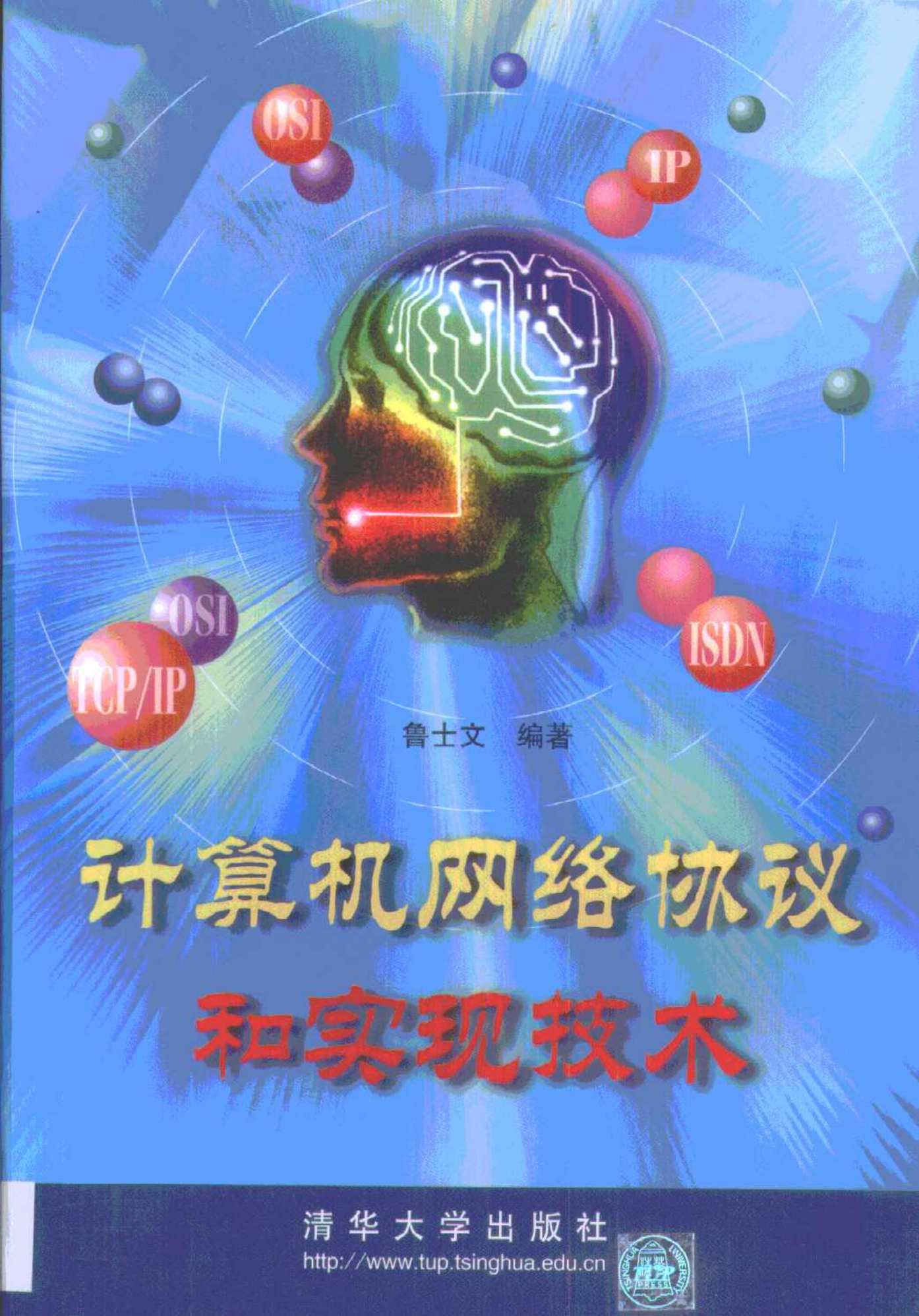


鲁士文 编著

计算机网络协议和实现技术

清华大学出版社
http://www.tup.tsinghua.edu.cn

计算机网络协议和实现技术

鲁士文 编著
抖斗书屋 审校

清华大学出版社

(京) 新登字 158 号

内 容 简 介

本书以计算机网络协议为核心，以流行的五层混合模型为主要线索，从低层到高层全面地论述了计算机网络原理、结构、标准和关键技术。全书共有 12 章，包括 OSI 参考模型和 TCP/IP 协议体系、数据信号传输和物理层协议、数据链路控制、局域网络和媒体访问协议、城域网和无线网、网络层运行机制和 X.25 公用网络、ISDN 和宽带网络、互联网和 IP 协议、端到端的传输、通信协议的描述验证和测试、面向应用的协议以及 TCP/IP 网络管理诸多内容。每一章都插有许多图表和示例，以供读者加深理解、深入研究。

本书可作为计算机科学、电子工程类以及相关专业的研究生和大学高年级的教学参考书，也可供从事计算机网络研究和开发的科技人员参考。

图书在版编目(CIP)数据

计算机网络协议和实现技术/鲁士文编著 .—北京：清华大学出版社，2000
ISBN 7-302-03920-8

Ⅰ.计… Ⅱ.鲁… Ⅲ.传输控制协议 Ⅳ.TN915.04

中国版本图书馆 CIP 数据核字 (2000) 第 30035 号

出版者：清华大学出版社(北京 清华大学学研楼，邮政编码：100084)
http://www.tup.tsinghua.edu.cn
印刷者：清华大学印刷厂
发行者：新华书店总店北京发行所
开 本：787×1092 1/16 印张：36.75 字数：871 千字
版 次：2000 年 7 月第 1 版 2000 年 7 月第 1 次印刷
书 号：ISBN 7-302-03920-8/TP·2290
印 数：0001～7000
定 价：42.00 元

第 1 章　OSI 参考模型和 TCP/IP 协议体系

计算机和通信的结合促进了现代社会的发展。在国家和国际范围内，分布在不同地区的计算机使用公共通信服务交换电子信息。在一个单位内，基于 PC 和工作站的分布式计算机使用局部通信网络访问昂贵的共享设备（如打印机、扫描仪、磁盘、磁带等）和企业内部的信息资源（通过数据库和 WWW 服务器等形式提供）。

从计算机与通信技术相结合的广义观点出发，我们可以把计算机网络定义为“计算技术与通信技术相结合，实现远程信息处理和进一步共享资源的系统”。照此定义，本世纪 50 年代的“远程终端-计算机网”，60 年代的“计算机-计算机网”以及目前发展的分布式计算机网均属于计算机网络。美国信息处理学会联合会（AFIPS）在 1970 年从共享资源角度出发，把计算机网络定义为“以能够相互共享资源（硬件、软件和数据等）的方式连接起来，并各自具备独立功能的计算机系统的集合”。随着“远程终端-计算机”通信发展到“计算机-计算机”通信，后来人们又提出了计算机通信网的定义：在计算机之间以传输信息为目的连接起来的计算机系统的集合，称为计算机通信网。

从物理结构上看，计算机网络又可定义为在协议控制下，由若干计算机、终端设备、数据传输设备和通信控制处理机等组成的系统集合。该定义强调计算机网是在协议控制下，通过通信系统实现计算机之间的连接，网络协议是区别计算机网络与一般的计算机互联系统的标志。

综上所述，根据目前流行的观点，我们在本书中将把计算机网络定义为：按照网络协议，以共享资源为主要目的，将地理上分散且独立的计算机互相连接起来形成的集合体。通常根据人们所处环境和研究的着眼点不同，可采用不同术语。当着重研究网络资源共享问题时，可称作计算机网络；当着重研究和分析通信方面问题时，常称作计算机通信网络。本书对这两个术语将不加严格区分，一般都称作计算机网络。

基本的通信硬件包括了在计算机之间传送位串序列的机制。但是，仅仅使用硬件来进行通信就好像用 0 和 1 二进制编程那样难以实现。为了方便网络程序设计，计算机通常都是连到使用复杂的软件的网络上。这些软件为应用程序提供了方便的高层接口，自动处理大多数低层的通信细节和问题。因此，大多数应用程序依靠网络软件通信，并不直接与网络硬件打交道。

网络中的通信是指在不同系统中的实体之间的通信。所谓实体，是指能发送和接收信息的终端、应用软件、通信进程等。实体之间通信需要一些规则和约定，例如，传送的信息块采用何种编码和怎样的格式？如何识别收发者的名称和地址？传送过程中出现错误如何处理？发送和接收速率不一致怎么办？简单地讲，通信双方在通信时需要遵循的一组规则和约定就是协议。协议主要由语义、语法和定时三部分组成，语义规定通信双方准备“讲什

么",亦即确定协议元素的种类;语法规定通信双方"如何讲",确定数据的信息格式、信号电平等;定时则包括速度匹配和排序等。

1.1 协议的分层结构

两个系统中实体间的通信是一个十分复杂的过程,为了减少协议设计和调试过程的复杂性,大多数网络的实现都按层次的方式来组织,每一层完成一定的功能,每一层又都建立在它的下层之上。不同的网络,其层的数量,各层的名字、内容和功能不尽相同,然而在所有的网络中,每一层都是通过层间接口向上一层提供一定的服务,而把这种服务是如何实现的细节对上层加以屏蔽。

如图 1-1 所示,层次结构包括以下几个含义:

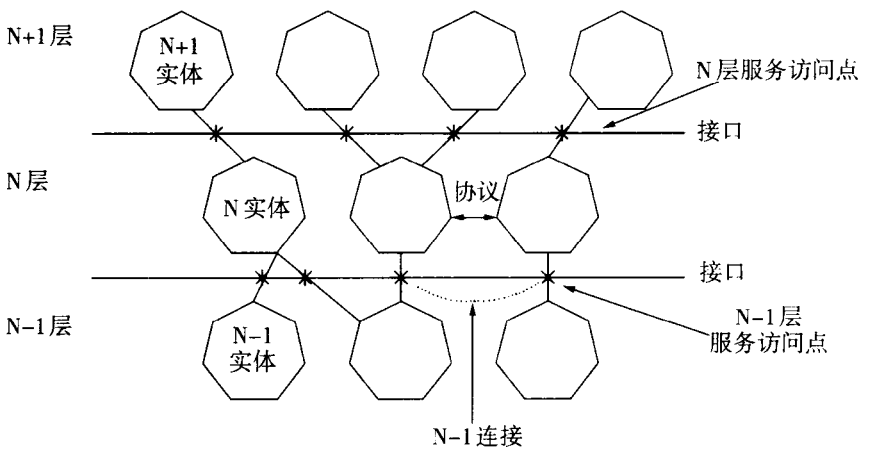


图 1-1 分层的概念

(1) 第 N 层的实体在实现自身定义的功能时,只使用 N - 1 层提供的服务。

(2) N 层向 N + 1 层提供服务,此服务不仅包括 N 层本身所执行的功能,还包括由下层服务提供的功能总和。

(3) 最低层只提供服务,是提供服务的基础;最高层只是用户,是使用服务的最高层;中间各层既是下一层的用户,又是上一层服务的提供者。

(4) 仅在相邻层间有接口,且下层所提供服务的具体实现细节对上层完全屏蔽。

N 层中的活动元素通常称为 N 层实体。不同机器上同一层的实体叫做对等实体。N 层实体实现的服务为 N + 1 层所利用。在这种情况下,N 层被称为服务提供者,N + 1 层是服务用户。服务是在服务访问点(SAP)提供给上层使用的。N 层 SAP 就是 N + 1 层可以访问 N 层服务的地方。每个 SAP 都有一个能够唯一地标识它的地址。在同样的意义上,我们可以把电话系统中的电话插孔看成是一种 SAP,而 SAP 地址就是这些插孔的电话号码。要想和他人通话,就必须知道他的 SAP 地址(电话号码)。类似地,在邮政系统中,SAP 地址是街名

和信箱。发一封信，必须知道收信人的 SAP 地址。

相邻层之间要交换信息，在接口处也必须遵循一定的规则。如图 1-2 所示，在典型的接口上，N + 1 层实体通过 SAP 把一个接口数据单元(IDU)传递给 N 层实体。IDU 由服务数据单元(SDU)和一些控制信息组成。SDU 是将要跨越网络传递给远方对等实体，然后上交给远方 N + 1 层的信息。控制信息被下层实体用来指导其功能任务的执行，但不是发送给远方对等实体的内容。

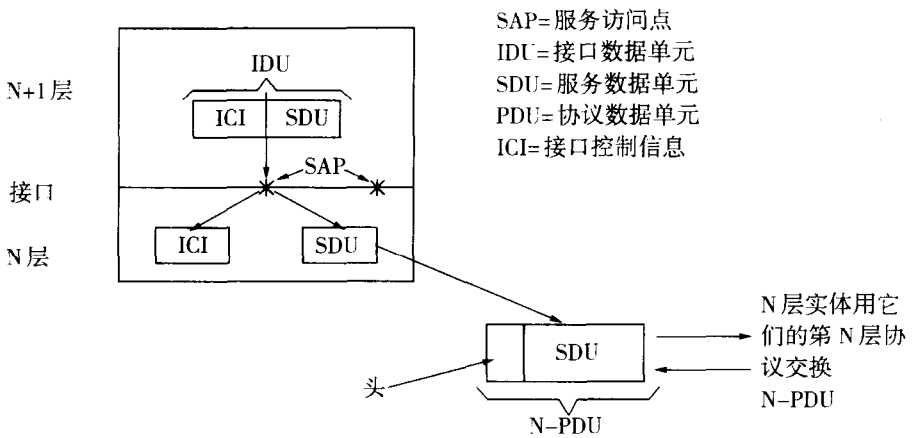


图 1-2　在接口处上下层之间的关系

为了传送 SDU，N 层实体可能把 SDU 分成几段，每一段加上一个头之后作为一个独立的协议数据单元(PDU)送出。PDU 被对等实体用于执行对等协议。对等实体根据 PDU 头部的信息分辨哪些 PDU 包含数据，哪些 PDU 包含控制信息，以及哪些 PDU 提供顺序号和计数等。

下层向上层提供的服务可以划分为面向连接的和无连接的两大类别。面向连接的服务类似于打电话。要和某人通话，拿起电话，拨号码，谈话，然后挂断。同样，在使用面向连接的服务时，用户首先要建立连接，传送数据，然后释放连接。连接本质上像个管道，发送者在管道的一端放入物体，接收者在另一端以同样的次序取出物体。

相反，无连接服务类似于邮政系统中普通信件的投递。每个报文(信件)带有完整的目标地址，并且每一个报文都独立于其他报文，经由系统选定的路线传递。在正常情况下，当两个报文发往同一目的地时，先发的先收到。但是，也有可能先发的报文在途中延误了，后发的报文反而先收到。而这种情况在面向连接的服务中是绝不可能发生的。

人们用服务质量(QOS)来评价每种服务的特征。通常，可靠的服务是由接收方确认收到的每一份报文，使发送方确信它发送的报文已经到达目的地这一方法来实现的。确认和有错时重传的处理过程增加了额外的开销和延迟，在许多情况下这是值得的，但有时也不尽然。对于文件传输这样的应用比较适合使用带有确认的面向连接的服务。文件的主人希望所有位都按发送的次序正确地到达目的地。想要传输文件的顾客不会喜欢一个虽然传输速度快但会不时发生混乱或丢失位的服务。对于另外一些应用，由确认和重传引起的延误则是不可接受的。数字化声音的传输就是一个例子。电话用户宁可听到线路上的一点杂音，

或偶尔混淆的语音，也不喜欢等待确认造成的延误。同样，在传输电影时，错了几个像素不会有伤大雅，但是电影突然停顿以等待传输错误的纠正却是很令人恼火的。

另外，也不是所有的应用程序都需要连接。例如，电子邮件越来越普及，电子邮寄宣传品的发送者可能不希望仅仅为了传一条消息而去经历建立和拆除连接的麻烦。

无确认无连接的服务称作数据报服务。电报服务与此类似，它不向发送者发回确认消息。在某些情况下，可能既希望免除建立连接的麻烦，又要求确保信息传送的可靠。此时，可以选用有确认的数据报服务。这很像寄出的一封挂号信又要求回执一样。当收到回执时，寄信人有绝对的把握相信信件已到达目的地而没有在途中丢失。

还有一种服务叫做“请求-应答”服务。使用这种服务时，发送者传送一个查询数据报，应答数据报则包含回答信息。例如，我们向图书馆询问某本书是否已经借出就属于这类情况。“请求-应答”服务通常被用于客户/服务器模式下的通信：客户发出一个请求，服务器作出响应。

服务在形式上是由一组原语(Primitive)来描述的。这些原语供用户和其他实体访问该服务时调用。它们通知服务提供者采取某些行动或报告某个对等实体的活动。服务原语可以划分为如表 1-1 所示的四类。现在我们以连接的建立为例，说明原语的用法。当一个实体发出连接请求(CONNECT. request)之后，一个 PDU(俗称分组)就被发送出去。

表 1-1 四类服务原语

原 语	意 义
Request	用户实体请求服务做某种工作
Indication	用户实体被告知某事件发生
Response	用户实体表示对某事件的响应
Confirm	用户实体收到关于它的请求的答复

接收方会收到一个连接指示(CONNECT. indication)，被告知某处的一个实体希望和它建立连接。收到连接指示的实体使用连接响应(CONNECT. response)原语表示它是否愿意建立连接。但无论是哪一种情况，请求建立连接的一方都能够通过连接证实(CONNECT. confirm)原语获知接收方的态度。

原语大多数都带有参数。例如，连接请求的参数可能指明要与哪台机器连接、需要的服务类别和在该连接上使用的最大报文长度。连接指示原语的参数可能包含呼叫者标识、需要的服务类别和建议的最大报文长度。如果被呼实体不同意呼叫实体所建议的最大报文长度，它可以在响应原语中作出一个反建议，呼叫方可从证实原语中获悉该反建议。这一协商的细节就是协议的内容。例如，在两个建议的最大报文长度不一致的情况下，协议就可能规定选择较小的值。

服务有“有证实”和“无证实”之分。有证实服务包括请求、指示、响应和证实四个原语，而无证实服务则只有请求和指示两个原语。CONNECT 服务总是有证实的服务，因为远程对等实体必须同意才能建立连接。在另一方面，数据传输可以是有证实的，也可以是无证实的，这取决于发送方是否要求确认。

为了使服务的概念更清楚，让我们继续考察一个简单的面向连接服务例子。它使用了如下所述的 8 个原语(如图 1-3 所示)：

(1) CONNECT. request：请求建立连接。

(2) CONNECT. indication：向被呼实体指示连接请求。

(3) CONNECT. response：被呼方用以表示接受或拒绝连接请求。

(4) CONNECT. confirm：通知呼叫方建立连接的请求是否被接受。

(5) DATA. request：请求发送数据。

(6) DATA. indication：表示数据的到达。

(7) DISCONNECT. request：请求释放连接。

(8) DISCONNECT. indication：通知对等实体释放的要求。在本例中，CONNECT 是有证实的服务(需要有明确的答复)，而 DISCONNECT 是无证实服务(不需要答复)。

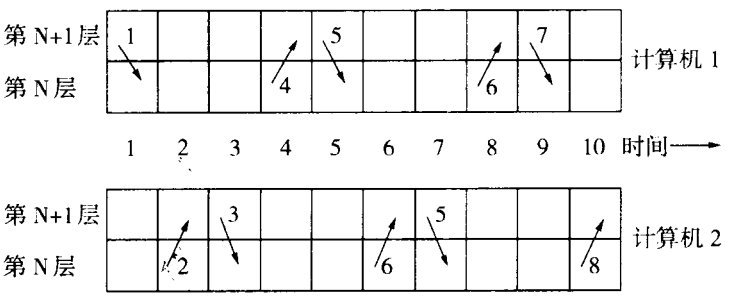


图 1-3　在一个面向连接的例子中服务原语的使用情况

图 1-3 示出了在连接建立、数据传送和连接释放的过程中对上述 8 个原语的使用情况。每一步骤都涉及一台计算机内两层之间的交互。图中的箭头尾部的数字是代表前面叙述的 8 种服务的号码。可以看出，每一“请求”或“响应”稍后都在对方产生“指示”或“证实”。在本例中，服务用户在第 N + 1 层，而服务提供者在第 N 层。

在典型的情况下，当接收到一个服务原语时，一层的协议实体读原语中的参数，并把它们与附加的协议控制信息相结合形成该层的 PDU。所产生的 PDU 再放到带有附加参数的服务原语的用户数据段中，以传递给相邻下层。这可以用图 1-4 来说明。

应该指出，服务和协议是完全不同的概念，但二者又常常被混淆在一起。它们之间的区别是如此重要，以致于我们在此必须再强调一次。服务是各层向它的上层提供的一组原语。尽管服务定义了该层能够为它的上层完成的操作，但丝毫也未涉及这些操作是如何完成的。服务定义了两层之间的接口，上层是服务用户，下层是服务提供者。

与之相对比，协议是定义在相同层次的对等实体之间交换的帧、分组和报文的格式及含义的一组规则。实体利用协议来实现它们的服务定义。只要不改变提供给用户的服务，实体可以任意地改变它们的协议。这样，服务和协议就被完全地分离开来。

我们可以把服务跟程序设计语言相类比。服务就像程序设计语言中的抽象数据类型。抽象数据类型定义了能在一个目标上执行的操作，但并不说明这些操作是如何实现的。协议关系到服务的实现，但对服务的用户来说是不可见的。

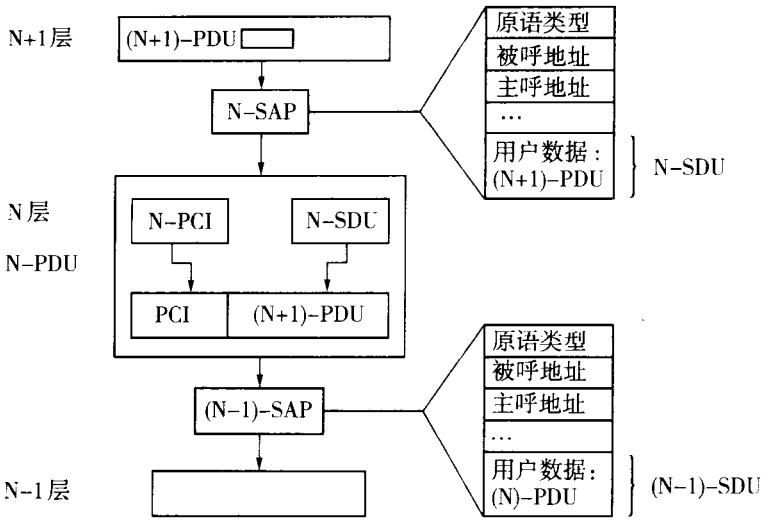


图 1-4 上下层之间的交互作用

1.2 OSI 参考模型

对于结构化的网络协议设计，一般将层和协议的集合叫作网络体系结构。制定通信协议的主要目的是要保证两个通信实体能够发送、接收并解释它们想要交换的信息。网络体系结构则定义大的框架，协议功能是在该框架中被适当地定义的。体系结构对于减少存在于端点到端点通信任务中固有的概念复杂性很有价值。现今大多数的体系结构都是基于层次的概念。在这种体系结构中，一个端到端的通信任务是通过逐次地在每个协议层中不断增加“确切含义”来完成的。

多年来国际标准化组织、学术团体、各个国家的许多研究机构和大的公司都十分重视对计算机网络体系结构的研究和开发。目前比较著名的体系结构是国际标准化组织(ISO)提出的开放系统互联(OSI)参考模型和美国国防部研制的 TCP/IP 协议体系。另外 IBM 公司的 SNA 体系(系统网络结构)、DEC 公司的 DNA 体系(数字网络结构)以及 Novell 公司和微软公司提出的局域网协议结构也很有影响。在本章内后面的讨论中，我们将重点叙述 OSI 参考模型和 TCP/IP 协议体系，并在此基础上介绍一个综合的实用模型，即采用五个层次的经修改的 OSI 模型。

OSI 参考模型如图 1-5 所示。该模型基于国际标准化组织(ISO)的建议，是作为要对各种层次上使用的网络协议实现国际标准化的工作的第一步而提出来的。它的提出是要为协调标准的研制提供一个共同的基础，允许现存的和正在演变的标准化活动有一致的框架和前景。其最终目的是，允许任一支持某种可用标准的计算机的应用进程自由地与任何支持同一标准的计算机的应用进程进行通信，而不管计算机是由哪个厂商制造的。正因为如此，该模型被称为开放系统互联(OSI)参考模型。

需要强调的是 OSI 给出的仅是一个概念上和功能上的标准框架，是将异构系统互联的

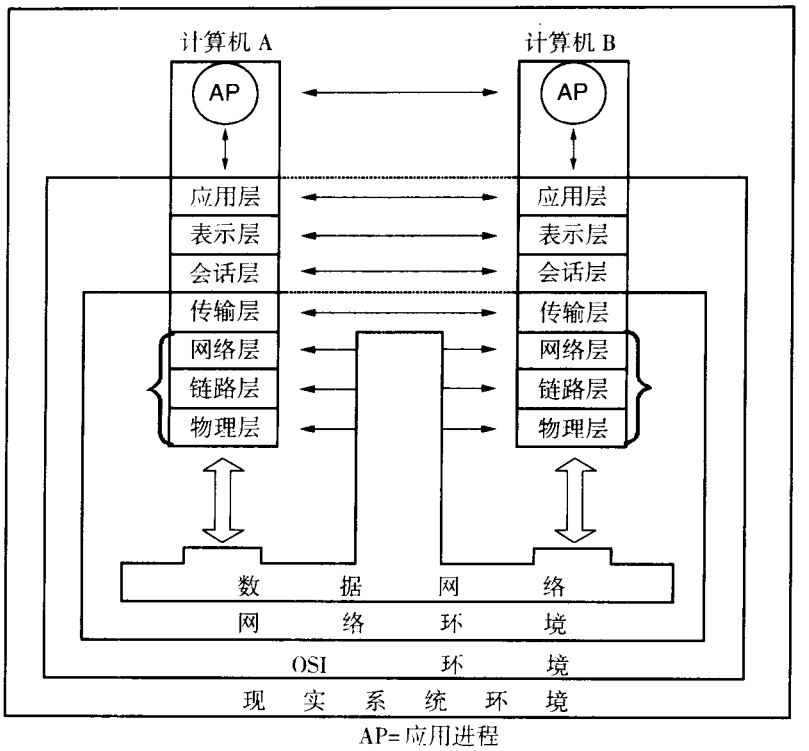


图 1-5　OSI 参考模型总体结构

一种标准分层结构。它定义的是一种抽象结构，而并非是具体的描述，模型本身不是一组有形的可操作的协议集合，即不包含任何具体的协议定义，也不包括强制的实现一致性。网络体系结构与实现无关。

1.2.1　模型结构

如前所述，ISO 为参考模型采用了分层的方法。整个通信子系统划分为若干层。每层执行一种明确定义的功能。从概念上讲，这些层可以被看成执行两类功能，即依赖于网络的功能和面向应用的功能。由此产生了下列三种不同的操作环境：

(1) 网络环境

涉及跟不同类型的下层数据通信网络有关的协议和标准。

(2) OSI 环境

包括网络环境和面向应用的协议和标准，允许末端系统(计算机)以开放的方式互相通信。

(3) 现实系统环境

建立在 OSI 环境之上，参与一个厂商自己的专有软件和服务，通过这种软件和服务完成特定的分布式信息处理任务。

OSI 模型依赖于网络的和与网络无关的这两个成分都实现为若干个协议层的形式，每

一协议层之间的边界,以及每一层所执行的功能根据早先标准化活动中所得到的经验产生。

在总体通信策略的指导下,每一层都执行一种明确定义的功能。它根据某种定义了的协议来运行,用户数据和附加的控制信息以报文形式(PDU)在本地层和远方系统的对应层之间进行交换。每一层在它自己和紧挨着的上层和下层之间都有明确定义的接口,这样就使一个特别协议层的实现独立于所有其他层。

在图 1-6 所示的 7 个 OSI 层次中,最低 3 层(1 ~ 3)是依赖网络的,牵涉到将两台通信计算机链接在一起所使用的数据通信网的相关协议。高 3 层(5 ~ 7)是面向应用的,牵涉到允许两个末端用户应用进程交互作用的协议,通常是由本地操作系统提供的一套服务。中间的传输层为面向应用的上 3 层遮蔽了跟网络有关的下 3 层的详细操作。本质上讲,它建立在由下 3 层提供的服务上,为面向应用的高层提供网络无关的信息交换服务。

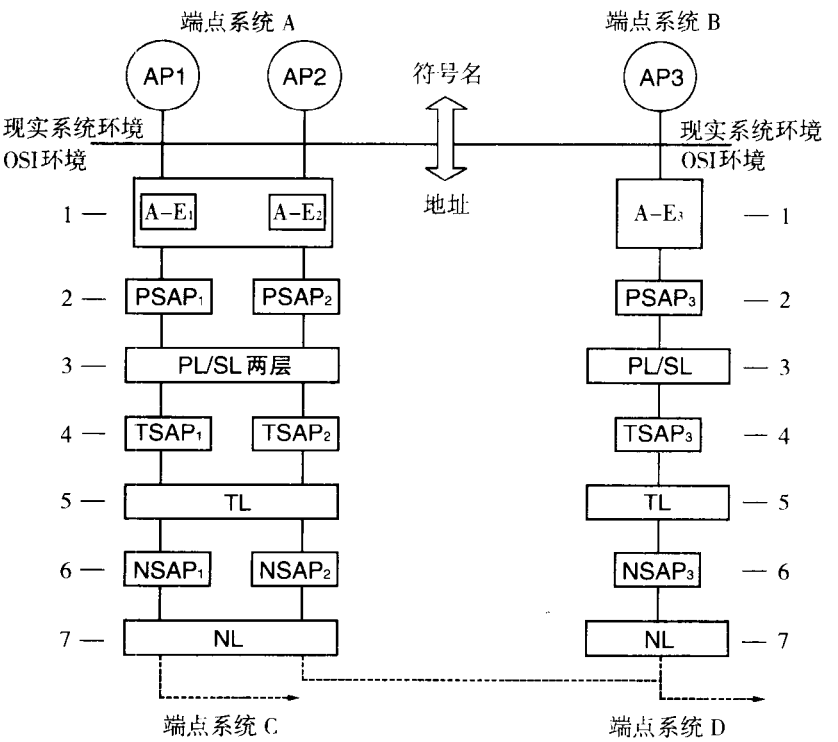


图 1-6 OSI 地址结构

每一层的功能以协议形式正规描述,协议定义了某层跟另一(远方)系统中的一个类似层(对等层)通信所使用的一套规则和约定。每一层向相邻上层提供一套确定的服务,并且使用由相邻下层提供的服务向远方对等层传输跟该层协议相关的信息单元。例如,传输层为它上面的会话层提供可靠的与网络无关的信息传输服务,并且使用其下面网络层所提供的服务将与传输层协议有关的一组信息单元传送给另一系统中的一对等传输层。在概念上,每一层都根据一个明确定义的协议跟一个远方系统中的一个类似对等层通信,但在实际上该层所产生的协议信息单元是借助于相邻下层所提供的服务传送的。

下面我们将从最下层开始，逐次讨论 OSI 参考模型的各层。注意，OSI 模型本身并未确切地描述用于各层的具体服务和协议，它仅仅告诉我们每一层应该做什么。不过，ISO 确实已为各层制定了一些标准，但它们并不是参考模型的一部分，它们是作为独立的国际标准公布的。

物理层

物理层(Physical Layer)涉及到通信在信道上传输的原始比特流。设计上必须保证一方发出二进制"1"时，另一方收到的也是"1"而不是"0"。这里的典型问题是用多少伏特电压表示"1"，多少伏特电压表示"0"；一个比特持续多少微秒；传输是否在两个方向上同时进行；最初的物理连接如何建立和完成通信后连接如何终止；网络接插件有多少针以及各针的用途。这里的设计主要是处理机械的、电气的和过程的接口，以及物理层下的物理传输介质问题。

数据链路层

数据链路层(Data Link Layer)指定在网络上沿着网络链路在相邻节点之间移动数据的技术规范。它的主要任务是加强物理层传输原始位的功能，使之对网络层显现为一条无错线路。发送方把输入数据分装在数据帧(Data Frame)里(典型的帧为几百字节或几千字节)，按顺序传送各帧，并且有可能要处理接收方回送的确认帧(Acknowledgement Frame)。因为物理层仅仅接收和传送比特流，并不关心它的意义和结构，所以只能依赖各链路层来产生和识别帧边界。可以通过在帧的前面和后面附加上特殊的二进制编码模式来达到这一目的。如果这些二进制编码偶然在数据中出现，则必须采取特殊措施以避免混淆。

传输线路上突发的噪声干扰可能把帧完全破坏掉。在这种情况下，发送方机器上的数据链路软件可能要重传该帧。然而，相同帧的多次重传也可能使收方收到重复帧，比如接收方给发送方的确认丢失以后，就可能收到重复帧。数据链路层要解决由于帧的破坏、丢失和重复所出现的问题。数据链路层可能向网络层提供几类不同的服务，每类都有不同的服务质量和价格。

数据链路层要解决的另一个问题(在大多数其他层上也存在)是防止高速的发送方的数据把低速的接收方"淹没"。因此需要某种流量调节机制，使发送方知道当前接收方还有多少缓存空间。通常流量调节和出错处理的功能同时完成。

如果线路能用于双向传输数据，数据链路软件还必须处理新的麻烦，即从 A 到 B 数据帧的确认帧将同从 B 到 A 的数据帧竞争线路的使用权的问题。捎带技术(Piggybacking)就是解决此问题的一种巧妙的方法，我们将在以后讨论它。

广播式网络在数据链路层还要处理新的问题，即如何控制对共享信道的访问。数据链路层的一个特殊的子层——媒体访问子层，就是专门处理这个问题的。

网络层

网络层(Network Layer)关系到子网的运行控制，其中一个关键问题是确定分组从源端到达目的端的路由。路由选择可以使用网络中固定的静态路由表，路由几乎保持不变；也可以在每一次会话开始时决定(例如通过终端对话决定)；还可以根据当前网络的负载状况，高度灵活地为每一个分组决定路由。

如果在子网中同时出现过多的分组，它们将相互阻塞通路，形成瓶颈。此类拥塞控制也属于网络层的功能范围。

因为拥有子网的人总是希望他们提供的子网服务能得到报酬，所以网络层常常设有记账功能。最低限度，软件必须对每一个顾客究竟发送了多少分组、多少字符或多少位进行记数，以便于生成账单。当分组跨越国界时，由于双方税率可能不同，记账则更加复杂。

当分组不得不跨越一个网络以到达目的地时，新的问题又会产生。第二个网络的寻址方法可能和第一个网络完全不同；第二个网络可能由于进来的分组太长而无法正确接收；两个网络使用的协议也可能不同。网络层必须解决这些问题才能够实现异种网络的互联。

在广播网络中，选择路由问题变得很简单，因此网络层很弱，甚至不存在。

传输层

传输层(Transport Layer)的基本功能是从会话层接收数据，并且在必要时把它分成较小的单位，传递给网络层，并确保到达对方的各段信息正确无误，而且，这些任务都必须高效率地完成。从某种意义上讲，传输层使会话层不受硬件技术变化的影响。

通常，会话层每请求建立一个传输连接，传输层就为其创建一个独立的网络连接。如果传输连接需要较高的信息吞吐量，传输层也可以为之创建多个网络连接，让数据在这些网络连接上分流，以提高吞吐量。另一方面，如果创建或维持一个网络连接不合算，传输层可以将几个传输连接复用到一个网络连接上，以降低费用。然而，在任何情况下，都要求传输层能使多路复用对会话层透明。

传输层也要决定向会话层提供什么样的服务。最流行的传输连接是一条无错的、按发送顺序传输报文或字节的点到点的信道。但是，还有的传输服务不能保证传输次序的独立报文传输和多目标的报文广播。究竟采用哪种服务是在建立连接时确定的。

传输层是真正的从源到目标的"端到端"的层。也就是说，源端机上的某程序，利用报文头和控制报文与目标机上的类似程序进行对话。在传输层以下的各层中，协议是每台机器包括中间节点都要参照执行的协议，而不是最终的源端机与目标机之间的协议，通常在它们中间可能还有多个路由器。这些路由器都要对路过的信息块进行 1 层 ~ 3 层的处理。图1-6说明了这种区别，1 层 ~ 3 层是链接起来的，4 层 ~ 7 层是端到端的。

很多主机有多道程序在运行，这意味着这些主机有多条连接进出，因此需要有某种方式来区别报文属于哪条连接。识别这些连接的信息可以放入传输层的报文头。

除了将几个报文流多路复用到一条通道上，传输层还必须解决跨网络连接的建立和拆除。这需要某种命名机制，使机器内的进程可以讲明它希望与谁会话。另外，还需要一种机制以调节通信量，使高速主机不会发生过快向低速主机传输数据的现象。这样的机制称为流量控制(Flow Control)，它在传输层(同样在其他层)中扮演着关键角色。

会话层

会话层(Session Layer)允许不同机器上的用户建立会话(Session)关系。会话层允许进行类似传输层的普通数据传输，并提供了对某些应用有用的增强服务会话，也可被用于远程登录到分时系统或在两台机器间传递文件。

会话层服务之一是管理对话。会话层允许信息同时双向传输，或任一时刻只能单向传输。若属于后者，则类似于单线铁路，会话层将记录此时该轮到哪一方了。

一种与会话有关的服务是令牌管理(Token Management)。有些协议保证双方不能同时进行同样的操作，这一点很重要。为了管理这些活动，会话层提供了令牌。令牌可以在会话

双方之间交换，只有持有令牌的一方可以执行某种关键操作。

另一种会话服务是同步(Synchronization)。如果网络平均每小时出现一次大故障，而两台计算机之间要进行长达两小时的文件传输时该怎么办？每一次传输中途失败后，都不得不重新传输这个文件。而当网络再次出现故障时，又可能半途而废了。为了解决这个问题，会话层提供了一种方法，即在数据流中插入检查点。每次网络崩溃后，仅需要重传最后一个检查点以后的数据。

表示层

表示层(Presentation Layer)完成某些特定的功能，由于这些功能常被请求，因此人们希望找到通用的解决办法，而不是让每个用户来实现。值得一提的是，表示层以下的各层只关心可靠的传输比特流，而表示层关心的是所传输的信息的语法和语义。

表示层服务的典型的例子是用一种大家一致同意的标准方法对数据编码。大多数用户程序之间并不是交换随机的比特流，而是诸如人名、日期、货币数量和发票之类的信息。这些对象是用字符串、整型、浮点数的形式，以及由几种简单类型组成的数据结构来表示的。不同的机器用不同的代码来表示字符串(如 ASCII 和 Unicode)和整型(如二进制反码和二进制补码)等。为了让采用不同表示法的计算机之间能进行通信，交换中使用的数据结构可以用抽象的方式来定义，并且使用标准的编码方式。表示层管理这些抽象数据结构，并且在计算机内部表示法和网络的标准表示法之间进行转换。

应用层

应用层(Application Layer)包含大量人们普遍需要的协议。例如，世界上有成百种不兼容的终端型号。如果希望一个全屏幕编辑程序能工作在网络中许多不同的终端类型上，每个终端都有不同的屏幕格式、插入和删除文本的换码序列、光标移动等，其困难可想而知。

解决这一问题的方法之一是定义一个抽象的网络虚拟终端(Network Virtual Terminal)，编辑程序和其他所有程序都面向该虚拟终端。而对每一种终端类型，都写一段软件来把网络虚拟终端映射到实际终端。例如，当把虚拟终端的光标移到屏幕左上角时，该软件必须发出适当的命令使真正的终端的光标移动到同一位置。所有虚拟终端软件都位于应用层。

应用层另一个功能是文件传输。不同的文件系统有不同的文件命名原则，文本行有不同的表示方式等。不同的系统之间传输文件所需处理的各种不兼容问题，也同样属于应用层的工作。此外还有电子邮件、远程作业录入、名录查询和其他各种通用和专用的功能。

1.2.2　协议层

当描述任何协议层的动作时，从一开始就要将该层所提供的服务、该层的内部操作(即协议)和该层所使用的服务区别开来。每一层的功能只能在与其他层的关系中进行定义。实现单个协议层的程序设计人员只需要知道该层向上层提供的服务，该层的内部协议，以及为了将跟该层协议有关的适当的信息项传送给远方系统中的类似由下层所提供的服务。该层软件实现人员不需要知道其他层更多的东西。

例如,为描述传输层的功能,只需要考虑:

(1) 传输层要向会话层提供的一套明确的服务,其目的是为了把会话层信息单元传输到远方系统的同等会话层。

(2) 传输层内部操作(协议),牵涉到诸如建立和管理与远方系统中对等传输层的逻辑连接,以及在所建立的链接上传送的传输层信息单元的错误处理及流控制这样的功能。

(3) 为了将这些信息单元传送给一个对等传输层由网络层所提供的服务。

在描述每一协议层的功能时,上述三个方面要分别对待。

每个协议层的描述包括两套文档:服务定义文本和协议描述文本。

服务定义文本包含该层向它的相邻上层提供的服务,即用户服务的描述。通常的形式是一组明确的服务原语(Service Primitives),每个原语都带有一套相关的服务参数。正是通过这样的服务参数,上层发起对远方系统相似通信层的信息传送。

协议描述文本包括:

(1) 准确定义本层协议实体跟远方系统中对等协议实体通信所使用的协议数据单元(Protocol Data Units-PDU)。

(2) 用于传输每个 PDU 类型本层所使用的服务(即相邻下层提供的服务)的描述。

(3) 用一种形式描述方法给出的对于协议实体运行的准确定义。

1.2.3 服务定义

在任何网络中,都有必要区别用户应用进程的标识和该用户进程在网络上驻留的位置。用户应用进程的标识通常采用符号名称的形式,而其在网络上的位置则用地址的形式表示。这类似于通过邮局寄信使用的名字和地址,名字是收信人的标识,地址则是收信人的当前住址。由于可能有许多应用进程驻留在同一计算机中,除了计算机在网内的物理地址之外,还必须在应用进程的地址中包括一台计算机内部附加的寻址信息。这类似于大楼内的层号和房间号。

由于在用户层使用符号名作为应用进程的标识,故有必要使得每个用户应用进程(AP)的名字在特定的 OSI 环境中是唯一的。通常,一个应用进程不知道其他应用进程在网络中实际的物理位置,因此一个应用进程跟其他应用进程通信只是简单地指定对方的名称。为了保证名字在特定的 OSI 环境中的唯一性,必须提供某种手段管理应用进程(用户进程和服务提供者进程)名字的分配。提供这种功能的系统称为名字服务器。

对于相对小的环境,比如连接到一个局域网的基于计算机的系统分布式群体,通常有一个名字服务器就足够了。然而对于较大的环境,几千个系统通过若干局域网和广域网互联,一个名字服务器就难以管理了。在这种情况下,每个分网分别使用一个名字服务器。为保证每个名字在整个环境内是唯一的,在一个子网中一个用户名字前面加上一个前缀,标识在整个 OSI 环境中的子网。

虽然在用户级使用名字,但在 OSI 环境内部都使用地址来确定应用进程当前驻留的计

算机在网内的物理位置,以及应用进程当前附接的应用层协议实体。OSI 环境负责将用户应用进程指定的通信符号名关联到一个具体的网络范围地址。符号名和地址之间的映射列表包含在一个系统目录中,因此在原则上只要简单地改变系统目录中的登录项就可以改变一个应用进程的实际物理位置。

在 OSI 环境中,地址的形式是将若干称为服务访问点(SAP—Service Access Points)的子地址串接在一起(如图 1-6 所示)。一个应用进程(AP)的地址表示为:

$$\text{AP 地址} = \text{PSAP} + \text{SSAP} + \text{TSAP} + \text{NSAP}$$

其中,PSAP 是在应用进程附接到的应用层协议实体和表示层之间的服务访问点子地址,SSAP 是在表示层和会话层之间的服务访问点子地址等等。实际上,PSAP 和 SSAP 是同一个子地址,而且,是 NSAP 包含 AP 所驻留的系统的网络范围的物理地址。P/SSAP 和 TSAP 地址在系统内部可决定用户应用进程当前所附接的具体应用层协议实体。

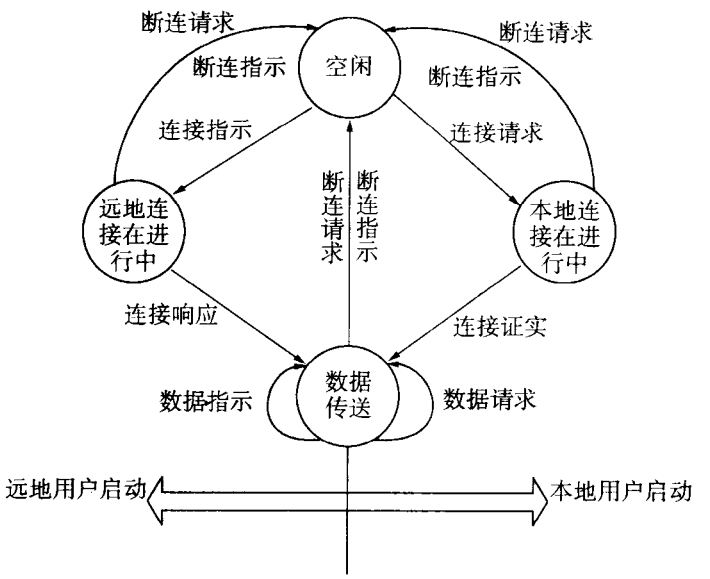
由一层提供的服务用一组原语来描述。通常,一次传送由该层的用户通过该层接口传递一个请求原语开始。这又引起该层内本地协议实体产生一个相关的 PDU,这个 PDU 使用下层提供的服务传递给远方系统中的一个通信(对等)协议实体。然后远方系统中的对等协议实体建立一个相关的指示原语,并将其向上传递给通信用户。在非证实服务的情况下,这就完成了传送,但对于证实服务,通信用户接着发一个响应原语。这又引起对方本地协议实体产生一个相关的 PDU,并使用下层提供的服务,回送给源协议实体。收到这个 PDU,源协议实体建立一个证实原语,并将其向上传递给用户,从而完成这次传送。

通常,原语名字包括原语类型和提供服务的层的标识,于是:

- T. CONNECT. request 是由传输服务用户(TS-user)——即会话层——发出的一个请求原语,其目的是要跟远方用户(会话层)建立一种(逻辑的)传输连接。
- S. DATA. indication 是由对等(通信)会话层发给它上面的表示层的一个指示原语,并且涉及从远方表示层收到的数据。

对应于每一层都有一套原语去提供诸如连接建立、数据传送之类的服务。当一层的接口收到一个服务原语时,首先确定该原语是否符合正确的顺序。通常在建立连接之前不可能发一个数据传送请求原语。因此在标准文本内,与一层相关的可以接受的服务原语顺序可用状态变迁图或者顺序表说明。图 1-7 给出了一个示例。

在示例中,所支持的用户服务是允许一个用户先跟一个远方(通信)用户建立一条逻辑连接,然后通过这条连接传送数据,最后断开(Clear)连接。通常,状态变迁图只是表示用户接口所允许的正确原语顺序。顺序表则是一个比较准确的定义,它示出所有可能的顺序,包括有效的和无效的。因此,顺序图用于实现的目的,确保一个收到的原语在允许的顺序中,通常收到一个不符合顺序的原语是一种协议违犯,其结果是使得有关的连接被清除(断连)。



(a) 状态变迁图

这个原语→ / 可以后随↓	连接请求	连接指示	连接响应	连接证实	数据请求	数据指示	…
连接请求							
连接指示							
连接响应		+					
连接证实	+						
数据请求			+	+	+	+	
数据指示			+	+	+	+	
⋮							

注：+ 表示允许　　空格表示出错

(b) 顺序表

图 1-7　原语顺序

1.2.4　协议描述

一层的协议描述文本包括：

- 对于与协议实体有关的 PDU 的类型及它们的目的的定性描述，加上在每个 PDU 中存在的段及其用法的描述。
- 对于在协议实体操作的各个阶段所遵从的过程和传送每个 PDU 类型时所使用的服务的描述。
- 使用一种形式描述方法对协议实体运行的准确定义。