socket system calls	**套接字系统的调用**
kernel n.	**内核**
DNS query	**域名服务器查询**
ISP	**互联网服务提供商**
resolve [ri'zɔlv] v.	**解析**

①应当注意HTTP应用协议如何调用了TCP所提供的服务。当HTTP客户软件首先需要建立TCP连接时，客户进行的是一系列的套接字系统的调用。这种调用类似于功能调用，只是当套接系统调用完成后，控制权移交给操作系统内核处理。套接系统调用定义了一序列确定的行动步骤，并包括一些参数，例如套接类型、TCP或UDP、以及地址信息。因此HTTP层与TCP层之间的互动是通过这些套接系统调用来实现的。

It is worth noting exactly how the HTTP application protocol invokes the service provided by TCP. When the HTTP client software first needs to set up the TCP connection, the client does so by making a series of *socket system calls*. These calls are similar to function calls except that control is passed to the operating system *kernel* when a socket system call is made. A socket system call specifies a certain action and may contain parameters such as socket type, for example, TCP or UDP, and address information. Thus the interaction between the HTTP layer and the TCP layer takes place through these socket system calls.①

● Example—DNS Query

The HTTP example notes that the client first needs to perform a ***DNS query*** to obtain the IP address corresponding to the domain name. This step is done by sending a message to a DNS server. The Domain Name System (DNS) is a distributed database that resides in multiple machines on the Internet and is used to convert between names and addresses and to provide e-mail routing information. Each DNS machine maintains its own database and acts as a DNS server that other machines can query. Typically the requesting machine accesses a local name server, which, for example, may reside in a university department or at an ***ISP***. These local name servers are able to ***resolve*** frequently used domain names into the corresponding IP addresses by caching recent information. When unable to resolve a name, the local name server may sometimes send a query to a root name server, of which there are currently 13 distributed globally. When a root server is unable to determine an IP address, it sends a query to an authoritative name server. Every machine on the Internet is required to register with at least two authoritative name servers. If a given name server cannot resolve the domain name, the queried name server will refer to another name server, and this process continues until a name server that can resolve the domain name is found.

We now consider a simple case where the resolution takes place in the first server. Table 2 shows the basic steps required for this example. After receiving the address request, a process in the host, called the ***resolver***,

resolver [ri'zɔlvə] 域名解析器	composes the short message shown in step 2. The OPCODE value in the DNS message header indicates that the message is a standard query. The question portion of the query contains the following information: QNAME identifies the domain name that is to be translated. The DNS server can handle a variety of queries, and the type is specified by QTYPE. In the example, QTYPE = A requests a translation of a name to an IP address. QCLASS requests an Internet address (some name servers handle non-IP addresses). In step 3 the **resolver** sends the message to the local server using the datagram communication service UDP.

Step	Event	Message content
1	Application requests name to address translation.	
2	Resolver composes query message.	Header: OPCODE=SQUERY Question: QNAME=tes1a.comm.toronto.edu, QCLASS=IN, QTYPE=A
3	Resolver sends UDP datagram encapsulating the query message.	
4	DNS server looks up address and prepares response.	Header: OPCODE=SQUERY, RESPONSE, AA Question: QNAME= tes1a.comm.toronto.edu, QCLASS=IN, QTYPE=A Answer: tes1a.comm.toronto.edu. 86400 IN A 128.100.11.1
5	DNS sends UDP datagram encapsulating the response message.	

Table 2 DNS query and response

Time-to-Live field 数据包的生存期字段（TTL）

User Datagram Protocol ['prəutəkɔl] 用户数据报协议(UDP)

connection-less 无连接的

The short message returned by the server in step 4 has the Response and Authoritative Answer bits set in the header. This setting indicates that the response comes from an authority that manages the domain name. The question portion is identical to that of the query. The answer portion contains the domain name for which the address is provided. This portion is followed by the **Time-to-Live field**, which specifies the time in units of seconds that this information is to be cached by the client. Next are the two values for QCLASS and QTYPE. IN again indicates that it is an Internet address. Finally, the IP address of the domain name is given (128.100.11.1).

In this example the DNS query and response messages are transmitted by using the communication service provided by the ***User Datagram Protocol (UDP)***. The UDP client attaches a header to the user information to provide port information (port 53 for DNS) and encapsulates the resulting block in an IP packet. The UDP service is ***connectionless***; no connection setup is required, and the datagram can be sent immediately. Because DNS queries and responses consist of short messages, UDP is ideally suited for conveying them.

The DNS example shows again how a protocol, in this case the DNS query protocol, is solely concerned with the interaction between the client and server processes. The example also shows how the transfer of messages between client and server, in fact, is virtual and occurs indirectly via UDP datagrams.

Simple Mail Transfer Protocol (SMTP)	**简单邮件传输协议**
plain [plein]	**adj. 普通的**
thereafter [ðεər'ɑ:ftə]	**adv. 然后**

● Example—SMTP and E-mail

Finally, we consider an e-mail example, using the ***Simple Mail Transfer Protocol (SMTP)***. Here a mail client application interacts with a local SMTP Server to initiate the delivery of an e-mail message. The user prepares an e-mail message that includes the recipient's e-mail address, a subject line, and a body. When the user clicks Send, the mail application prepares a file with the above information and additional information specifying format, for example, ***plain*** ASCII or Multipurpose Internet Mail Extensions (MIME) to encode non-ASCII information. The mail application has the name of the local SMTP server and may issue a DNS query for the IP address. Table 3 shows the remaining steps involved in completing the transfer of the e-mail message to the local SMTP Server.

Before the e-mail message can be transferred, the application process must set up a TCP connection to the local SMTP server (step 1). ***Thereafter***, the SMTP protocol is used in a series of exchanges in which the client identifies itself, the sender of the e-mail, and the recipient (steps 2-8). The client then transfers the message that the SMTP Server accepts for delivery (steps 9-12) and ends the mail session. The local SMTP Server then repeats this process with the destination SMTP Server. To locate the destination SMTP server, the local Server may have to perform a DNS query of type MX (mail exchange). SMTP works best when the destination machine is always available. For this reason, users in a PC environment usually retrieve their e-mail from a mail server using the Post Office Protocol version 3 (POP3) instead.

Step	Event	Message content
1	The mail application establishes a TCP connection (port 25) to its local SMTP server.	
2	SMTP daemon issues the following message to the client, indicating that it is ready to receive mail.	220 tesla.comm.toronto.edu ESMTP Send mail 8.9.0/8.9.0; Thu, 2 Jul 1998 05:07:59 -0400 *(EDT)*
3	Client sends a HELO message and identifies itself.	HELO bhaskara.comm.utoronto.ca
4	SMTP daemon issues a 250 message, indicating the client may proceed.	250 tesla.comm.toronto.edu Hello bhaskara.comm [128.100.10.9], pleased to meet you
5	Client sends sender's address.	MAIL FROM: <banerjea@comm.utoronto.ca>
6	If successful, SMTP daemon replies with a 250 message.	250<banerjea@comm.utoronto.ca> .... Sender ok
7	Client sends recipient's address.	RCPT TO: <alg@nal.utoronto.ca>
8	A 250 message is returned.	250<alg@nal.utoronto.ca> Recipient ok
9	Client sends a DATA message requesting permission to send the mail message.	DATA

10	The daemon sends a message giving the client permission to send.	354 Enter mail, end with “ ” on a line by itself
11	Client sends the actual text.	Hi AI, This section on email sure needs a lot of work ...
12	Daemon indicates that the message is accepted for delivery. A message 10 is returned.	250 FAA00803 Message accepted for delivery
13	Client indicates that the mail session is over.	QUIT
14	Daemon confirms the end of the session.	221 tesla.comm.toronto.edu closing connection

Table 3 Sending e-mail

EDT 美国东部地区时间

1.2 TCP and UDP Transport Layer Services

The e-mail, DNS query, and HTTP examples show how multiple protocols can operate by using the communication services provided by the TCP and UDP protocols. Both the TCP and UDP protocols operate by using the connectionless packet network service provided by IP.

①UDP 协议简单且快速，但是在信息传递和顺序方面不提供保障。

UDP provides connectionless transfer of datagrams between processes in hosts attached to the Internet. UDP provides port numbering to identify the source and destination processes in each host. ***UDP is simple and fast but provides no guarantees in terms of delivery or sequence addressing***①.

congestion control[kən'dʒestʃən kən'trol] 拥塞控制

TCP provides for reliable transfer of a byte stream between processes in hosts attached to the Internet. The processes write bytes into a buffer for transfer across the Internet by TCP. TCP is considerably more complex than UDP. TCP involves the establishment of a connection between the two processes. To provide their service, the TCP entities implement error detection and retransmission as well as flow control algorithms. In addition, TCP also implements ***congestion control***, which regulates the flow of segments into the network. This topic is discussed later.

file-sharing 文件分享

Indeed, an entire suite of protocols has been developed to operate on top of TCP and UDP, thereby demonstrating the usefulness of the layering concept. New services can be quickly developed by building on the services provided by existing layer protocols.

peer-to-peer applications [æpli'keiʃən] 对等网络应用

- Peer-to-Peer File Sharing

transient ['trænziənt] adj. 短暂的

File-sharing applications such as Napster and Gnutella became extremely popular as a means of exchanging MP3 audio and other files. The essence of these ***peer-to-peer applications*** is that ordinary PCs (“peers”) attached to the Internet can act not only as clients, but also as ***transient*** file servers while the applications are activated. When a peer is interested in finding a certain file, it sends a query. The response provides a list of peers that have the file and additional information such as the speed of each peer's connection to the Internet. The requesting peer can then set up a TCP connection to one of the peers in the list and proceed to retrieve the file.

The technically difficult part in peer-to-peer file sharing is maintaining the database of peers that are connected at a given point in time and the files that they have available for sharing. The Napster approach used a centralized database that peers could contact when they became available for file sharing and/or when they needed to make a query. The Gnutella approach uses a distributed approach where the peers organize themselves into an ***overlay*** network by keeping track of peers that are assigned to be adjacent to them. A query from a given peer is then broadcast by sending the query to each neighbor, their neighbors' neighbors, and so on ***up to some maximum number of hops***.

Peer-to-peer file sharing provides another example of how new services and applications can be deployed very quickly over the Internet. Peer-to-peer file sharing also brings up many legal, commercial, and cultural issues that will require many years to resolve.

2. The OSI Reference Model

The early network architectures developed by various computer ***vendors*** were not compatible with each other. This situation had the effect of ***locking in customers with a single vendor***. As a result, there was pressure in the 1970s for an ***open systems architecture*** that would eventually lead to the design of computer network equipment that could communicate with each other. This desire led to an effort in the International Organization for Standardization (ISO) first to develop a reference model for open systems interconnection (OSI) and later to develop associated standard protocols. The OSI reference model ***partitioned*** the communications process into seven layers and provided a framework for talking about the overall communications process and hence was intended to facilitate the development of standards. The OSI work also provided a ***unified*** view of layers, protocols and services. This unified view has provided the basis for the development of networking standards to the present day.

The Seven-Layer OSI Reference Model

Consider an application that involves communications between a process in computer A and a process in computer B. The OSI reference model divides the basic communication functions required for computers A and B to communicate into the seven layers shown in Figure 4. In this section, we will discuss the functions of the seven layers starting from the bottom (physical layer) to the top (application layer).

The physical layer deals with the transfer of bits over a communication channel, for example, the digital transmission system and the transmission media such as ***copper wire pairs***, ***coaxial cable***, radio, or optical fiber. The layer is concerned with the particular choice of system parameters such as voltage levels and signal durations. The layer is also concerned with the procedures to set up and release the physical connection, as well as with mechanical aspects such as socket type and number of ***pins***. For example, an Ethernet physical layer standard defines the connector and signal interfaces in a PC.

overlay	**n. 覆盖**
up to some maximum number of hops	**直到某个最大的网络跳（段）数**
vendor ['vendɔː]	**制造商**
locking in customers with a single vendor	**将客户绑定在某个制造商上**
open systems architecture ['ɑːkitektʃə]	**开放的系统结构**
partition [pɑː'tiʃən]	**v. 划分，分割**
unified ['juːnifaid]	**adj. 统一的，规范的**
copper wire pairs ['kɔpə 'waiə pɛə]	**铜芯双绞线**
coaxial cable [kəu'æksəl 'keibl]	**同轴电缆**
pin	**n. 插脚**

Presentation layer	表示层
Session layer	会话层
Transport layer	传输层
Network Layer	网络层
Data link layer	数据链路层
Physical layer	物理层
be prone to [prəun]	易于，倾向于
coordinate [kəu'ɔ:dinit]	vt. 协调，整理
flat addressing space	平等的地址空间
hierarchical addressing scheme [ˌhaiə'rɑ:kikəl] [ski:m]	分层次的地址方案
network nodes [nəud]	网络节点
traverse ['trævə(:)s]	vt. 横过，穿过

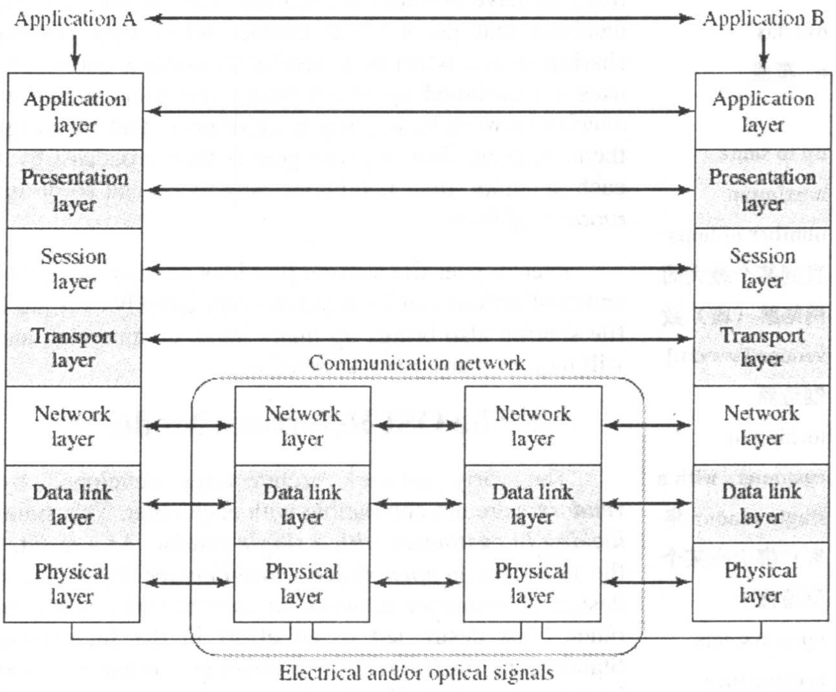


Figure 4 The seven-layer OSI reference model

The data link layer provides for the transfer of frames (blocks of information) across a transmission link that directly connects two nodes. The data link layer inserts framing information in the sequence of transmitted bits to indicate the boundaries of the frames. It also inserts control and address information in the header and check bits to enable recovery from transmission errors, as well as flow control. The data link control is particularly important when the transmission link ***is prone to*** transmission errors. Historically, the data link layer has included the case where multiple terminals are connected to a host computer in point-to-multipoint fashion.

The OSI data link layer was defined so that it included the functions of LANs, which are characterized by the use of broadcast transmissions. The notion of a "link," then, includes the case where multiple nodes are connected to a broadcast medium. As before, frames flow directly between nodes. A medium access control procedure is required to ***coordinate*** the transmissions from the machines into the medium. A ***flat addressing space*** is used to enable machines to listen and recognize frames that are destined to them. Later in this chapter we will discuss the Ethernet LAN standard.

The network layer provides for the transfer of data in the form of packets across a communication network. One key aspect of the network layer is the use of a ***hierarchical addressing scheme*** that identifies the point of attachment to the network and that can accommodate a large number of network users. A key aspect of the packet transfer service is the routing of the packets from the source machine to the destination machine, typically ***traversing*** a number of transmission links and ***network nodes*** where routing