第一部分 引言与底层网络技术

第一章 引言

主要内容

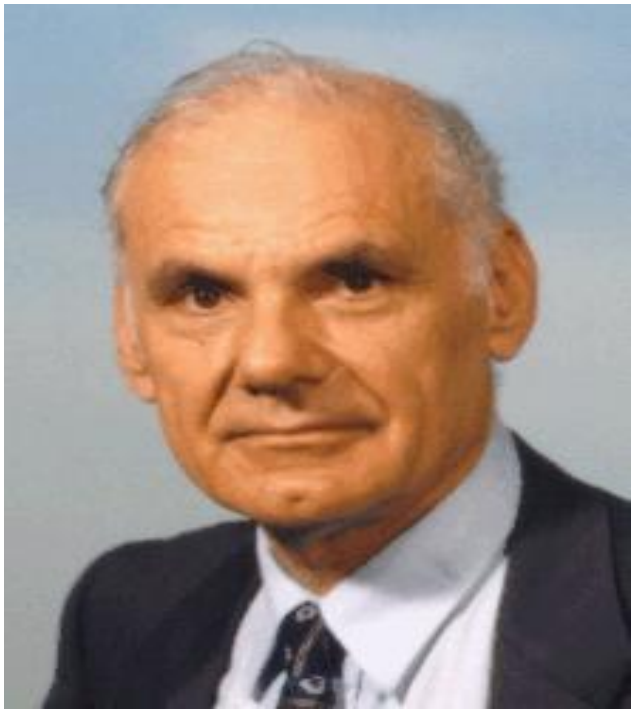
- 1.1 协议体系结构
  - 1.1.1 协议体系结构概念
  - 1.1.2 协议层次模型
  - 1.1.3 协议
  - 1.1.4 OSI与TCP/IP参考模型
- 1.2 网络互联与TCP/IP
  - 1.2.1 用IP实现异构网络互联
  - 1.2.1 TCP/IP协议族的引入
- 1.3. 协议的描述与验证
  - 1.3.1 协议工程的概念
  - 1.3.2 有限状态机模型
  - 1.3.3 停等协议的有限状态机

1995年10月24日，美国联邦网络委员会为**Internet**作了如下**定义**:

Internet是一个全球性的信息系统，系统中的每台主机都有一个全球唯一的主机地址，地址格式通过IP协议定义。系统中主机与主机间的通信遵守TCP/IP协议标准，或是其它与IP兼容的协议标准来交换信息。在以上描述的信息基础设施上，利用公网或专网的形式，向社会大众提供资源和服务。

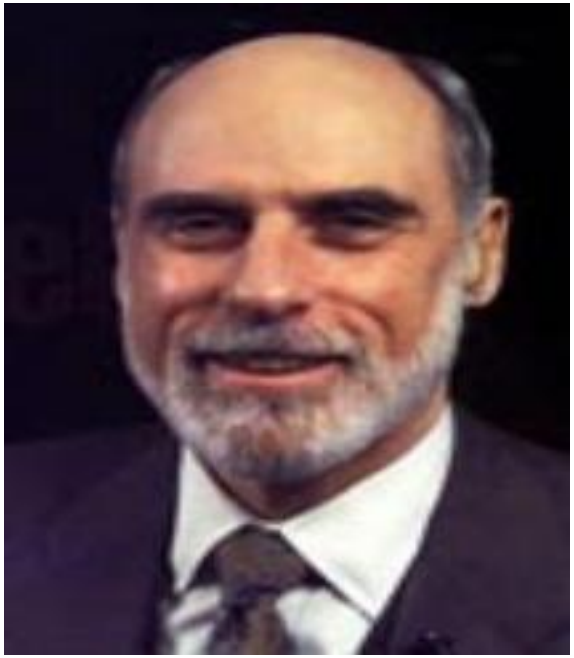
上述定义充分说明了TCP/IP与Internet的关系。

美国著名科学家Lawrence Roberts因提出、设计并实现了ARPANET，而被称为“ARPANET之父”， ARPANET 使用了报文交换机和分组交换技术。



Lawrence Roberts
“ARPANET之父”

美国著名科学家Vinton G. Cerf和Robert Kahn因在TCP/IP上作出的突出贡献，获得2004年图灵奖，并于2014年3月被IEEE评选为Internet之父。

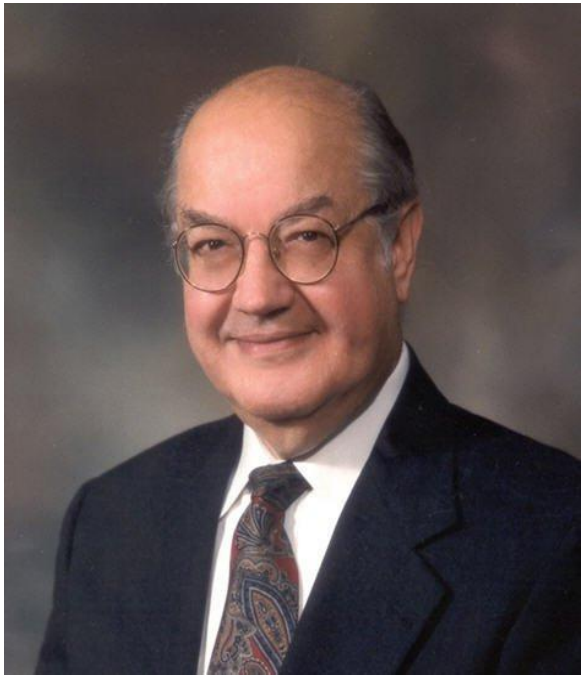


Vinton G. Cerf　　Robert Kahn

“Internet之父”

美国科学家Leonard Kleinrock和Paul Baran 因在分组交换技术上的杰出贡献，于2014年3月Leonard Kleinrock和Paul Baran被IEEE评选为Internet之父。

Leonard Kleinrock
伦纳德·克兰罗克



Paul Baran
保罗·巴兰

欧洲原子核研究组织 CERN，英国科学家Tim Berners-Lee因发明了万维网，而被称为“万维网之父”

Tim Berners-Lee
“万维网之父”

1.1 协议体系结构

1.1.1 协议体系结构概念

- 计算机网络是一个十分复杂的系统
  - 不同的通信介质——有线(光缆、双绞线……)、 无线……
  - 不同种类的设备——主机、路由器、交换机……
  - 不同的操作系统——Unix、Windows… …
  - 不同的业务种类——分时、交互、实时… …
  - 不同的底层网络技术——以太网、令牌环网……

- 相互通信的两个计算机系统必须高度协调工作才行，而这种“协调”是相当复杂的。

- 分层模型，分而治之!

诸多不同类型的计算机和网络设备要进行通信，必须使它们采用相同的信息交换规则（协议），为了减少网络协议设计的复杂性，通常情况下，采用分层模型开发网络协议。

  - 分层：将整个通信过程划分成许多小问题，为每个小问题单独设计一个协议。每层实现一种特定的服务（功能）。

    这样简化了每个协议的设计、分析、编码、测试，并且简化了网络的故障排除。

  - 依赖下层提供的服务，实现自己内部的功能，并向上层提供服务。

- 分层的原则：
  - 将相似的功能集中在同一层内，必要时可将层的功能再分成子块，层数不宜过多，以避免层间接口的开销变大。
  - 每一层实现定义明确的功能，并应尽量局部化，当功能差别较大时应分层处理，减少层与层之间必要的消息传递。
  - 各层只对相邻的上下层定义接口，提供或访问服务。
  - 层次的划分应有利于标准化工作。
- 层次结构方法有什么优点?
  - 独立性强

    上层只需了解下层通过层间接口提供什么服务—黑箱方法。
  - 适应性强

    只要服务和接口不变，每层的实现方法可任意改变。
  - 易于实现和维护

    系统结构清晰，实现、调试和维护变得简单和容易。

    设计人员能专心设计和开发所关心的功能模块。

- **协议体系结构（protocl architecture）**：为完成计算机间的通信合作，将计算机互联的功能划分成定义明确的层次，规定了相邻层间通信的接口与服务，以及同层实体通信的协议。这种层和协议的集合称之为**协议体系结构（protocol architecture）**。它精确定义了计算机网络及其部件所应完成的功能。
- **实现 (implementation)**：遵循这种体系结构的前提下用何种硬件或软件完成这些功能的问题。体系结构是抽象的，而实现则是具体的，是真正在运行的计算机硬件和软件。
- **协议栈（protocol stack)**：协议体系结构中竖直排列的这些层中的一组协议（每一层一个或几个协议）。也就是在一个系统中所使用的一组协议。

简单的协议体系结构

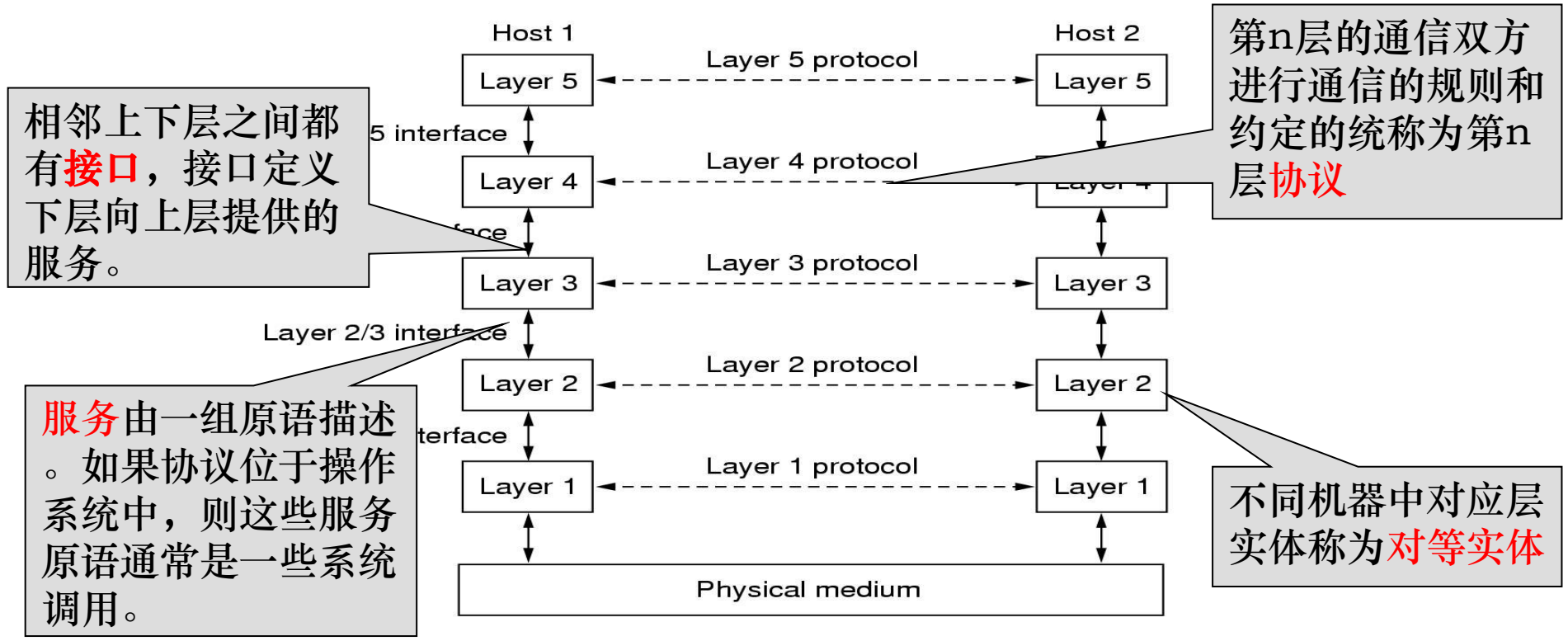
为有助于理解协议体系结构中多层通信，给出一个简单的协议体系结构。分布式数据通信涉及三个部分，应用程序、计算机和网络。应用程序如浏览器，它运行在计算机上，并且计算机支持多个应用进程的并发运行。计算机连接到网络上，需要交换的数据通过网络从一台计算机传送到另一台计算机上。即在两台不同计算机上的运行的应用进程间进行数据传递。这样可将通信划分为三个相对独立的层次：网络接口层、运输层和应用层。

网络接口层：实现计算机与所连网络之间的数据交换，网络接口层根据所使用的网络类型（不同的局域网和广域网）和技术（电路交换或分组交换）而采用不同的网络软件，这样与网络接入有关的功能划分为一个独立的层次，位于该层之上的运输软件就不必关心所使用的网络类型了。

运输层：通过网络连接的计算机上的应用进程之间的数据交换通常要求是可靠的，而这种可靠性机制与应用程序类型无关，并且都集中在同一层中，由所有应用程序共享，这一层称为运输层。

应用层：包含用于支持各种不同的用户应用逻辑，对不同类型的应用，需要一个独立的专门负责该应用的模块。

- 层(layer)
- 协议(protocol)
- 对等实体（peer entity)
- 接口（interface)
- 服务(service或业务)
- 原语(primitive)



实际上，数据并不是从一台机器的第 n层直接传递到另一台机器的第n层。相反，每一层都将数据和控制信息传递给它的下一层，这样一直传递到最低层，低一层下面的物理介质（physical medium)，通过它进行实际的通信。

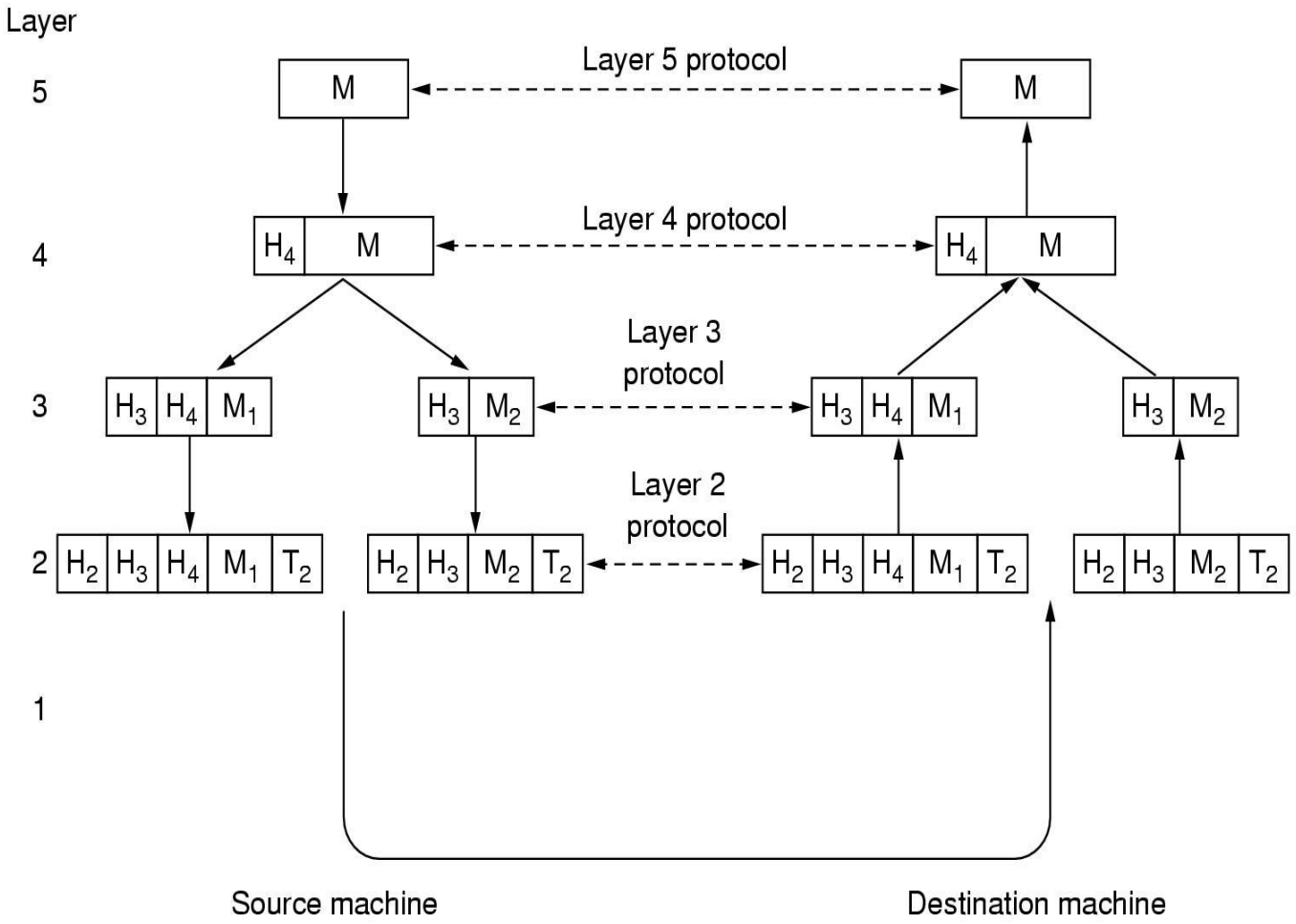
服务与协议的关系

- 协议是“水平的”，即是控制两个对等实体进行通信的规则的集合。 协议的实现保证了能够向上一层提供服务。
- 服务是“垂直的”，即服务是由下层向上层通过层间接口提供的功能。上层使用服务原语获得下层所提供的服务。

本层的服务用户只能看见服务而无法看见下面的协议。即下面的协议对上面的服务用户是透明的。

在协议的控制下，两个对等实体间的通信使得本层能够向上一层提供服务。要实现本层协议，还需要使用下层所提供的服务。

五层网络结构通信示例


Layer
5
4
3
2
1
M
Layer 5 protocol
M
H4 M
Layer 4 protocol
H4 M
Layer 3
protocol
H3 H4 M1
H3 M2
H3 H4 M1
H3 M2
Layer 2
protocol
H2 H3 H4 M1 T2
H2 H3 M2 T2
H2 H3 H4 M1 T2
H2 H3 M2 T2
Source machine
Destination machine

- 5层：应用程序产生消息M，并向下传递给第4层。
- 4层：在M前加上首部H4，构成本层的协议数据单元PDU，并向下传递给第3层。首部包含控制消息，如地址， 用于目的机器上对等实体向上递交消息。首部控制消息还可以是消息序号、消息大小和时间等。

  控制消息：根据各层不同需求确定，首部或尾部添加的控制信息是根据接收方的需求确定的。
- 3层：分割消息为较小的单元，并加上第3层首部H3构成分组后，独立选择输出线路。并把分组传递给第2层。
- 2层：在每个分组的前面加上一个首部，在后面加上一个尾部。通过接口传递个1层。
- 1层：向下在物理介质上传递消息。
- 接收端机器自底向上逐层传递消息，在传递过程中逐层剥离各层首部，消息M最终交给目的机器第5层对等实体。

协议数据单元（Protocol Data Unit，PDU）：每个层次将从上层接收的消息加上首部和尾部后就构成本层的协议数据单元。

虚拟通信：通信是水平的，每层对等实体之间，使用本层协议。对等实体之间类似于发送到对方（SendToOtnerSide)和从对方接收（GetFromOtherSide）这样的过程

物理通信：通信是垂直的，通过相邻层之间的层间接口与底下的层进行通信，并不是直接与另一端进行通信。

1.1.2 协议层次模型

为确保所形成的通信系统是完整的和有效的，必须认真构筑一整套协议。在这套协议的总体设计规划中，各协议的设计在功能上不能重复，每个协议只处理通信功能的一部分，而所有协议相互协作能高效统一的完成所有的通信功能，这套协议称为协议族或协议栈。把协议族中的各种协议集中成为一个统一的整体的抽象结构，被称为层次模型（layer model）。层次模型所描述的就是如何把通信问题的所有方面划分成一个个协调工作的层次结构。

使用OSI体系结构中的概念来描述n层协议模型，在OSI中，实体表示任何可以发送和接收信息的硬件或软件进程。在多数情况下，实体就是一个特定的模块。两个（N）实体在（N）层协议的控制下通信，使（N）层能够向上一层（N+1）提供服务。这种服务就是（N）服务。接受（N）服务的是上一层实体，即（N+1）实体。它们也称为（N）用户或更严格地称为（N）服务用户。

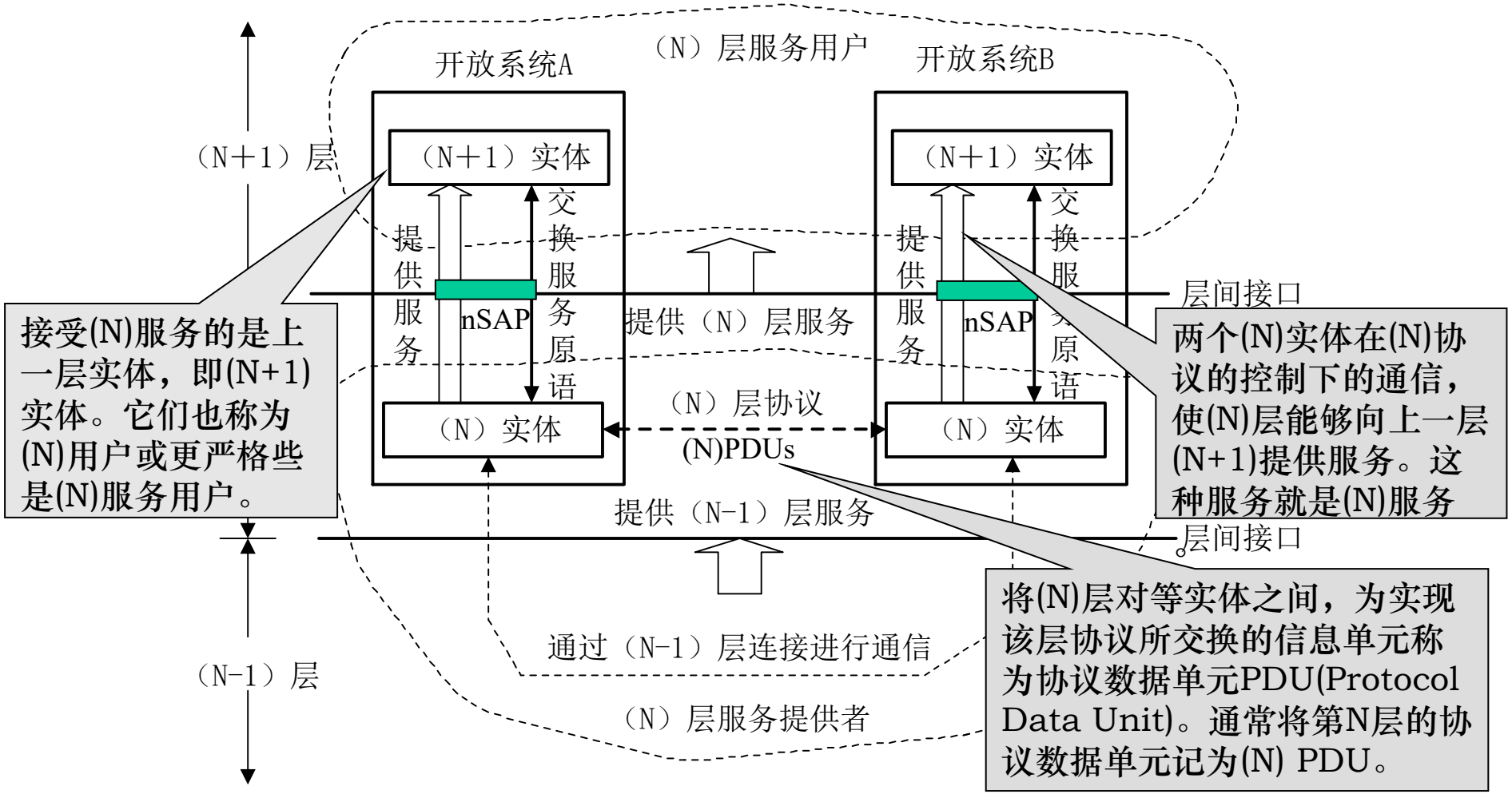
一个（N）实体向上一层所提供的服务由以下三部分构成：

（1）（N）实体自己提供的某种功能。

（2）从（N-1）层及其以下各层以及本地系统环境得到的服务。

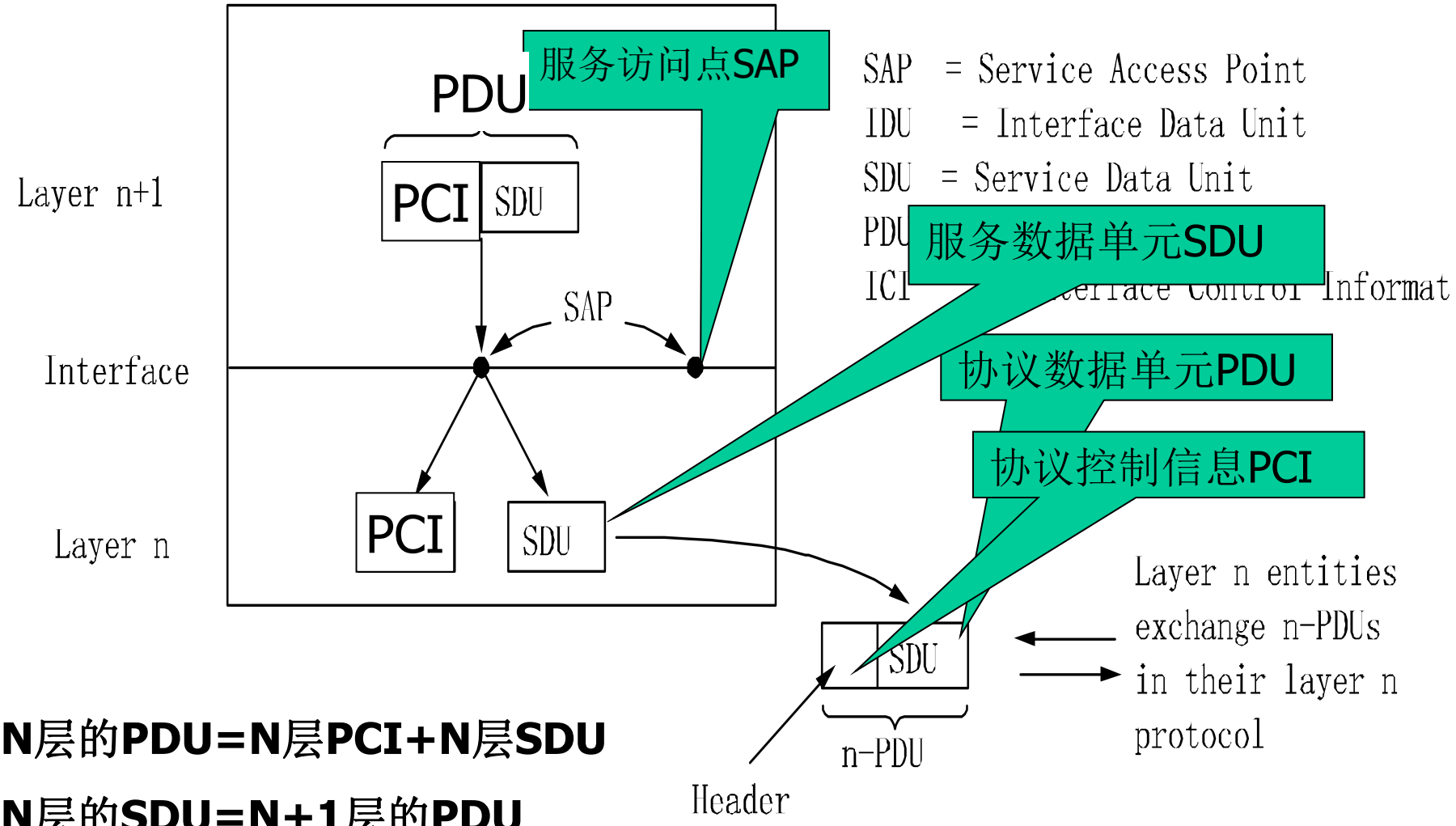
（3）通过与处在另一系统中的对等（N）实体的通信而得到的服务。

服务示意图



相邻层的上层都是通过下层完成本层的功能，其中下层称为服务提供者，上层称为服务使用者或服务用户，服务的提供和使用都是通过相邻层的接口,即服务访问点（SAP)进行。PDU通常有两部分构成：用户数据和协议控制消息PCI,PCI一般作为首部或尾部加在用户数据的前面或后面。

数据单元的名称与关系



N层的PDU=N层PCI+N层SDU

N层的SDU=N+1层的PDU

在OSI/RM中，数据单元是数据传送的单位可分为三种：协议数据单元PDU、接口数据单元IDU和服务数据单元SDU。

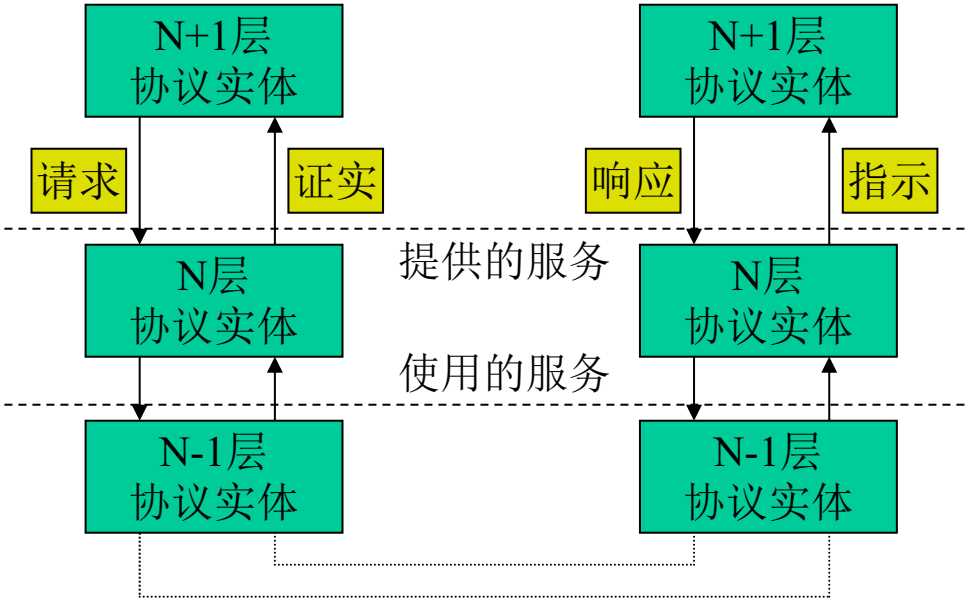
- **协议数据单元PDU**：用户数据到了N+1层之后，N+1层实体要在它前头加上一些附加的信息，即协议控制信息（PCI）。形成该层的一个协议数据单元（PDU)。协议控制信息的作用是为了协调对等实体之间的共同操作，即实现本层的对等协议。它所包含的内容是根据该层所要完成任务的不同而制定的，比如说，到了某一层之后，就可以在这个协议控制信息中加入源和目标地址，以此来识别发送方和接收方。所以我们经常说，PDU是被对等实体用来执行对等协议的，其实这主要还是协议控制信息的功劳。

- **接口数据单元IDU**：N+1层实体与N层通过服务访问点传递的数据单元。IDU就是由上层的协议数据单元和另外一些附加的信息组成。这些附加的信息就称为接口控制信息（ICI），它的作用是为了协调上层与下层实体之间的共同操作，并不是发送方给收方的内容，它只要完成接口上的数据传递之后，就不复存在了。
- **服务数据单元SDU**：从图中我们可以看出，到了下层之后，接口控制信息已经不复存在了，真正要跨越网络传给收方N层对等实体的接口数据单元的数据部分，即上层的协议数据单元。但在这个时候，上层的协议数据等于又被起了另一个名称，叫服务数据单元（SDU)。

服务原语（primitive)

在协议体系结构中，相邻层之间的服务在形式上是由一组原语（或操作）和参数来描述。服务原语定义的是所执行功能，即原语通知服务提供者执行某个动作，或者报告将某个对等实体的活动报告给用户，而参数则用于传递数据和控制信息。服务原语真正形式取决于它的实现方式，过程调用是服务原语的一个实例。 ISO的标准种使用了4种类型的服务原语来定义该体系结构种相邻层之间的交互作用，如表中所示。

原 语	含 义
请求(request)	由服务用户发出的服务原语将调用某些服务，并传递必要的参数，以完整地指明所请求的服务
指示(indication)	由服务提供者发出的服务原语，用以下两者之一： 1. 指示连接上的对等服务用户已调用了某规程，并且提供了相关的参数 2. 向服务用户通知一个提供者激活的动作
响应(response)	由服务用户发出的服务原语，用于应答或完成某些规程，这些规程在早些时候已经由向该用户发出的指示调用
证实(confirm)	由服务提供者发出的服务原语，用于应答或完成某些规程，这些规程在早些时候已经由服务用户的请求调用。



服务原语工作示意图

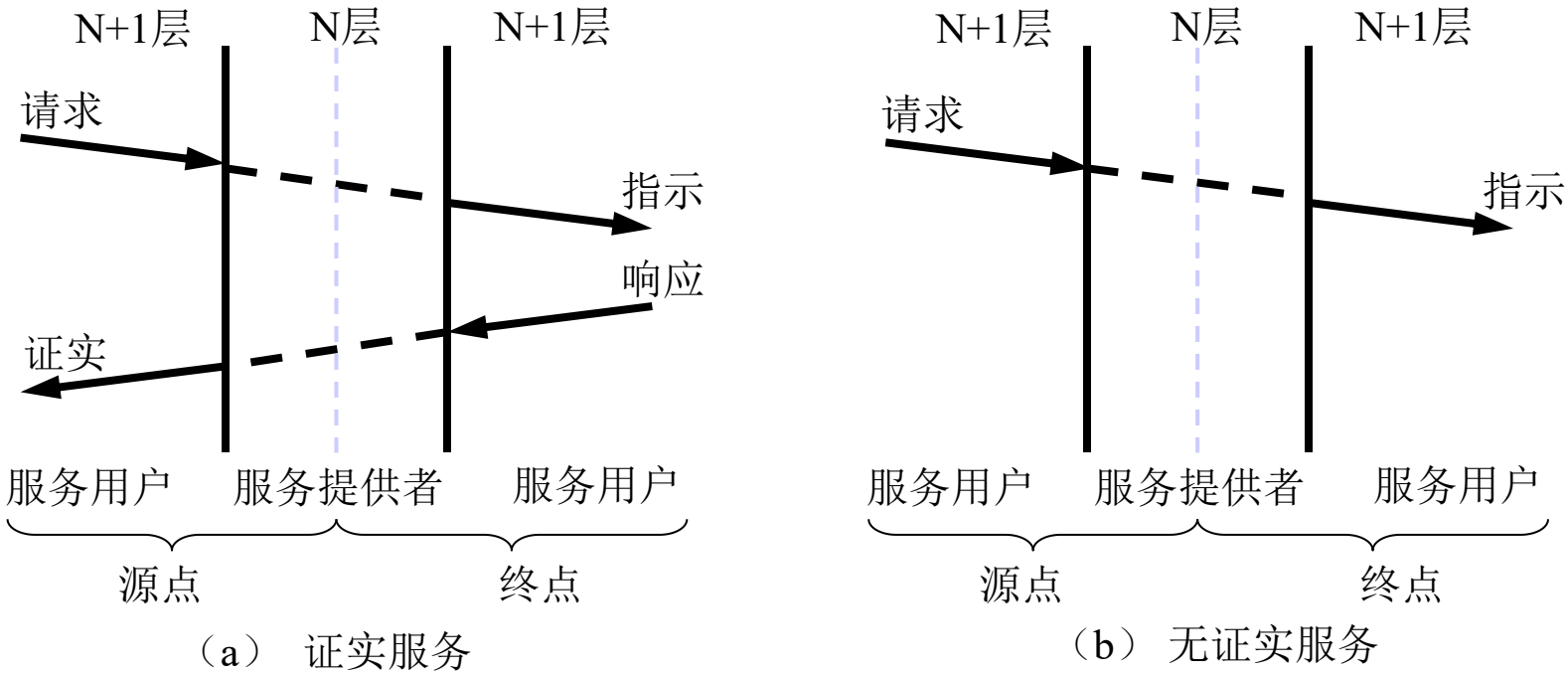
- 例如，设想从发送方（N+1）实体向另一个系统的对等（N+1）实体传送数据，步骤如下：
  - 源点的（N+1）的实体用请求服务原语调用它的（N）实体。与这个服务原语相关联的是必要的参数，比如传输的数据和终点地址等。
  - 源点的（N）实体准备一个（N）PDU发送到它的对等（N）实体。

- 终点的（N）实体通过指示服务原语把数据交付给终点上适当的（N+1）实体，其参数包括数据和源点地址。
- 如果要求确认，那么终点的（N+1）实体向它的（N）实体发送一个响应服务原语。
- 终点的（N）实体用一个（N）PDU来运送这个确认给源点（N）实体。
- 这个确认在源点以证实服务原语的形式交给（N+1）实体。

证实型（confirmed）和无证实型（unconfirmed）服务的区别：

- 证实型服务包含所有4种服务原语。
- 无证实型服务只包含请求和指示2种服务原语。

图中描述了事件发生的事件顺序。图（a） 事件序列称为证实服务，因为发起者会收到一个证实，表明它所请求的服务在另一端得到了预期的效果。如果仅涉及请求和指示这两个服务原语，则这个服务的对话就是无证实服务，发起者不会收到表明其请求的动作已发生的证实信息，如图（b）所示。



服务原语时间序列

1.1.3 协议

网络协议 (network protocol)，简称为**协议**，是为进行网络中的数据交换而建立的规则、标准或约定。这些**规则**明确规定了所交换的数据的**格式**以及有关**同步**的问题（同步含有时序的意思）。

英格兰国家物理实验室NPL(National Physical Laboratory)的R.A. Scantlebury和K.A.Bartlett在一份备忘录(A **protocol** for use in the NPL data communications network)中最早将“协议(protocol)”一词用于描述数据通信过程。

备忘录将协议定义为 **“协议是关于分布式系统进行信息交换时的一种约定，协议应按照语言的方式进行定义。”**

协议的定义类似于**语言**的定义，从语言的角度，网络协议主要有三个要素：**语法**、**语义**和**同步**。

协议定义：语法

- 在语言学中，语法是指语言组成成分(词、句等)之间以什么样的关系结合而构成语句或语言，也即是语言的结构方法和构成规则。
- 网络协议中，可以将交换的报文（即OSI体系中的协议数据单元PDU)分为两种：用于传输用户数据的数据报文和用于协议控制的控制报文。
- 网络协议中的语法的含义

网络协议中的**语法体现为数据报文中的控制信息(通常在报文的首部)和各种控制报文的结构、格式**，也即是规定报文的长度，报文中划分多少个域(Field)，每个域的名称、意义、数据类型、长度等。其中，报文中各个域的类型、长度及相互间的位置、顺序关系则构成了词法。

协议定义：语义

- 在语言学中，语义是指语言组成成分（词、句等）的含义。
- 网络协议的语义可以理解为协议数据报文中的控制信息和控制报文中各个域所约定的含义，即需要发出何种控制信息，完成何种动作以及做出何种响应。
- 例如：
  - 报文首部控制信息中的目的地址信息指明了报文的目的地，接收到此报文的网络结点均将其作为进行路由选择的依据，因而规定在首部控制信息中在给定域给出目标结点地址就是一种语义。

协议定义：语义（Cont.）

- 例如（续）：
  - 连接的建立：为了实现有连接的传输服务，设计了一套实现连接的控制报文。发起连接方构造一个请求连接的协议控制报文，这个“请求连接”就是该控制报文的语义。收端收到这个控制报文后，根据已知的格式分析规定域中报文类型码就可了解这个“请求连接”的语义，从而作出“允许连接”或“拒绝连接”的响应。“允许连接”报文和“拒绝连接”报文的格式和语义也是协议中约定好的，通信双方通过这些约定语义的控制报文按一定时序关系的交换即可实现建立连接的功能和提供有连接服务。
  - 其它的在协议中常使用的“数据收到确认”报文(ACK报文)、“出错通知”报文、“紧急通知”报文等等均可看成为具有特定语义的网络协议的语句，即协议的组成部分。

协议定义：同步

- 简单地说，同步是指事件实现顺序的详细说明。
  - 具体来说，同步是指通信过程中各种控制报文传送的顺序关系，例如“允许连接”或“拒绝连接”报文必须是作为请求连接报文的一种响应来发送，“释放连接”报文也必须在建立连接后的某种条件下发送等等。
  - 这种控制报文发送的时序关系，也决定了通信双方所处的通信状态(发送状态、接收状态、等待状态等)的制约关系，所以常用通信双方的有限状态机的方法来描述网络协议。

1.1.4 OSI与TCP/IP参考模型

1. ISO-OSI参考模型

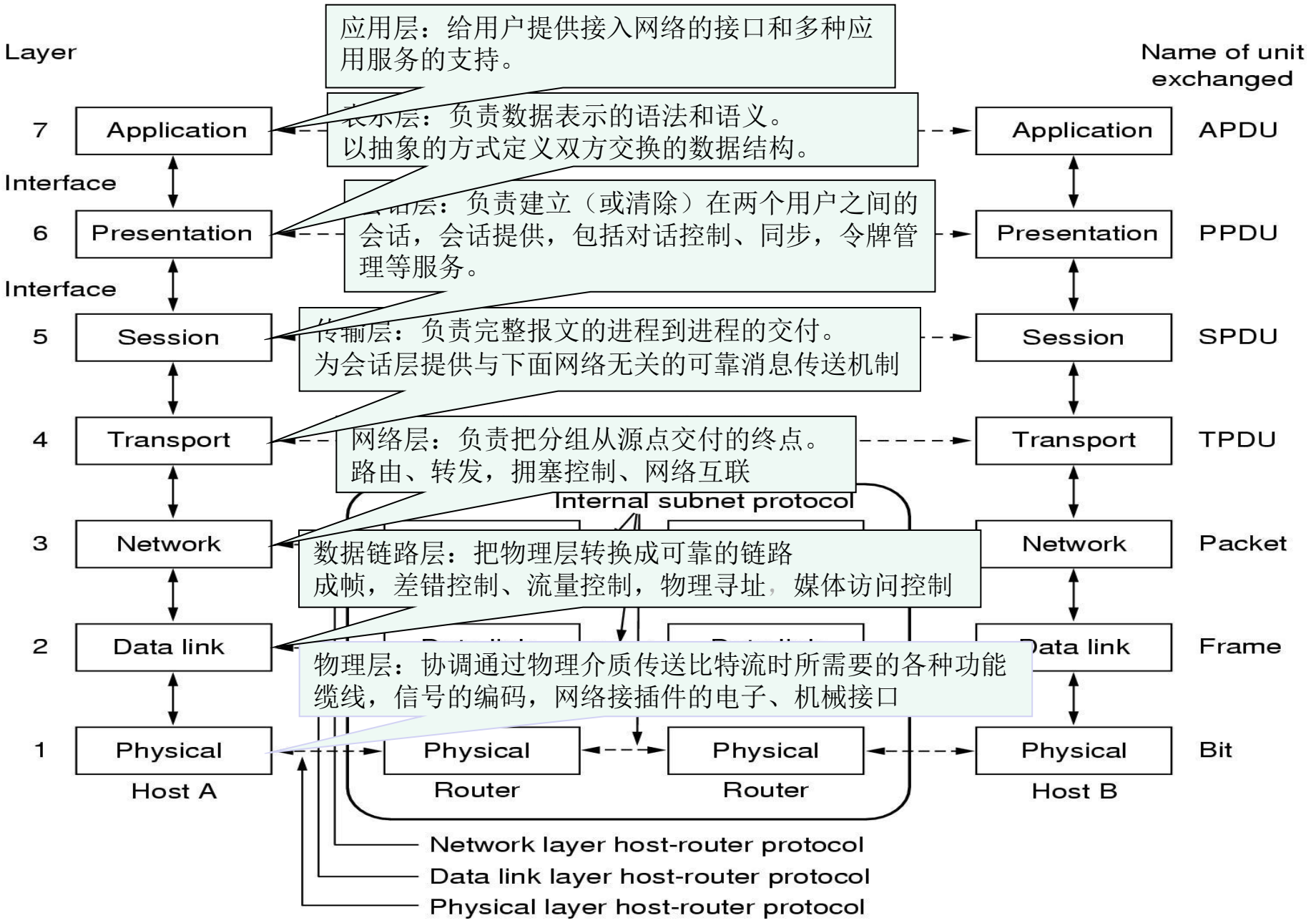
- 在对网络技术和理论进行一系列研究之后，各大计算机公司纷纷制定了自己的网络技术标准。其中比较知名的有，1974年，IBM公司首先提出了系统网络体系结构（System Network Architecture，SNA）标准，1975年，DEC公司也公布了数字网络体系结构（Digital Network Architecture，DNA）标准。这些标准均采用层次结构，但在层次的划分、各层协议等问题上不相同，使得采用不同体系结构的网络很难互联。

- 针对这一问题，国际标准化组织（International Organization for Standardization，ISO）于1984年提出了著名的Open System Interconnection/Reference Model，开放系统互连参考模型，简称OSI/RM。
- 开放系统OSI标准定制过程中所采用的方法是将整个庞大而复杂的问题划分为若干个容易处理的小问题，这就是分层的体系结构方法。在OSI中，采用了三级抽象，即**体系结构、服务定义和协议规范（协议规格说明）**。

- **体系结构**：OSI参考模型定义了开放系统的层次结构、层次之间的相互关系及各层所包含的可能的服务。它是作为一个框架来协调和组织各层协议的制定，也是对网络内部结构最精炼的概括与描述。
- **服务定义**：OSI的服务定义详细说明了各层所提供的服务。某一层的服务就是该层及其下各层的功能，它通过接口提供给更高一层。各层所提供的服务与这些服务是怎么实现的无关。同时，各种服务定义还定义了层与层之间的接口和各层的所使用的原语，但是不涉及接口的具体调用方法。
- **协议规范**：OSI标准中的各种协议精确定义了应当发送什么样的控制信息，以及应当用什么样的过程来解释这个控制信息。

- ISO/OSI参考模型不是一个标准，它描述了一些概念，是一个在制定标准时所使用的概念性的框架。OSI参考模型用来协调进程间通信标准的制定，没有定义具体的实现方法。在OSI范围内，各种网络中所使用的协议只要与OSI的协议规范相一致就能实现网络互联。

ISO-OSI/RM


Layer
Name of unit exchanged
7 Application
Application APDU
应用层：给用户提供接入网络的接口和多种应用服务的支持。
表示层：负责数据表示的语法和语义。
以抽象的方式定义双方交换的数据结构。
Interface
6 Presentation
Presentation PPDU
会话层：负责建立（或清除）在两个用户之间的会话，会话提供，包括对话控制、同步，令牌管理等服务。
Interface
5 Session
Session SPDU
传输层：负责完整报文的进程到进程的交付。
为会话层提供与下面网络无关的可靠消息传送机制
4 Transport
Transport TPDU
网络层：负责把分组从源点交付的终点。
路由、转发，拥塞控制、网络互联
Internal subnet protocol
3 Network
Network Packet
数据链路层：把物理层转换成可靠的链路
成帧，差错控制、流量控制，物理寻址，媒体访问控制
2 Data link
Data link Frame
物理层：协调通过物理介质传送比特流时所需要的各种功能
缆线，信号的编码，网络接插件的电子、机械接口
1 Physical
Physical
Physical
Physical Bit
Host A
Router
Router
Host B
Network layer host-router protocol
Data link layer host-router protocol
Physical layer host-router protocol

2. TCP/IP参考模型

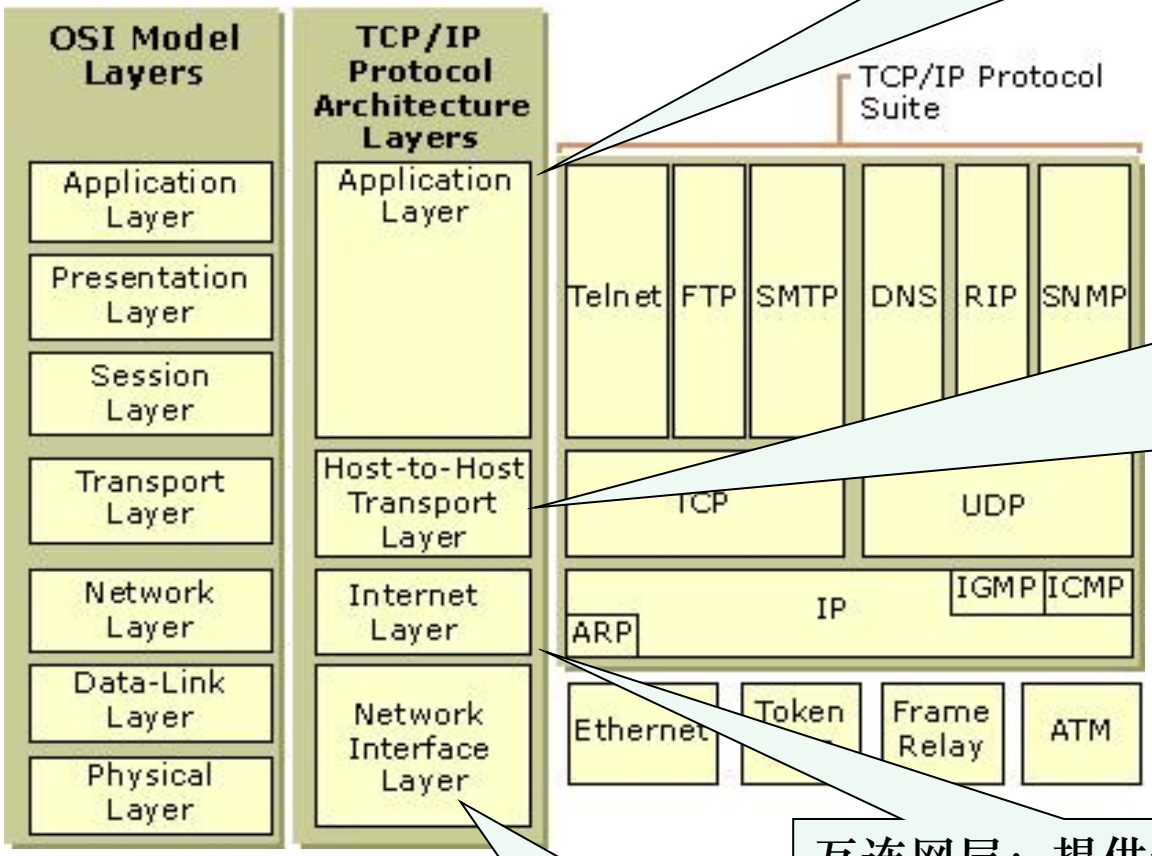
TCP/IP参考模型（Transmission Control Protocol/Internet Protocol Reference Model）TCP/IP参考模型是计算机网络的鼻祖ARPANET和其后继的互联网使用的参考模型。

两个设计目标：

- 实现网络互联；
- 残存性强，即使中间机器和线路失去控制，只要源和目的机器在工作通信就不会中断。也就是说网络不受子网硬件损失的影响，已经建立的会话不会被取消。-动态路由和网状拓扑

TCP/IP协议族由应用层、传输层、互联网层、网络接口层构成。每一层的功能由一个或多个协议实现。

TCP/IP协议族（Transmission Control Protocol/Internet Protocol Suite）是一个在Internet使用的协议系列，TCP/IP协议族包含了大量由Internet体系结构委员会（Internet Architecture Board, IAB）作为Internet标准发布的协议。



应用层：向用户提供应用服务。如FTP，TELNET等

传输层：提供可靠的面向连接的传输层协议TCP和不可靠的非连接传输层协议UDP。

互连网层：提供无连接的分组交换功能

主机至网络层（或网络接口层），在TCP/IP模型中很少提及。

OSI与TCP/IP参考模型的比较

相同点：

1.两者都以协议栈的概念为基础，并且协议栈中的协议彼此互相独立。

2.两个模型中的各个层的功能大致相似，如两者都有功能相似的应用层、传输层、网络层。

不同点：

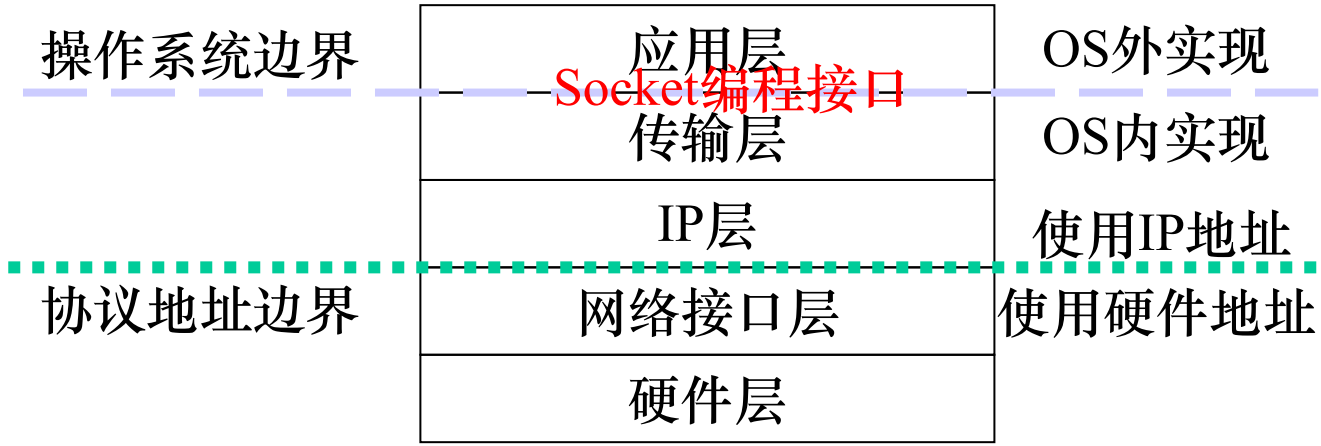
1.在OSI模型中，严格地定义了服务、接口、协议；在TCP/IP模型中，并没有严格区分服务、接口与协议。

2.OSI模型更具通用性，TCP/IP模型一点也不通用。

3.OSI模型支持无连接和面向连接的网络层通信，但在传输层只支持面向连接的通信；TCP/IP模型只支持无连接的网络层通信，但在传输层有支持无连接和面向连接的两种协议可供用户选择。

4.TCP/IP模型中不区分、甚至不提起物理层和数据链路层。

3. TCP/IP分层模型中的两个边界



传输层及其以下的各层协议，应用层中被IETF标准化的协议和操作系统厂商自定义的协议属于操作系统内部实现。用户自己开发的应用程序所制定的相应协议属于操作系统外部实现。

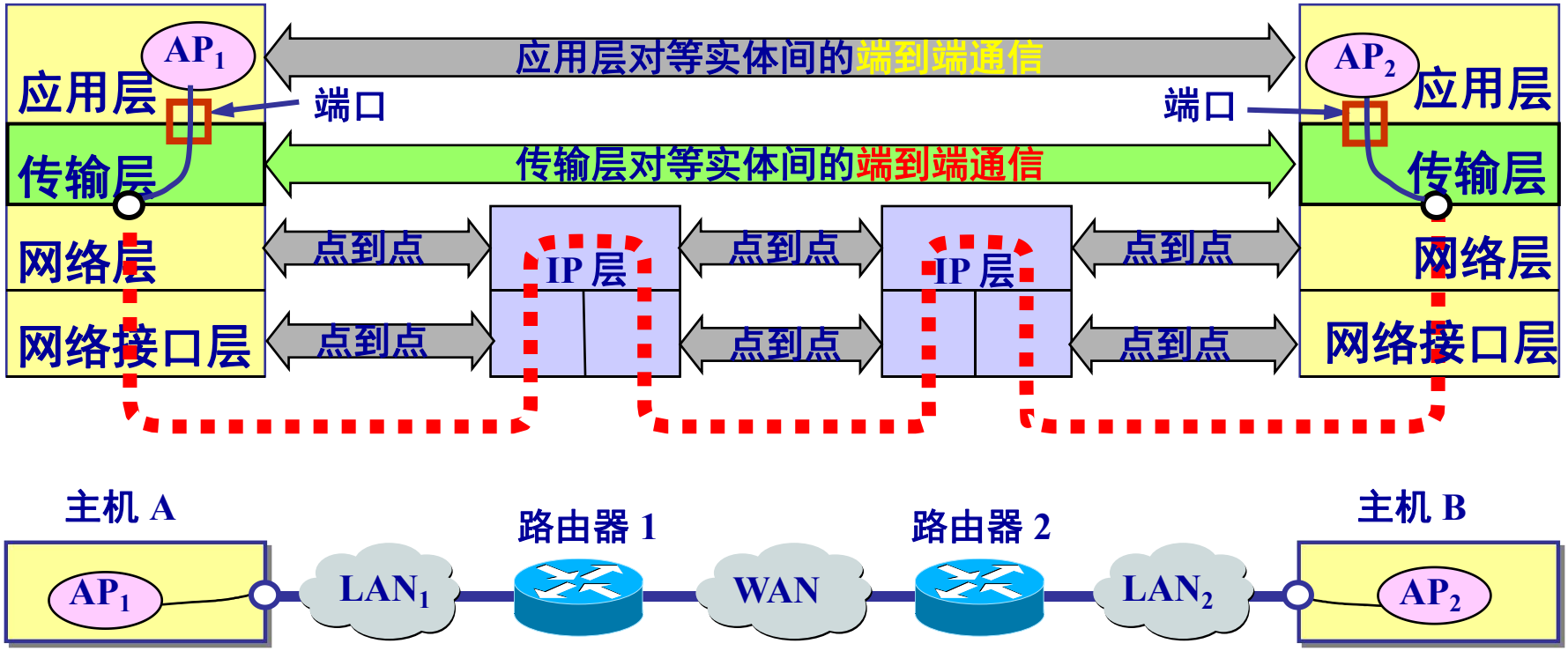
编写网络应用程序时如何访问操作系统内部的服务？？

SOKET是连接网络应用与操作系统的一个常用接口。

问题：如何实现IP地址和到物理地址的映射?

解答：使用ARP和RARP协议

4. 点到点和端到端



TCP/IP分层模型中点到点和端到端概念示意

主机A发出的数据在自己的协议栈中逐层向下递交并通过LAN1投递给R1，R1收到数据后，提取其中包含的IP数据报选路后沿着自己的协议栈逐层向下递交，并通过WAN转发给R2，R2经过与R1相似的过程，经过LAN2转发给主机B，主机2沿着自己的协议栈逐层向上递交，最后到达相应的应用层。

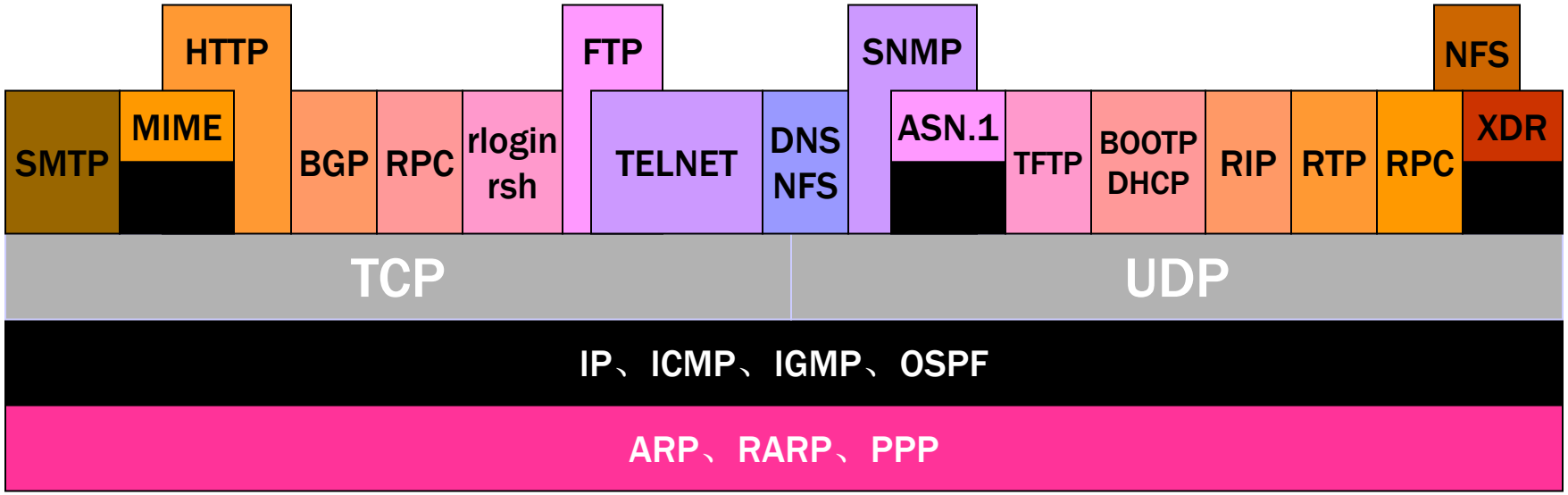
在整个过程中，主机1的应用层和传输层看到的就是其通信对等端（主机2）的相应层，而主机1、R1、R2和主机2的IP层、网络接口层看到的都是与自己直连的下一跳设备的相应层。从这一点看，我们称应用层和传输层是端到端的，而其下各层是点到点的。

点到点：指对等实体间的通信由一段直接相连的机器间的通信组成。

端到端：指对等实体间的通信向拥有一条直接线路，而不管中间要经过多少通信节点。

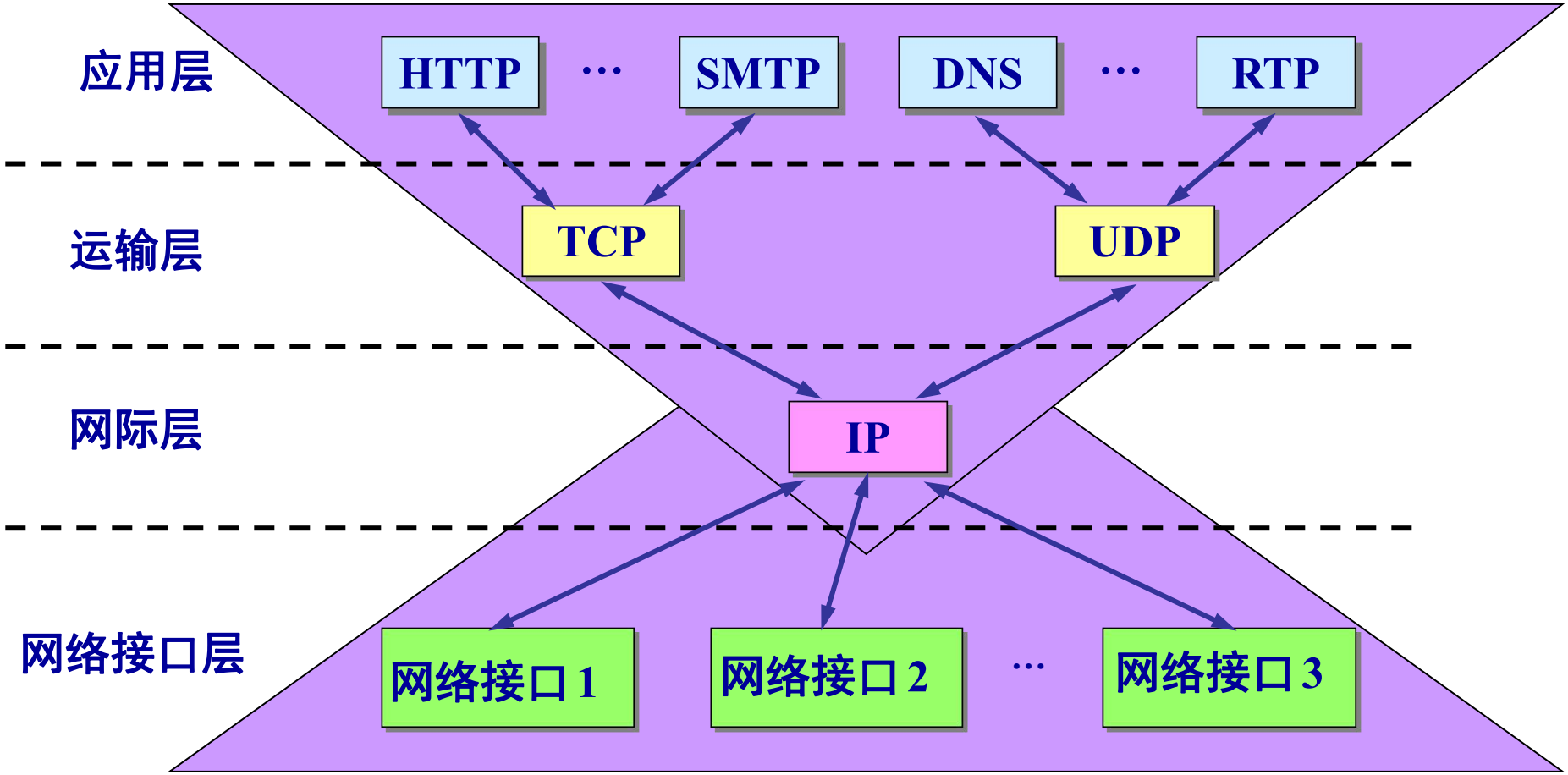
5. 协议依赖关系

整个TCP/IP协议族包含了很多协议，这些协议之间存在相互依赖关系，其中DNS比较特殊，它既依赖TCP，也依赖UDP。此外，OSPF用于路由信息维护，它的报文直接封装在IP数据报中投递，所以把它列在1P层。



TCP/IP协议族中各协议的依赖关系

IP over Everything
IP可应用到各式各样的网络上



沙漏计时器形状的 TCP/IP 协议族

IP位于所有通信的中心，是唯一被所有应用程序所共用的协议，从功能上看，IP屏蔽了底层物理网络之间的差异，向上提供一种一致的服务，实现了不同物理网络之间的互通，是Internet的核心技术。

6. 多路复用和多路分解

协议依赖关系体现了TCP/IP协议栈的重要思想：多路复用和多路分解

- 多路复用：发送数据时，各类基于TCP的应用都会把数据封装在TCP报文段中，基于UDP的应用则把数据封装在UDP数据报中；TCP报文段、UDP数据报和ICMP报文都要封装在IP数据报中；IP数据报、ARP报文等则封装在物理帧中。
- 多路分解：当目的主机收到一个以太网数据帧时，数据就开始从协议栈中由底向上升，数据链路层以首部中的帧类型字段进行分用，网络层以IP首部中的协议字段的值进行分用，运输层以TCP或UDP首部中的目的端口号字段进行分用。

1.2 网际互连与TCP/IP

Internet是由各种不同的网络构成的，这些网络在信道的访问方式和数据的传送方式上存在差异。

出现各种不同网络是因为没有一种单一的网络硬件技术可以满足所有的要求。而用户又期待一种通用的互联。

网络互联的目的就是要隐藏底层网络硬件的细节，同时提供一般的服务通信。

例如：以太网和点对点网络（使用PPP协议）通信时使用不同的帧格式和物理地址。

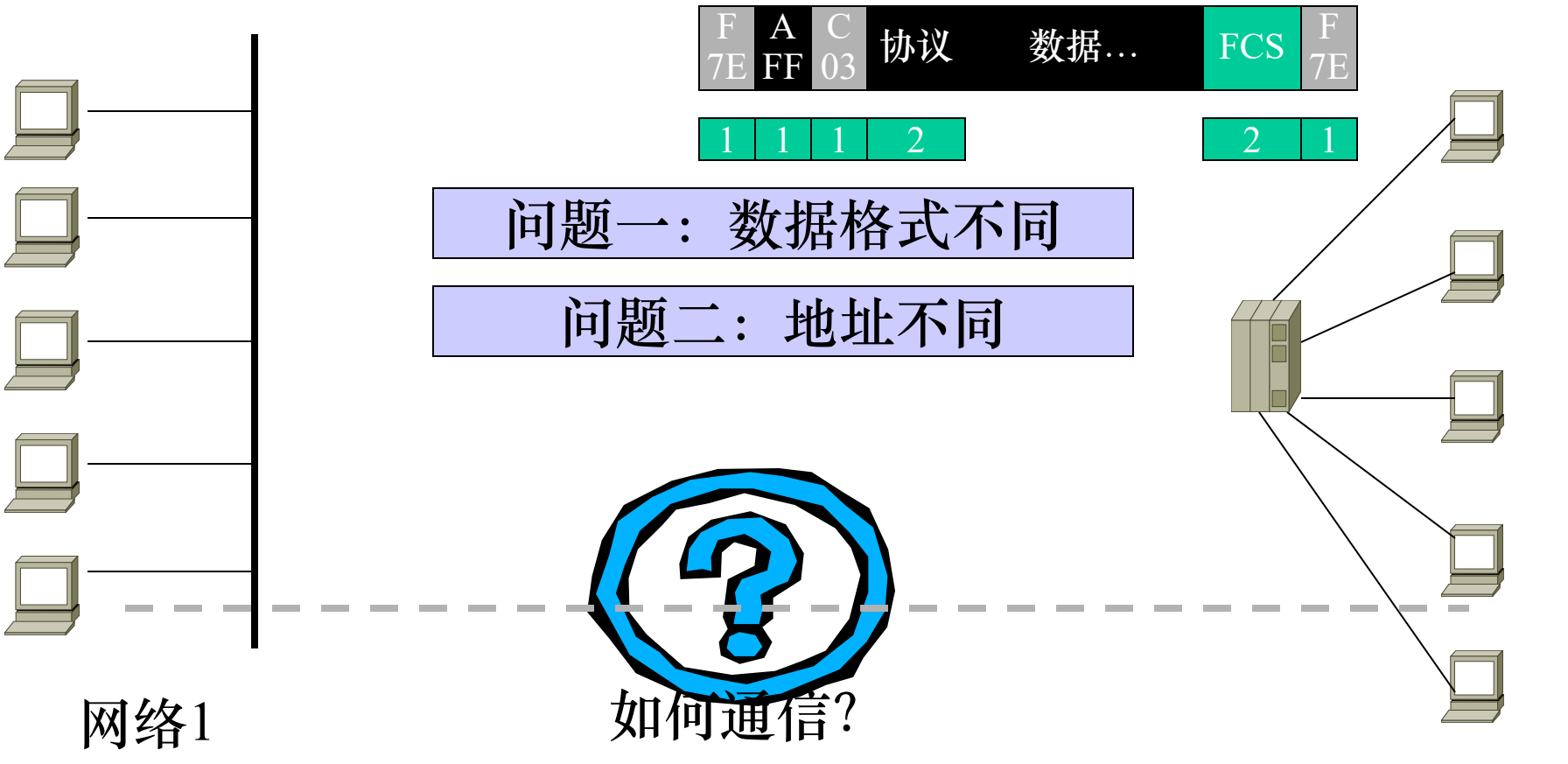
Vinton Cerf提出网络互联的技术思路:

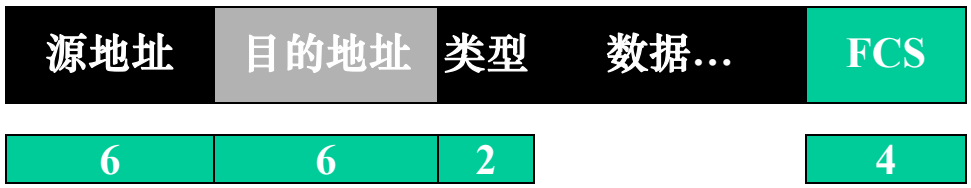
在每个网络内部使用各自的通信协议，每个网络与其他网络通信时使用TCP/IP协议族。

- **Cerf 的Kahn的网络互联原则:**
  - 最低限度的，自治的-不因网络互联而改变各网络的内部（黑盒子设计）-minimalism, autonomy - no internal changes required to interconnect networks ，
    - 每个网络既可自行运行也可网间互联
  - 尽力而为的服务模型-best effort service model
    - 丢弃的分组被重传而实现误码恢复
  - 无状态路由器-stateless routers-任何类型的网络可通过网关连接到另一个其他类型网络
    - 不保存任何经过的数据信息
  - 分布式控制-decentralized control
    - 没有集中的网络管理和控制
- 定义了今天的互联网体系结构

TCP/IP协议是Internet最基本的协议。在Internet没有形成之前，世界各地已经建立了很多小型网络，但这些网络存在不同的网络结构和数据传输规则，要将它们连接起来互相通信，就好比要让使用不同语言的人们交流一样，需要建立一种大家都听得懂的语言，而TCP/IP就能实现这个功能，它就好比Internet上的“世界语”。

1.2.1 用IP实现异构网络互联





解决：引入IP层，屏蔽不同物理网络技术的差别。

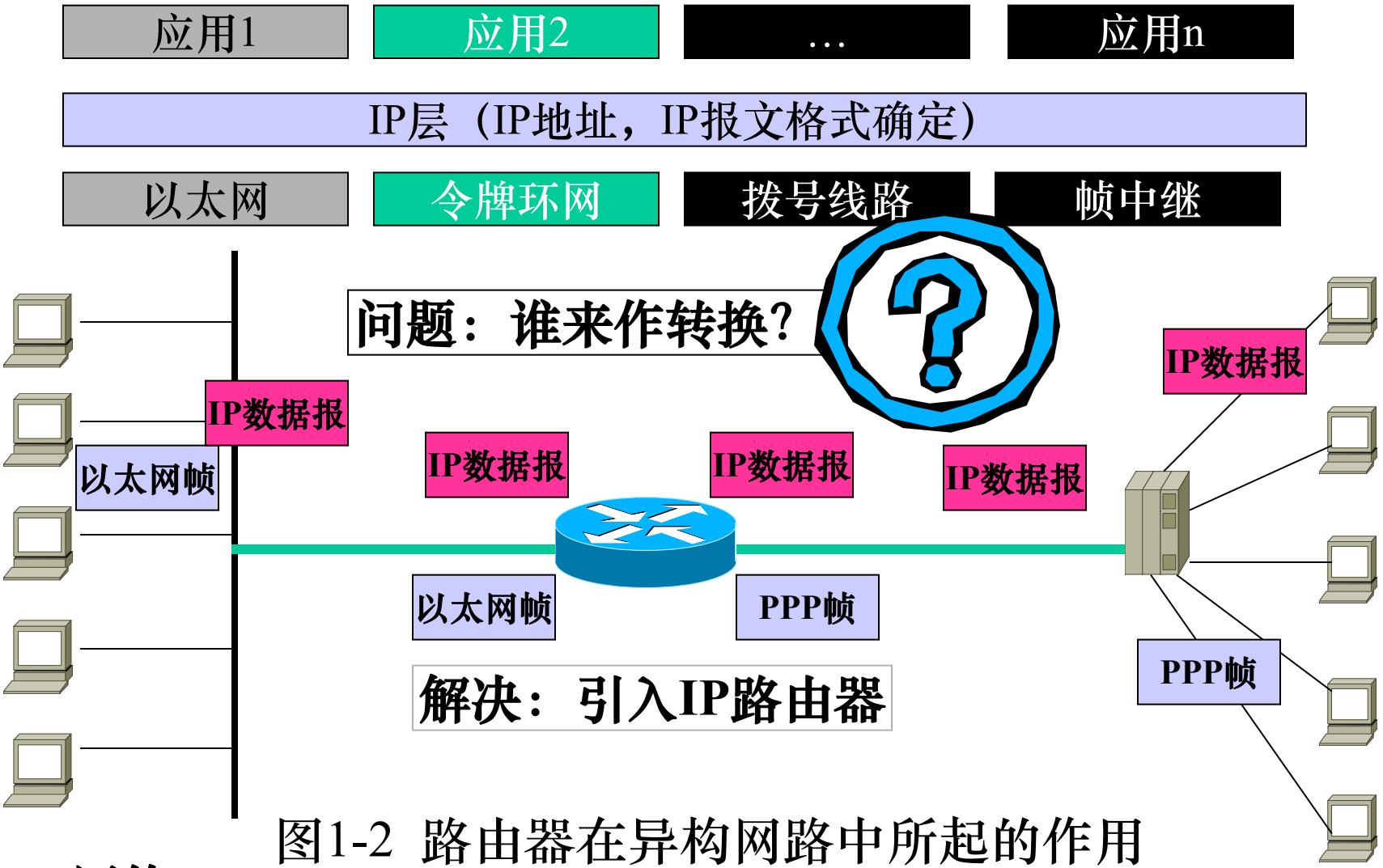
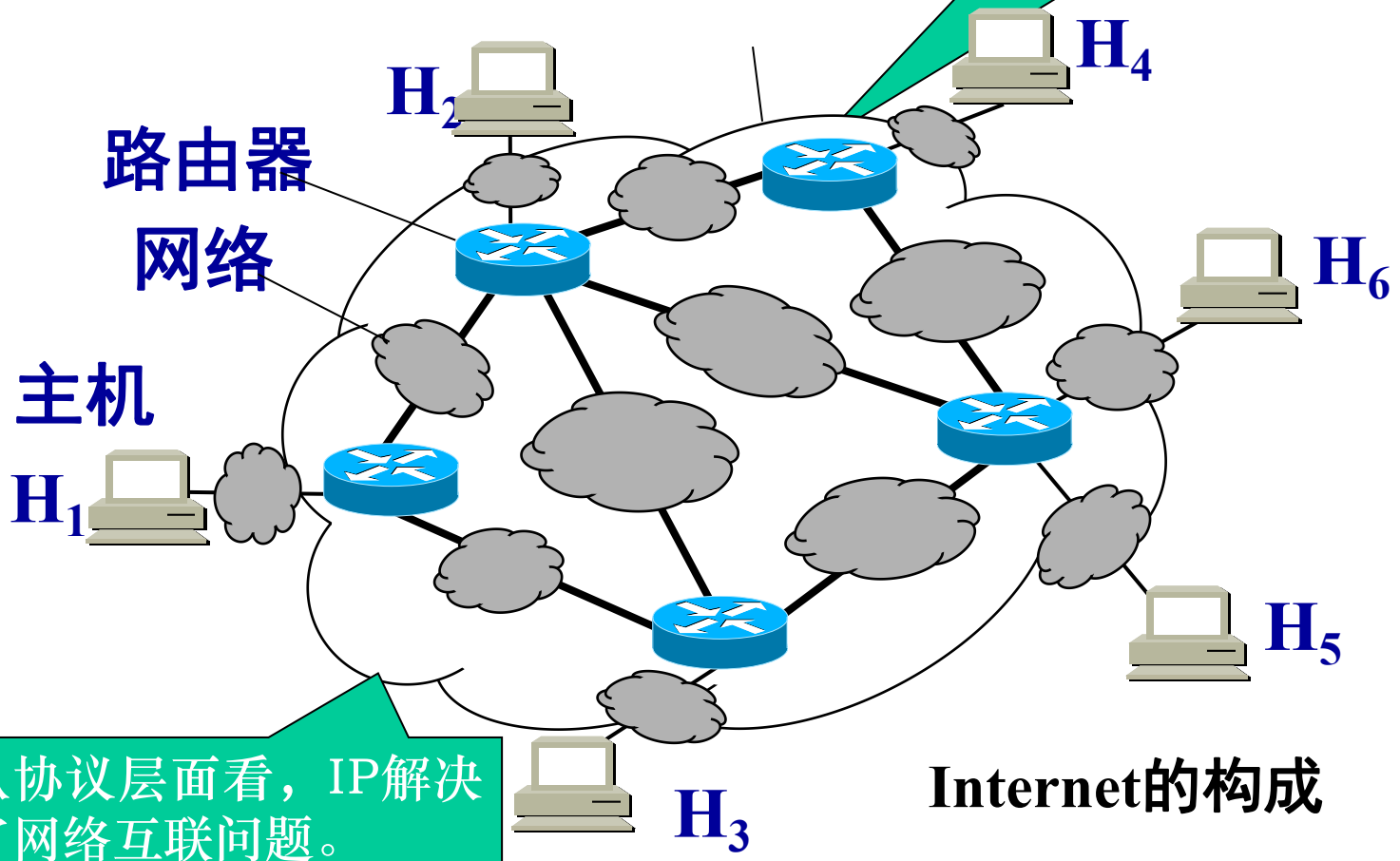


图1-2 路由器在异构网路中所起的作用

Internet的物理体系结构

用路由器连接起来的多个物理网络



除了实现异构网络互联外，路由器的另一个重要的功能是在其所连接的多个网络之间转发IP数据报。

1.2.2 TCP/IP协议族的引入

遗漏的问题及解决方案：

- **选路问题**：要正确转发数据报，所有路由器对这个复杂系统的认识必须一致，路由协议（RIP、OSPF、BGP等）解决了这个问题。
- **可靠传输问题**：传输线路存在差错、丢失、乱序等问题，传输控制协议TCP用于提高可靠性。
- **网络控制问题**：复杂系统的流量、拥塞、服务质量等的控制，Internet控制报文协议（ICMP）、TCP等协议实现。

上述协议在解决了网络基础设施正常高效运转的问题后。就可以基于这个基础设施构建各种应用。

FTP、HTTP、TELNET、POP3、SMTP、SNMP、DNS等等

1.3 协议描述与验证

1.3.1 协议工程的概念

- **协议工程：集成化(Integrated)、形式化(Formal)的协议开发过程称为协议工程。**“集成化”，就是指协议开发的各个阶段前后衔接，并在同一个开发系统中完成。协议开发包括如下几个阶段：
  - 协议说明（Protocol Specification）
  - 协议验证（Protocol Verification）
  - 协议实现（Protocol Implementation）
  - 协议测试（Protocol Testing）
    - 一致性测试（Conformance Testing）
    - 互操作性测试（Interoperability Testing）
    - 性能测试（Performance Testing）

- **协议说明**
  - 必须既定义一个协议实体提供给它的用户的服务，又定义该协议实体的内部操作。
- **协议验证**
  - 验证协议说明是否完整、正确。
- **协议实现**
  - 用硬件和/或软件实现协议说明中规定的功能。
- **协议测试**
  - 用测试的方法来检查协议实现是否满足要求，包括：协议实现是否与协议说明一致（一致性测试）、协议实现之间的互操作能力（互操作性测试）和协议实现的性能（性能测试）等。
- 在协议的说明、验证、实现和测试过程中使用形式化描述技术，不仅可以比较容易地理解协议，而且可以使协议描述更加精确，大大简化了协议的研究工作。

协议验证

- 验证协议说明是否完整正确，以协议描述为基础，涉及逻辑证明。
- 主要用于系统实现前的设计阶段，为了避免可能出现的设计错误。原则上验证涉及协议所有可能的状态。
- 可达性分析是一种常用的验证方法，可达性分析能够发现协议描述中的各种错误。
  - 利用图论知识可以解决状态的可达性问题；
  - 可达性分析能够用来解决协议的不完整性、死锁和无关变迁（多余的转换）等问题。

不完全：如果每类帧在某一状态出现是可能的，而有限状态机并没有说应采取什么行动，描述不完全。

死锁：如果存在某个状态集，从这些状态没有出口，也没有完成任何进展（收到正确的帧）。

多余的转换：在一种状态下如何控制一个事件，而这个事件永远不可能发生。

- **形式化描述的意义**
  - 实际使用的协议非常复杂，给协议的理解、验证、实现和测试等工作带来困难，需要采用形式化的、数学的描述方法来描述协议。但是目前大多数协议还是采用自然语言描述。
- **自然语言描述协议的缺点**
  - 冗余；
  - 多义性；
  - 结构性不好；
  - 不便于自动验证、测试、实现。
- **形式化描述技术FDT（Formal Description Technique）/形式化方法FM（Formal Method）广泛应用于协议工程研究中**

- **一种形式化方法总是以一种形式体系为基础，只是在具体应用时，大都做了便于描述的改进和扩充。**
- **常用的形式化方法**
  - 有限状态机FSM（Finite State Machine）
    - 扩展：EFSM
  - 形式化语言模型
    - LOTOS，Estelle，SDL
    - 都有相应扩展
  - Petri网
    - 扩展：时间Petri网，随机Petri网，高级Petri网
  - 过程代数（Process Algebra）
    - 扩展：随机过程代数

1.3.2 有限状态机模型（Finite State Machine）

有限状态机是许多协议模型的一个关键概念。其协议机（发送方或接收方）在每一个时刻总是处在一个特定的状态，其状态是由所有变量值组成的，包括程序计数器在内。

定义：系统被描述成有限的状态，在一定的前提条件下会发生一系列的输入事件，这些事件使得系统采取相应的动作，并从一个状态转换成另一个状态，我们称为状态的变迁或转换（transition）。

各状态总是选择在协议机等待下一个事件发生的时刻。协议机的状态由其变量的状态决定，状态数则有2^n个，这里的n是表示所有状态变量所需位数。

从每个状态，有0或多个可能的**转换**到其他状态。当某些事件发生了，就会发生转换，对于一个协议机而言，当发送一帧，接收一帧，计时器超时，产生一个中断等，都可以引起转换。对于信道而言，典型的事件是协议机将一个新帧插入信道，传输一帧到协议机，或由于突发噪声而丢失帧。

从初始状态通过一系列转换，可以到达某些状态，或许是全部状态。利用图论中可传递封闭包就可以确定哪些状态是可达的，哪些是不可达的。这项技术称做**可行性分析**。

有限状态机可以把协议形式化为一个四元组模型(S, M, I, T)，其中：

S是进程和信道可能进入的状态集合，

M是在信道上进行交换的帧的集合，

I是进程初始状态的集合，

T是两两状态之间转换的集合。

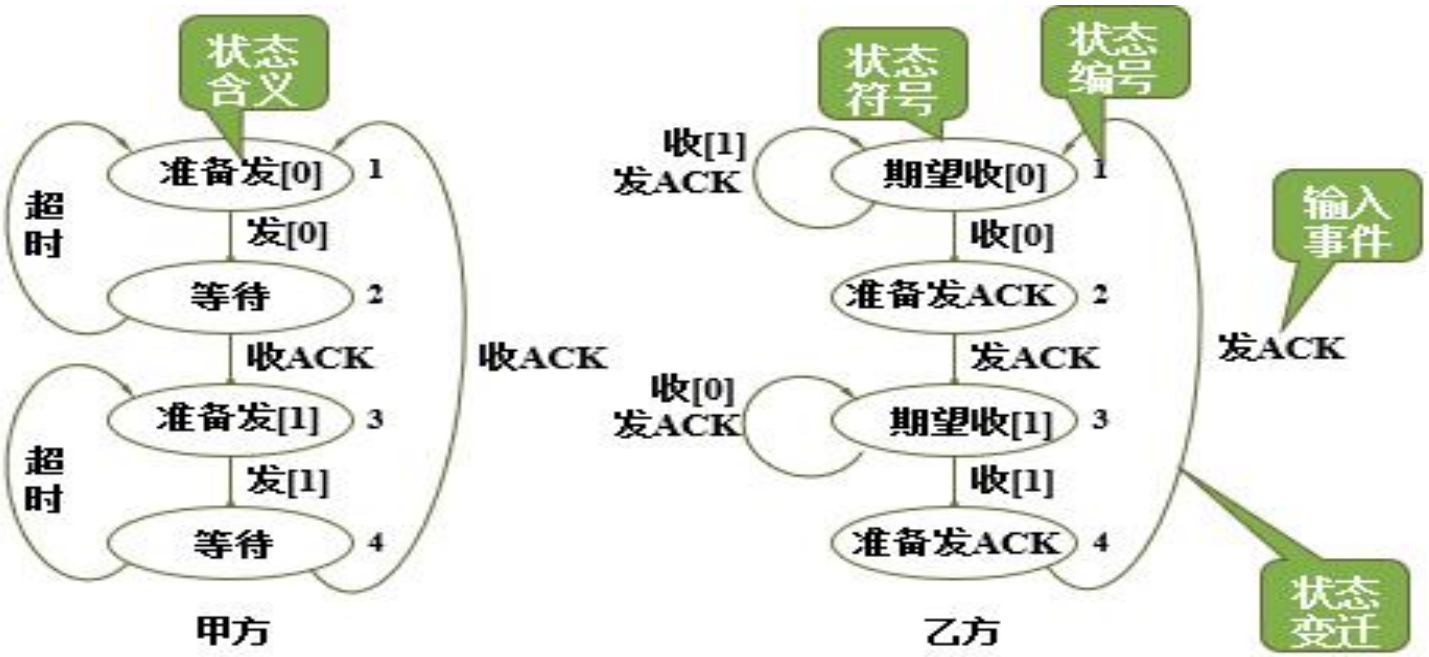
起初，所有的进程都处于初始状态。然后事件开始发生，诸如有帧可供传输，或计时器到点。每个事件可能引起一个进程或信道采取行动，从而转换成新的状态，通过仔细计算每个状态的每个可能的后继，就能建造出可达性图，并且分析这个协议。

- **通信协议建模**
  - 基本出发点：认为通信协议主要是由响应多个“事件”的相对简单的处理过程组成；
  - 事件
    - 命令（来自用户）
    - 信息到达（来自低层）
    - 内部超时
  - 优点：简单明了，比较精确；
  - 缺点：对许多复杂的协议，事件数和状态数会剧增，处理困难。

1.3.3 停等协议的有限状态机

（1）半双工信道的停止等待协议

设通信双方进行半双工通信，发送方发送信息帧，接收方回送确认帧。双方约定采用停等协议，因此发送方仅需1比特来编号，下面将0号帧和1号帧分别记为帧0和帧1。当收到有差错的帧时，则丢弃此帧，同时不发送任何应答帧。当收到序号不正确但无差错的帧时，要发送确认帧，同时丢弃此帧，不发送给主机。我们还假定接收方在准备发送确认帧ACK时，暂不接收外面发来的帧。这样就得到甲乙双方各自的有限状态机。



图中椭圆表示状态符号，其右方数字为状态标号，椭圆内的字表示状态的意义。带箭头的直线或弧线表示状态的变迁，而直线或弧线旁边的字代表协议机的输入事件，例如，甲方协议机中的“发帧0”就是一个输入事件。图中在某些方面进行了一些简化。例如，当乙方处于“期望收帧0”的状态时，若收到无差错的帧1帧，仍然应当先进入“准备发ACK”状态，然后才发出ACK。但这里就将“收帧1”与“发ACK”合并成为一个事件，其余部分都很清楚，此处不再解释。

发送方和接收方各自的有限状态机实际上并不是孤立地工作而是协调在一起工作的。如果将双方各自的有限状态机合在一起，用一个有限状态机来描述整个系统。在这种情况下，状态数目将大大增加。这是因为发送方和接收方的状态各有4个，而信道上的状态也有4个（0，1，A或 -/ S（ 空）），这样就共有4^3=64种状态，用状态图很难清楚的表示出来。

比较实用的办法是合并一些状态，即不考虑一些次要的细节。例如发送方的状态1和状态2，状态3和状态4可以合并。接收方的状态1和状态4，状态2和状态3也可以合并。整个系统的状态是通信双方两个协议和信道的所有状态的组合，信道状态由其内容决定的，例如，如果帧0正在信道上（即帧已被部分地发送出去，部分地被接收，但在接收方主机上还未处理），则信道的状态为0。

- 系统状态：（发送方状态、接收方状态、信道状态）

  每个状态用三个字母表示：XYZ

  X：发送方正发送的帧序号，为0或1；

  Y：接收方正等待的帧序号，为0或1；

  Z：信道状态，为0，1，A或 -/ S（空）。

  初始状态为（000）

- 发送方状态由发送方发送帧号状态标志位的取值
  - 0——对应于发送了0号帧
  - 1——对应于发送了1号帧
- 接收方状态由期待帧号状态标志位的取值
  - 0——对应于期待接收0号帧
  - 1——对应于期待接收1号帧

- 信道状态则有四种

  0——表示信道上有0号帧

  1——表示信道上有1号帧

  A——表示信道上有Ack帧

  S——表示信道上没有任何帧，即处于空状态

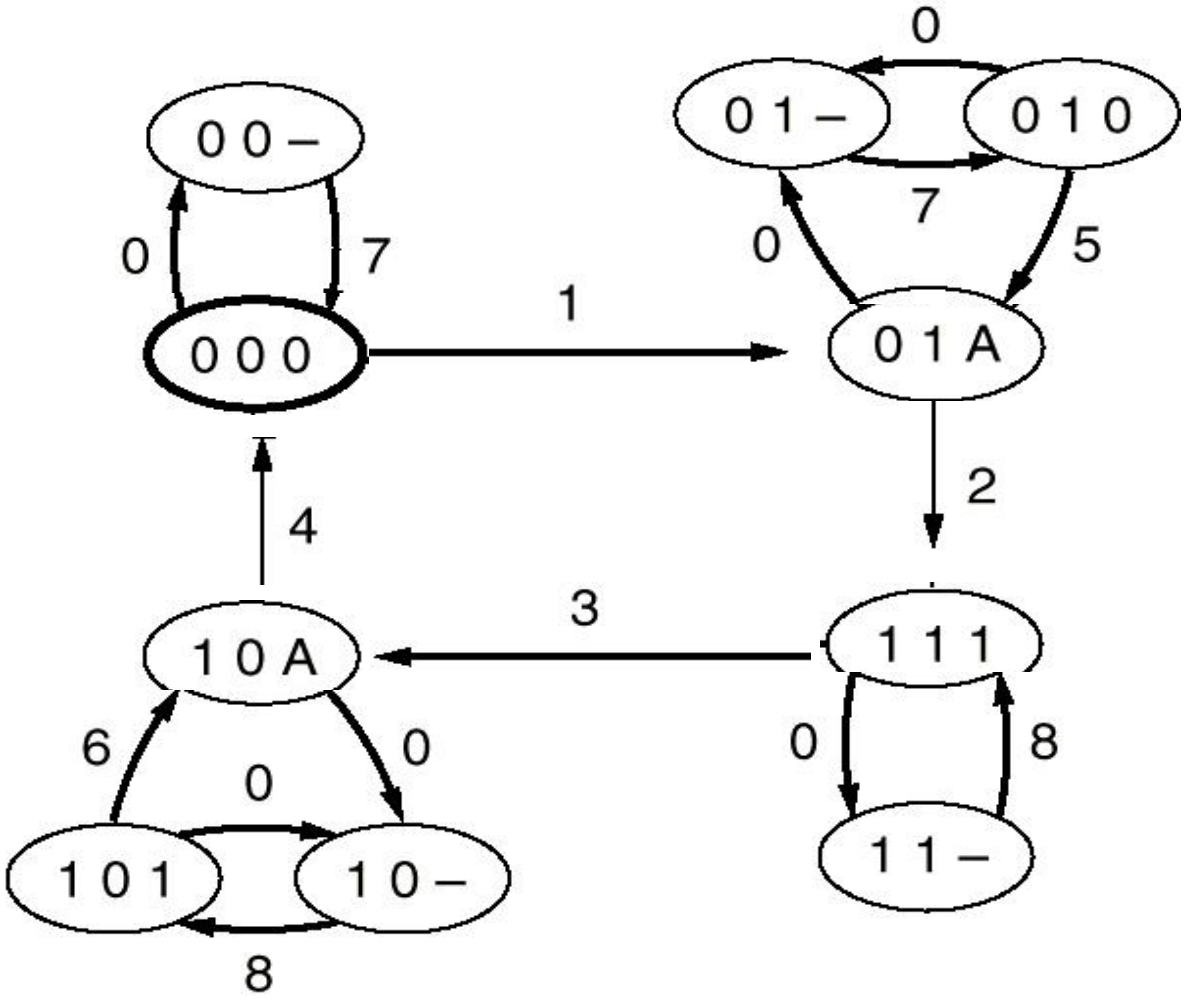


图 停止等待协议状态图

假设系统一开始处在（000）状态。这表示发送方发完帧0，接收方期望收到帧0，而信道上传送的也是帧0。在无差错的情况下，系统的状态仅在4个状态中循环：

（000）→（01A）→（111）→（10A）→（000）→…。

如果信道丢失了帧0，则进行从状态（000）到状态（00-）的转换。最终，发送过程超时（转换7），且系统回到状态（000）。确认帧的丢失更为复杂，需要两种转换：7和5，或8和6，直到修复损坏。从理论上讲，应当共有2×2×4 = 16种不同的状态。去掉没有意义的组合后，还剩下10种状态，而导致状态变迁的输入事件共有9种（标号0~8）。

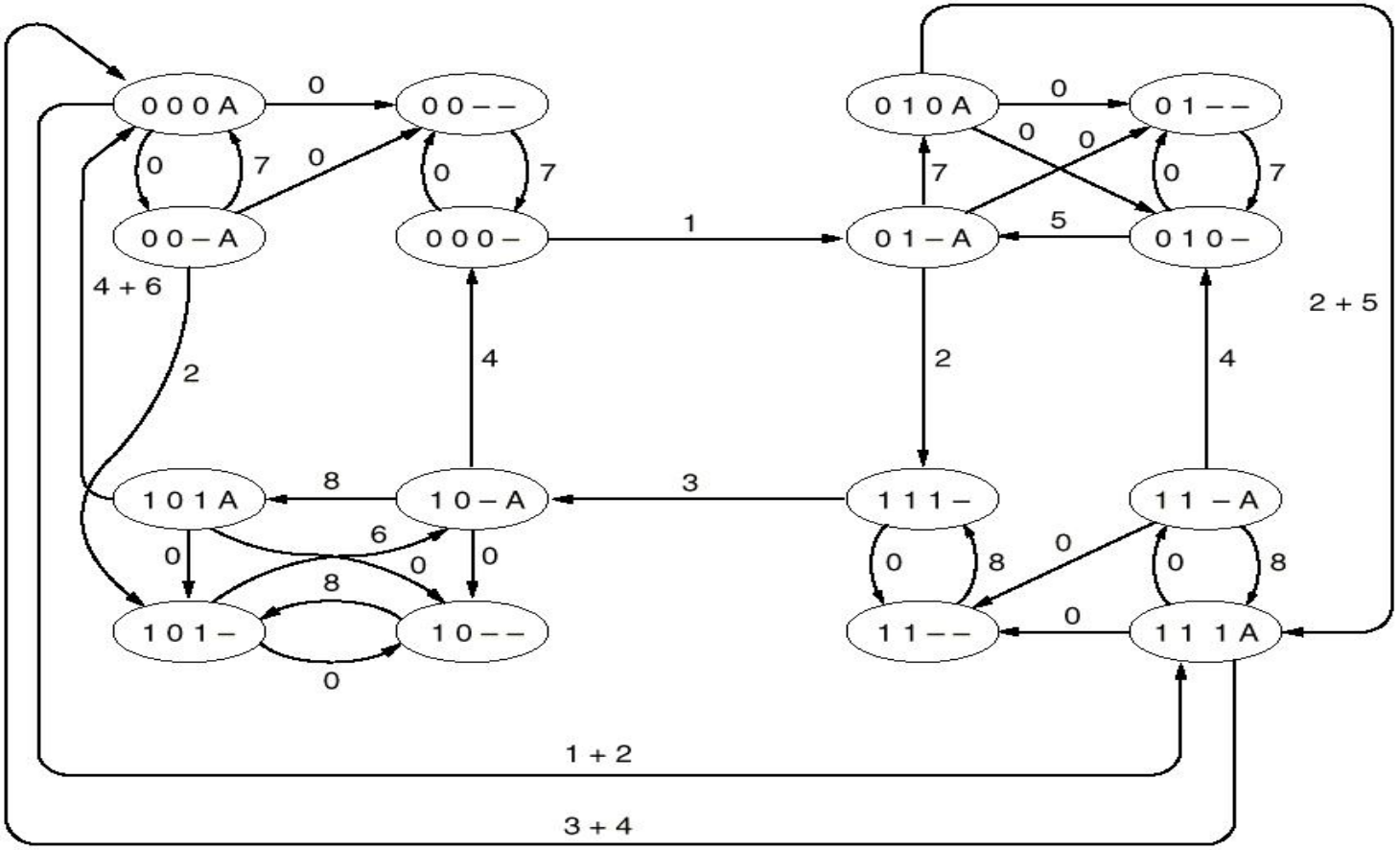
表1　事件及状态变迁表（注:表中X代表为任何状态)

序号	执行方	接收的帧	发送的帧	送网络层	状态变迁	说明
0					XXX→XXS	丢失帧，帧0、帧1或确认帧ACK
1	R	0	A	是	000→01A	收到帧0，期待帧1，向信道发送ACK，帧0送网络层
2	S	A	1	-	01A→111	收到ACK，向信道发送帧1
3	R	1	A	是	111→10A	收到帧1，期待帧0，向信道发送ACK，帧1送网络层
4	S	A	0	-	10A→000	收到ACK，向信道发送帧0
5	R	0	A	否	010→01A	收到帧0并丢弃，向信道发送ACK，不向信道发送帧0
6	R	1	A	否	010→10A	收到帧1并丢弃，向信道发送ACK，不向信道发送帧1
7	S	0	0	-	0XS→0X0	等待超时，重发帧0
8	S	1	1	-	1XS→1X1	等待超时，重发帧1

（2）全双工信道的停等协议，

传送数据帧的正向信道和传送Ack帧的反向信道分开，2个协议机的状态和2条信道的状态的合成状态来描述该协议。转换与半双工一样，只有当数据帧和确认帧同时出现在信道上时，才稍有不同。全双工通信的停止等待协议状态变迁图如下。

- 系统状态：（发送方状态、接收方状态、正向数据信道状态、反向Ack信道状态）
- 则正向信道状态有三种
  - 0——信道上有0号帧
  - 1——信道上有1号帧
  - -/S——信道空
- 而反向信道状态只有两种
  - A——信道上有Ack帧
  - -/S——信道空



(0 0 0 –), (0 1 – A), (0 1 0 A), (1 1 1 A), (1 1 – A), (0 1 0 –), (0 1 – A), (1 1 1 –)

(b)

图a 正反向信道分开的状态变迁图　　图b 错误的状态序列

从图中可以看出，接收过程不能自己单独把数据帧拿走，因为这样做必须在一条信道上同时有两个确认帧，这在我们的模型中是不允许的。同样发送方也不能自己单独拿走确认帧，因为这样做就必须在第1帧接受以前，发送第2个数据帧，如果这两个事件必须一起发生，例如，在状态（000A）和状态（111A）之间的转换，则在图中标记为1+2。至于是先发生1号事件或先发生2号事件则是无关紧要。