网络专业人员书库

网络互连（第2版）

网桥·路由器·交换机和互连协议

(美) Radia Perlman 著

高传善 等译

机械工业出版社
China Machine Press

本书被认为是讲述网络理论和实践的主要书籍之一。除介绍了一般的网络概念外，对路由算法和协议、编址、网桥、路由器、交换机和集线器的功能结构等都提供了权威和全面的信息。包括网络领域的最新发展，如交换和桥接技术、VLAN、快速以太网、DHCP、ATM以及IPv6等。作者以专家的洞察力分析了网络的运作过程和工作机理，并深入到技术背后的概念和原理，帮助读者获得对可用的解决方案的更深理解。

本书适用于作为大专院校计算机专业本科生网络课程的教材，也适用于从事网络研究的技术人员和其他对网络技术有兴趣的人员。

本书版权登记号：图字：01-1999-3473

图书在版编目（CIP）数据

网络互连：网桥·路由器·交换机和互连协议/（美）潘曼（Perlman, R.）著；高传善等译. -北京：机械工业出版社，2000. 12

（网络专业人员书库）

书名原文：Interconnections（Second Edition）:Bridges, Routers, Switches, and Internetworking Protocols

ISBN 7-111-08270-2

Ⅰ.网… Ⅱ.①潘… ②高… Ⅲ.计算机网络-终端互连 Ⅳ.TP393

中国版本图书馆CIP数据核字（2000）第50961号

机械工业出版社（北京市西城区百万庄大街22号 邮政编码 100037）

责任编辑：陈贤舜

北京市密云县印刷厂印刷·新华书店北京发行所发行

2000年12月第1版第1次印刷

787mm × 1092mm 1/16 · 24.25印张

印数：0 001-6 000册

定价：39.00元

第1章　网络基本概念

本章介绍了一些基本概念，这些概念对理解包含网桥和路由器的计算机网络的特定领域是必需的。OSI的网络结构7层参考模型是网络概念的基础，本章介绍的内容覆盖了其中的第二层和第三层。本章还讨论了网络设计中的不同要素，例如网络的覆盖范围、可扩展性、健壮性和自动配置性等。此外，本章还阐述了提供双方可靠通信的典型技术，因为路由器所用的这种技术可以和其他层所用的技术相互影响。

1.1　网络分层模型

不把一个大的任务划分成几个小的子任务而想试图去理解、设计和建立一个网络，将是一件很困难的事情，一般需要把网络分层以便更好地理解它。层的思想是：每一层都将利用它的下一层的服务来向它的上一层提供服务。

每一层都通过协议和其他节点的同一层进行通信。这种通信通过与下一层之间的直接通信来完成。第n层与第$n-1$层之间的通信称为接口。

OSI(开放系统互联)的分层模型定义了7层，如图1-1所示。至于为什么分7层以及各层的功能如何划分其实并不神秘。这个模型在相应协议出现以前就已经设计好了，然后为此成立了一个专门的委员会来设计每一层。实际应用中，很多层经常再被分成若干个子层。至于各层的功能有时并不容易区分，比如网桥和路由器就是这样一个很典型的例子，人们通常搞不清哪一个是哪一层的。但是关于层划分的语义上的争吵并没有什么实际价值，分层的概念应该被视为一种有用的讨论框架而非圣经。

ISO定义的层模型：

1) 物理层：物理层通过链路来传送比特信息。它主要处理以下问题：接插件大小和形状的选择，每一针的作用，数据比特的电信号变换和比特级的同步。通常一个网络内可以有好几种不同的物理层类型，甚至一个节点也可能有多种不同的物理层类型，这是因为不同的技术要求各自的物理层。
2) 数据链路层：数据链路层（有时也称为链路层）通过物理链路来传输成块的信息。它主要负责处理以下任务：数据出错校验、协调共享媒体的使用（如在一个LAN中）以及编址（当多个系统都可以访问时，如在某个LAN中）。另外，不同的链路通常也有不同的数据链路层实现；而且，同一个节点可以支持几种不同的数据链路层协议，节点所连的每一类链路都有自己的协议。
3) 网络层：网络层使得网络中的任何一对系统间都可以相互通信。一个全互连的网络是指其中的每一个节点都和其他节点直接相连，但是这种拓扑结构不可能用于有很多节点的情况。比较典型的情况是，网络层必须找到一条通过一系列相连节点的路径，且路径上的每一个

节点必须向适当的方向转发数据包。网络层处理的主要任务是：路由计算、数据包的分段和重组（当网络中的不同链路有不同的最大包大小限制时）和拥塞控制。

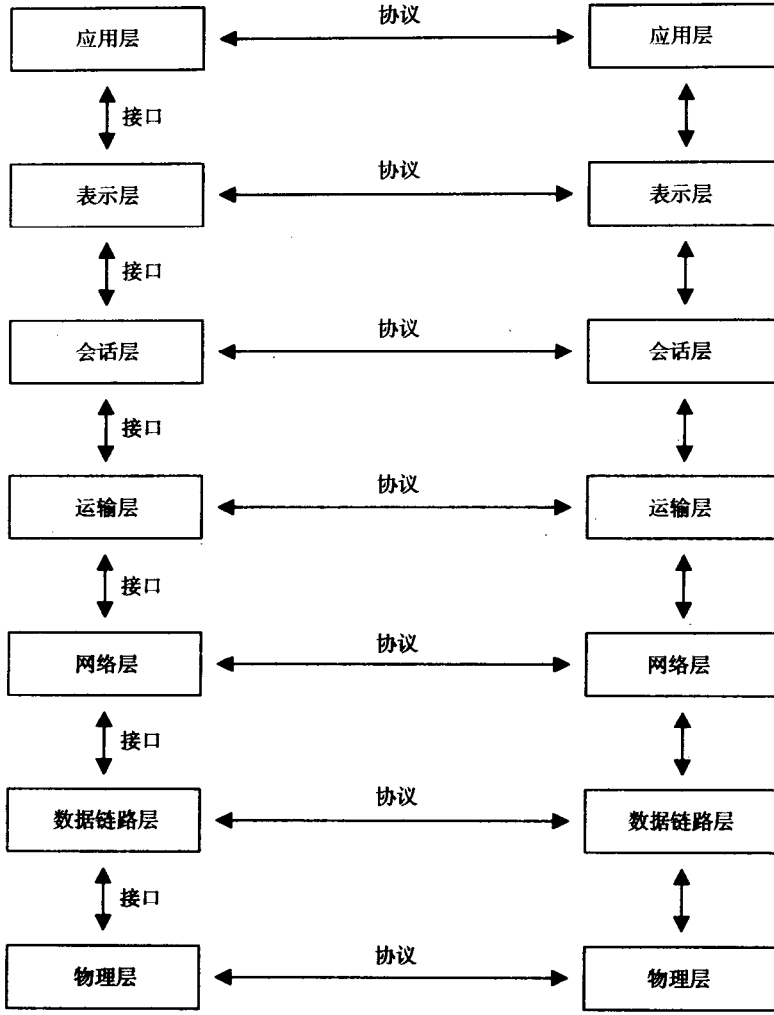


图1-1 OSI参考模型

4) 运输层：运输层在两个系统之间建立一条可靠的通信链路。它主要处理一些由网络层引起的错误，比如包丢失和重复包等错误，以及对包进行重新排序、分段(这样运输层用户就可以处理大的报文)和重装(这样可以避免网络层进行低效的分段和重装)。另外，这也有助于运输层在网络发生拥塞时可以相应降低发送数据的速率。

5) 会话层：ISO认为会话层对于因特网体系来说并没有太大作用。ISO会话层提供的服务超出了运输层提供的简单全双工可靠通信流，比如对话控制（实现系统间的特殊通信模式）和链接（捆绑一组数据包，使得它们要么都发送，要么都不发送)。不管这一层是什么，

它都跟下层的网络设备如网桥以及路由器等无关。

6) 表示层：这一层的设计目的是为了对数据的表示取得一致，这样人们就可以定义自己的数据结构，而不必担心比特/字节顺序或者浮点数该如何表示之类的问题了。ISO在ASN.1(Abstract Syntax Notation 1)中制定了标准。尽管我不是很喜欢ASN.1，因为它太复杂而且效率太低（在空间和处理方面），但是很多IETE(Internet Engineering Task Force)的标准都使用了它。

7) 应用层：桥接和路由之所以吸引人，实际是因为人们需要利用这些功能的相应应用。应用包括文件传输、虚拟终端及Web浏览等。在一个节点上通常有多个应用程序同时运行。

本书是和数据链路层有关的，因为网桥在该层操作，且该层提供的服务与路由器相关。路由器在网络层操作，因而本书和网络层也是相关的。另外，本书与运输层也有一定的关系，一是因为它使用了网络层提供的服务，二是因为网络层做的一些决定（比如说是否允许数据在对等的几条链路上同时传输等）会影响到运输层。在运输层上面的那些层就跟网桥以及路由器基本无关了。

通常，第n层从第n+1层得到一大块包含有一些必要的附加信息（如目的地址等）的数据后，第n层把这些数据传输给目的节点的第n层处理，目的节点的第n层再把这些数据送给该节点的第n+1层。第n层经常需要在数据包上附加一些信息，比如说，目的节点地址（它将被其他的第n层实体解释）。为了得到目的节点的信息，第n层将传递一个数据块给第n−1层，里面包括从第n+1层上来的数据以及第n层上附加的一些控制信息。另外，第n层还可能在这个数据块中加入一些其他的信息，比如说第n层与第n−1层之间的接口信息。

让我们来看看分层工作的例子。假设物理层允许比特流从一台机器传送到另一台机器。数据链路层在该串比特流上做上标记，用以表明数据包的开始和结束位置。此外，它还附加上校验和信息，使得接收的机器可以检测线路上是否有噪声而导致数据错误。

目前人们已经使用了好几种有意义的技术来确保那些表明数据包开始和结束的标志的比特串不会出现在数据包中。一种是比特填充技术，标志为六个连续的1。为了确保6个连续的1不会出现在数据包中，传输方在每5个连续的1后面插入一个附加的0。接收方在接收比特流时，如果发现连续的5个1后面是0，则这个0必须被删除或忽略；如果在5个连续的1后面还是1，这就标志着数据包的开始或者结束。

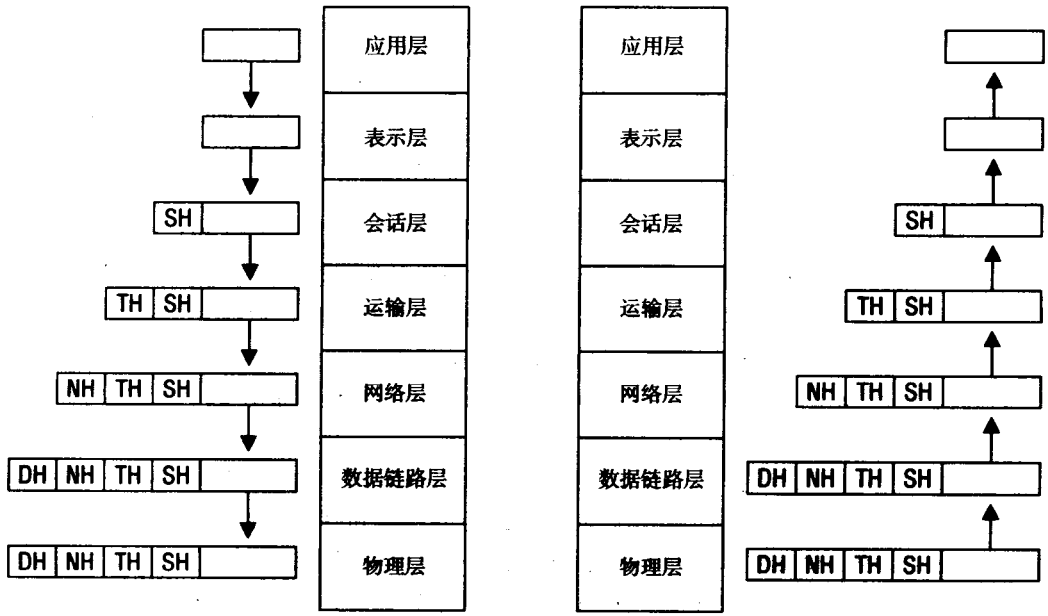
网络层通过与其他相连机器的网络层之间的合作，可以计算出一条路由，从而使得信息能经过多个跳到达目的节点。

当网络层从运输层接到一个待传输的数据包时，它给这个数据包加上一个“信封”，一般是在原数据包的前面（称为报头）（有可能同时在后面（称为报尾））附加上信息。这个“信封”包括一些诸如源地址、目的地址等的信息。网络层将选定一条比较好的路径来发送数据包，然后把连同网络层信封在内的数据包传递给数据链路层，后者将负责把数据发送出去。

当数据包到达中间节点时，由数据链路层进行处理。首先把数据链路层的附加信息除去（把“信封”去掉），然后把它传递给网络层，这时的数据包看起来就跟前一个网络层传递给数据链路层时的一样，即包括运输层下传的所有信息加上网络层信封。接收端的网络层查看这个数据包的网络层附加信息（“网络层信封”），确定这个数据包的去向，必要的时候，修改这些附

加信息（比如增加跳数，以表明数据包已经过的节点数），然后再把修改过的数据包交给负责发送链路的数据链路层处理（见图1-2）。

在前面的叙述中，像数据包这样的词比较容易使人混淆。ISO发明的术语让一切都变得简单明确。每一层通过协议数据单元（PDU）和它的对等层进行通信。为了更准确地表示出当前讨论的是哪一层，通常在该层的PDU前面增加一 个单字母的前缀，如数据链路层通过传送LPDU和对等的数据链路层进行通信。网络层则通过NPDU和其他网络层进行通信。类似地，运输层通过TPDU和其他运输层进行通信。



SH = 会话层报头
TH = 运输层报头
NH = 网络层报头
DH = 数据链路层报头

图1-2　低层封装

当第n+1层发送消息给第n层时，这个消息称为SDU（服务数据单元）。和PDU一样，在SDU前面，我们也用一个字母来表示它属于哪一层。当一个运输层想传输一个TPDU给另外一个运输层时，它需要先给网络层一个NSDU。网络层接收了这个NSDU后，给它加上一个“信封”，然后再通过数据链路层把这个NPDU发送出去(参见图1-3)。

作为一条规则，本书不采用ISO的术语，因为它过于冗长且难以迅速理解（除非你已经参加过三次以上的标准会议）。但是，在需要的时候，我们还是偶尔会采用它的术语来进行说明。

1.2 服务模型

一般说来，第$n-1$层提供给第n层的服务是通过数据传送来实现的。第n层提供数据（SDU）以及一些附加信息（如目的地址）给第$n-1$层。第n层也能通过第$n-1$层给它的一个通告信息，从对等的第n层接收数据。

第$n-1$层既可提供无连接服务，也可以提供面向连接服务。在无连接服务的情况下，它可以接收第n层发给它的数据包，也可以发送数据包给第n层。

在面向连接的情况下，数据传输之前首先要建立连接。通信包括以下三个步骤：

1) 连接建立。

2) 数据传输（发送或接收数据）。

3) 连接释放。

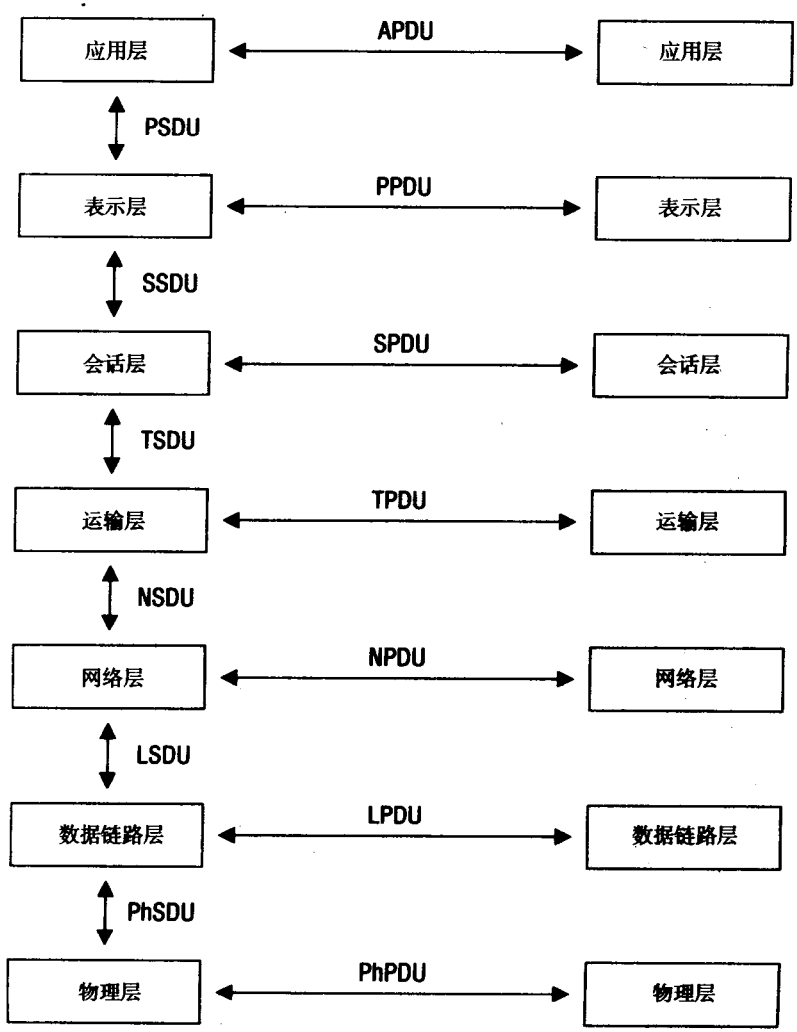


图1-3 PDU和SDU

相应地，对于上面的每一步，都有两个活动：一是第n层初始化该活动；二是第$n-1$层通知第n层它的对等层已经初始化好该活动了。

对于连接建立，可以是第n层请求与某个目的地址建立连接，或者是第$n-1$层通知第n层有某个节点希望与它建立连接。

对于连接释放，第n层请求释放某个连接，或者第n-1层通知第n层有某个节点希望释放它们之间的连接。

不同的接口设计可能还会提供其他功能，但前面所述的是最基本的。

不同服务的可靠程度也不同。纯数据报服务（也称为best-effort）接收数据，但不对发送做任何保证。数据可能丢失，也可能重复，乱序，还可能被损坏。而可靠服务将保证（或宣称它将保证）数据按正确顺序发送，不会被破坏（比特流不被破坏），也不会出现重复数据包或者数据包被丢失的情况。建立可靠的面向连接的网络(比如X.25)和数据报(比如ATM）是可能的。无连接的网络层只提供数据报服务，因为根据定义它无法知道发送的是什么。IP、IPX、DECnet、CLNP和Apple talk都是无连接网络层的例子。图1-4列举了几种不同类型的网络层。

	面向连接	无连接
数据报	ATM	IP, IPX, DECnet
可靠的	X.25	

图1-4 网络层类型举例

直观地讲，提供可靠服务应是需要的，但这种可靠服务是要付出代价的。它通常使得第$n-1$层花费增多、效率降低。虽然有的时候，在多层上都提供可靠服务是有理由的，但绝对不需要在所有层上都提供可靠服务。

我们将在第6章讨论面向连接与无连接服务之间，以及数据报与可靠服务之间的折衷。在ISO和IEEE制订的标准里，提倡面向连接可靠服务的人和提倡无连接数据报服务的人，一直都无法说服对方，所以在ISO和IEEE的协议中，这两种服务类型都可能提供。这就是为什么在第2章将讨论的LAN中，要提供两种风格的、运行在LAN上的数据链路层，这种服务称为LLC(逻辑链路控制)。LLC类型1是无连接的数据报服务，LLC类型2提供的是可靠的面向连接的服务。ISO在这之上定义了两种网络层服务，其一是CONS（面向连接的网络服务），其二是CLNS（无连接的网络服务）。

正因为人们没有就选择哪种服务类型达成一致，ISO定义了5类运输层，从TP0到TP4。TP0假定网络层负责完成几乎所有的事情，而TP4认为网络层只提供简单的数据报服务。

在TCP/IP协议中，网络层(IP)是无连接的。有两种运输层，一种是TCP（传输控制协议）提供可靠的面向连接的服务；另一种是UDP（用户数据报协议）提供数据报服务。

ATM提供面向连接的不可靠服务，这时可以认为它提供的是网络层的服务。但是如果有其他的网络层服务，比如基于ATM的IP，则ATM被当作是IP的数据链路层。正如前面提到过的，不必太严格地区分层的界限。

第6章中有更详细的有关服务模型的讨论，包括性能保证的问题。

1.3 网络的重要特性

不同的网络设计有可能看起来提供了相似的功能，但是，它们在细节上却可能不同。在评估一个网络体系的时候，必须考虑下面的特性：

1) 范围：一个网络体系要尽可能地解决一切常见的问题。它要设计成可以全面支持应用程序和全面支持下层的各项技术。如果网络是为特定的一个应用程序而设计或是建立在一种特殊的技术之上，它也许会在那种条件下表现得更加出色。但是，想这样针对每一个特定的要求设计一种单一的网络是不太可能的。除非一个全面解决方案并不能满足你的需要，否则最好还是设计一个可以对付大范围的应用程序和下层技术的网络。

2) 可扩展性：一个理想的网络设计不仅要能够在非常大的网络上运作得好，而且要在小网络上也很高效。在过去，一个拥有上千个节点的网络就可以被认为是很“大”了。而在现在的标准化设计中，我们应考虑体系结构是否适合上百万个乃至上亿个节点。当然在理想的情况下，这个设计运用在一个非常小的网络里（譬如只有20个节点），效率不会降低，但不可能只要求实现这样的目标。在这种情况下，一个针对少量节点设计的网络可能会非常高效，譬如它可以使用更少的地址空间。但是，我们更希望一个折衷的完整解决方案，只要它能够在专门的网络上取得足够的效率。

3) 健壮性：健壮性的某些方面是显而易见的，绝大多数的网络设计也都考虑到这一点。例如，即使节点或链路失效，网络也应该可以继续运行；大多数网络采用的路由算法都能适应改变了的拓扑结构。但是，健壮性还有其他更多的、比较微妙的方面。

大多数的网络在一个理想的理论环境下都能正常工作，前提是：不出现未检测到的数据错误；所有节点都正常运行所有算法；参数设置对所有节点都是兼容的；以及所有节点都有足够的处理器能力及时完成必需的算法运算等。

但是，我们面对的实际环境不是乌托邦，难以察觉的数据错误伴随着未被检测到的传输错误、节点内存中的错误及内部数据总线传送中的数据错误而时时发生。有缺陷的实现会连到网络上。硬件故障有可能导致不可预料的后果。实现耗尽了内存或CPU，并引起不可预料的行为，而不是立即停止操作或是执行与网络的持续有效的功能相兼容的操作。人为因素（一个系统中最不可靠的部分）导致的错误配置等。

因此，仅仅有计算可选路由的健壮性是不够的。一个网络还必须有下面几种类型的健壮性：

a. 安全壁垒：对于绝大多数网络来说，恶劣的性能可能导致大范围的瘫痪。但是有一些经过精心设计的网络，能够保证错误不会扩散到一个安全的范围（安全壁垒）之外，因而瘫痪只会限制在网络的一部分，而不会影响整个网络。

举例来说，LAN上的“广播风暴”对于TCP/IP网络层协议来说，始终是让人头痛的事。“广播风暴”是指触发严重网络拥塞的事件，它通常是由于应用中的小错误（bug）、有歧义的协议规范以及不正确的配置而引起的，严重的“广播风暴”可以使LAN失效。当两个LAN通过桥连接到一起时，桥就合并了两个LAN，任意一个LAN中的“广播风暴”会使两个LAN一起失效。如果LAN间是用路由器连接的，那么“广播风暴”只限

制在它发生的那个LAN中。因此，路由器就起到了使“广播风暴”不能扩散的安全壁垒的作用。

另一种安全壁垒可以通过设计一种层次路由算法，将网络划分成区域（area）或域（domain）。这种路由算法可以被设计成即使一个分区中发生瘫痪，也不会扩散到其他部分。

b. 自稳定性：这个概念是指，即使发生了因为硬件故障或者未察觉的数据错误而引起的任何数据库崩溃，经过一定的时间后，只要发生故障的硬件被从网络上断开或被修复，或一段时间内没有更多的数据错误出现，网络就可以在无人干预的情况下恢复正常运作。如果缺少这种健壮性，一个错误就可能使整个网络一直不能运作，直至所有的节点都被同时关掉然后重新启动。在第12章讨论路由算法时，我们将可以发现ARPANET中实现的路由算法不具有自稳定性。

这种健壮性不能保证网络在连接有故障设备的情况下能够正常运作，但是它使得在诊断出问题后，能相对容易地修复网络，只需要拆除出错的设备。许多设备陷入了“粘着状态”，通常重新上电自检就可以立即修复。但是网络并没有开/关按钮，也就不容易重新上电自检了。网络的健壮性（从它是分布式的，即使其中一部分瘫痪了仍可以保持可运行的意义上说）就是指如果系统不是自稳定的，则它必须被关闭以消除错误的任何残余。

如果一个网络不是自稳定的，那么蓄意破坏的人可以发一些错误的包，网络就会永远瘫痪或者直至由复杂而昂贵的人工干预来修复它。而如果一个网络是自稳定的，那么蓄意破坏的人，只能一直不断地发送错误的包，以使网络一直瘫痪。这比前面的趁无人监管时偷偷连上网，发一些坏的包，然后悄悄溜走要冒更多的风险。

此外，如果网络是自稳定的，修复是相当容易的。当找到罪魁祸首后，只要将这个设备从网络上断开，就可以使网络重新回复到可运行状态了。

c. 错误自检：虽然现今的任何一个网络都不能在有活跃的错误节点（Byzantine错误，下面会讨论到）的情况下正常运行，但如果网络能够自我诊断并标志出出错的设备也是很有帮助的。所有的网络都有一些发现错误的能力，但都不能完美地做到这一点，而且不同网络能发现的错误的层次也不尽相同。

d. Byzantine健壮性：术语“Byzantine错误”是从计算机科学中的一个著名问题“Byzantine一般问题”中得来的。Byzantine错误是指一个出错节点不是不能运行，而是错误运行。这样的错误通常是因为有缺陷的实现、硬件错误或者主动破坏而产生的。具有Byzantine健壮性的网络能够在即使部分节点存在Byzantine错误的情况下仍然正确运行。虽然现今的网络没有这个特性，但是这样的网络是可能的（详见第16章）。

4) 自动配置：有些网络设计方案需要一些非常聪明的管理员进行很多复杂的管理工作和经常性地修改参数才能正常工作。这样的网络方案使那些懂得管理它们的人勿需担心他们手中的饭碗，但是这种网络远不能满足未来的需要。网络将会变得非常大，它们被分成由不同组织管理的各个部分，人们将更加依赖那些要靠少数专家来保证运行的网络。

未来的网络必须能够最大可能地实现自我运作。理想情况下，普通用户可以从当地的零

售商店购买相应设备，把它们插入网络，就得到想要的运行网络。他们不需理会复杂的参数配置，不需要向负责地址分配的“头头”申请得到一个地址（否则，如果“头头”渡假去了，或者压根就辞职不干了，或者写有分配到的地址的信封丢了，可就麻烦了）。用户们不用为了把他们的节点信息加入到数据库中而去找另外一些节点的管理人员。

5) 绞合性：网络应该有自己合理的默认值，在理想情况下，应该能够自动配置。但同时它们应该有一些定时器和其他的参数，使得那些喜欢研究的网络管理员能有机会为某些特定情况优化网络性能（理想情况下，参数的任何设定都可以产生合理的（如果不是最优的）性能，这样即使十分喜欢冒险的网络管理员也不会引起太大的损害）。

6) 决定性：根据决定性的特点，相同的条件将产生相同的结果。例如，在一个决定性网络的设计中，相同的物理拓扑结构就会有相同的路由。相反，在一个非决定性网络的设计中，网络中节点的启用顺序不同，路由就可能会有差别。并不是所有的人都认为决定性是有价值的，因为在某些情形下如果最高优先权的元素一直在失败和重新启动，就相当于崩溃。但决定性的支持者们认为，通过保证可再现的条件，决定性使网络的分析更加容易。

7) 迁移：一个网络的设计不会永远地保持下去。设计网络协议，使新的特性能逐个加入到节点中而不干扰当前的操作，就显得很重要。能够修改的设计也同样重要，例如地址的改变，可以按逐个节点修改的方式进行而不会破坏网络操作。

1.4 可靠的数据传输协议

所有提供可靠服务的数据链路和运输协议几乎有着相同的结构。本节将介绍基本的思想并引出相关的更深层的问题——这些问题会在本书的后面部分详细展开。

协议必须按与源相同的顺序发送一系列的包，如果任一包丢失、损坏、错序或重复，协议便失败了。基本的思想是一个包被传送，接收方对它的到达发出确认。包有一个校验和，使接收方能（以很大的可能性）检测出包是否在传送过程中受到了破坏。

在十分简单的方案中，发送方发出一个包，等待确认（也可称为ack），然后发送下一个包（见图1-5）。假设正在传送的数据是“1000 from acct A to acct B”这条消息，且每个包中只能有三个字符。如果一个确认没有到达，发送方必须对数据进行重发。因为发送方不能肯定确认是否会到达，它只能设置一个计时器，并假设如果确认在那段时间内不能到达，包（或者确认）可能丢失了。

如何设置计时器的值是十分棘手的一件事情。如果接收方是一台很忙的服务器，那么它的响应时间可能会变化很大。当系统满载的时候，产生确认的时间也许会很长以至于发送方已经放弃并重传。如果传输数据的链路是一个计算机网络，一些包可能会采用不同的路径，而各条路径有着不同的时延；或者网络有时会变得拥挤（在美国的高速公路网络上通过相同的路径时，在高峰期间要比在凌晨3点花费更多的时间）。

如果计时器的值太小，包就会被不必要地重传，增加了网络的拥挤程度或接收方的处理负担（见图1-6）。如果计时器的值太大，包丢失后吞吐量会被延迟，因为必须等到计时器超时才能进行数据传输的下一步过程。

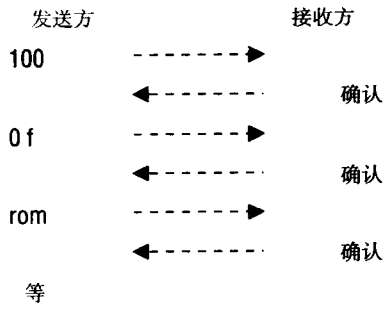


图1-5 简单协议，无差错

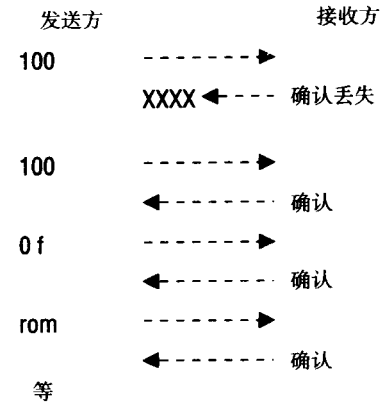


图1-6 简单协议，丢失确认

在这一点上，也许比优化计时器更严重的问题就是协议出错了。如果一个确认被丢失或延迟了，接收方会收到同一个包的两个拷贝并且不知道它们是重复的。结果就是接收方会认为信息是“1001000 from acct A to acct B”。账户B的所有者会因为多收了一百万美元而感到高兴；但账户A的所有者会提出反对。因此，协议必须作些修改。

解决的方法是加上包的编号，并在确认上加上相应的编号，这样一个确认就可以与正在被确认的包对应起来了（见图1-7）。

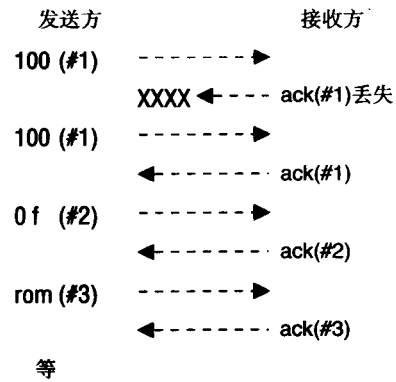


图1-7 增加包编号

接收方接收了100（#1）两次，但因为两个包都被标记为（#1），因而得知它们是重复的并且只保留一个。

通常数据在两个方向上同时传输。在这种情况下，从右到左的包的编号完全和从左到右的包的编号无关。这不会产生不确定性：从右到左传输的确认中的编号与从左到右传输的带编号的数据包流有关，而从左到右传输的确认中的编号与从右到左传输的带编号的数据包流有关。

如果发送方在发出每一个包后必须等待确认，吞吐量就会不必要地变低。在数据发送方完成发送一个包以后至得到确认以前，有三件事情是一定会发生的：包必须经过发送方和接收方之间

的路由；接收方必须对包进行处理并产生确认；确认也必须经过接收方和发送方之间的路由。因为发送方在完成发送以后和收到确认之前有空闲的时间，应利用这段时间来传输更多的数据。

在接收到前面的数据的确认之前发送额外的数据，称为“流水线操作(pipelining)”。允许发送方拥有的“多出的”（即发送后还没有收到确认）包的数量称为“窗口”。

另一个问题是包的数量的大小。理想的情况是这个数量可以无限大，通话的第一个包将是1号，第120亿个包将是12 000 000 000号。这样会使协议十分简单，发送方可以以最快的速度发送所有的包。接收方将对它所收到的所有包做出确认，并且发送方可以把洞填补好（注意到某些包的确认没有接收到，比方说17号、112号和3178号）。

然而，协议通常被设计成使用尽可能少的位来表示数据以外的信息（如果不是这样，负责商标真实性的政府机构会要求将服务称为“报头报文”而非“数据报文”，因为报头占用的位数太多）。通常，会为包的编号保留有限的位数，在这个例子中（见图1-8）我们使用3位，在真正的协议中可能希望得到更大的位数。

使用3位的包号码，7号包被传送以后会发生什么事情？答案是包编号循环了。包是以0，1，2，3，4，5，6，7，0，1，2，3，4，...来编号的。

我们通常会设计这种协议：不必为每一个包都传送确认。而是将每个确认积累起来，这样一个4号包的确认就确认了所有4号前的包都被正确接收了。

有限的包编号、流水线和累积的确认，如果不谨慎地实现，会产生问题。下面是一些例子。

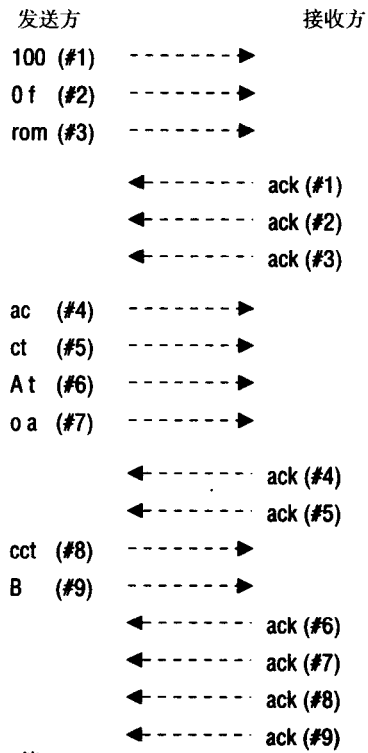


图1-8 增加流水线机制

1) 发送方发送的包的编号为0，1，2，3，4，5，6，7，0，1，2，3。假设头4个以后的包丢失了，接收方就会收到0，1，2，3号包并返回3号确认。发送方和接收方都会认为所有包都已正确到达。
2) 发送方发送的包的编号为0，1，2，3，4，5，6，7，0，1，2，3，而提供数据报服务的网络改变了次序。接收方将无法知道哪个2号包先到达，哪些包重复了。
3) 发送方发送编号为0，1，2，3，4，5，6，7的包。接收方全部接收并返回(#7)确认，而这个确认在传输中丢失了。发送方会重传全部的8个包，接收方则会将它们作为新的数据接收下来并返回另一个(#7)确认。

解决这些问题的方法是谨慎地设计计时器和窗口。包的编码空间足够大是很重要的，从而使得包的编号在出现最长时延时也不会循环。因此，如果可以想像到网络对一个包的时延可达15秒，那么，发送方以最快的速度传送足够多的包时，包的编号一定不能在15秒内出现循环。如果接收方保留错序的包（例如，如果接收方接收到1号包然后是3号包，它会保留3号包并期待2号包能最终到达），则窗口不能大于包的编号的大小的一半也很重要。在严格的顺序信道中，包可能丢失，但次序绝不会发生错误。如果能保证接收方放弃除了下一个连续编号的包之外的所有包，窗口可以和包的编号的大小减1一样大。

习题

1. 假设会话层把信息传递给运输层，运输层用一个包来传输这个信息（加上一些可能的控制信息）。在ISO模式中，这称为“单个SPDU产生单个TSDU，单个TSDU产生单个TPDU”。假设会话层把一个太大的信息传递给运输层，运输层会用一组包来传送它，并有足够的控制信息使目标运输层能重新组装它们得到会话层的信息，这在ISO模式中称为什么？
2. 现在假设由于效率的原因，运输层能够接收从会话层传下来的许多小块信息，并把它们放到一个大“盒子”中以便能用一个包来传送。在这种方式下，目标运输层可以把这些小块的独立信息重新挑选出来。在ISO模式下如何表达这种情形？
3. 现在假设网络层有一个包要发送，但一些中继的路由器注意到这个包太大而不能在链路上传输。中继节点的网络层把包分成多个小块，使得它们能在目标网络层被重组。这在ISO模式下如何表达？
4. 现在假设包编号的大小为n，信道是顺序的。给出一个协议失败的例子，在这个协议中发送方有一个大小为n的窗，接收方丢弃次序出错的包。证明如果窗的大小为$n-1$，则不会出错（假设开始时所有旧的包都从信道上删除了）。
5. 现在假设接收方不丢弃次序出错的包。也就是说，接收方保留n号包，期待$n-1$号包最终能到达，并且接收到后对两个包都作出确认。给出一个协议失败的例子，在这个协议中发送方有一个大小为$n/2+1$的窗口。证明如果窗口的大小为$n/2$，则不会出错。
6. 讨论在数据链路层上可靠服务和数据报服务之间的折衷，假设在两种情况下运输层都能

提供可靠的面向连接的服务。

考虑以下几点：

a. 包能够通过一系列链路的概率（这取决于这些链路的出错率）；

b. 包能够成功传送到目的节点且确认能传送回源所需要经过的全部跳数；

c. 希望运输层的吞吐量最大化；

d. 为了决定什么时候重传一个没有确认的包，运输层需要估计往返的时延时间。

第2章 数据链路层

本章讨论影响网桥和路由器的数据链路层的问题。问题之一是数据链路层提供的服务应该是可靠的还是面向数据报的。另一个问题是如何使多种网络层协议并存在同一链路上。当节点收到一个数据包时，它怎么知道这个数据包是由什么协议产生的？尽管本章也讨论了很多LAN技术这个令人入迷的话题，但并不意味着本章是对LAN技术的详尽介绍。更准确地说，这里解释了一般LAN的一些技术，和那些影响桥接和网络层协议的特定LAN技术。

2.1 一般的LAN

2.1.1 什么是局域网

当人们使用“LAN”一词的时候，他们可能指的是某些拥有和LAN联系在一起的特性的技术。下面就是其中的一些特性：

- 多个系统连接到同一个共享媒体上。
- 高带宽(由所有站点共享的全部带宽)。
- 低时延。
- 低差错率。
- 广播能力，也被称为组播（multicast）能力（传送单个消息给多个接收点的能力）。
- 有限的地理区域（几公里内）。
- 有限数量的站点（数百个)；
- 各站点之间的对等性（与多个终端和一台主机的情况相对）。在对等情况下，所有连接的站点都是等同的。在主从情况中，一个特殊的节点叫做master，其余的则是slave。
- 被限制成私用的，不使用公共电话电报网络。

注意“低”、“高”、“有限”等词都是相对的，随时间而变化。根据本书的目的，我们并没有必要去严格区分局域网（LAN）、城域网（MAN）和广域网（WAN）（幸运的是,现在LAN技术的提高扩展了地域，而广域网技术的发展增加了带宽，使LAN和广域网之间的区别越来越不明显，而且要指出哪些是城域网是徒劳的）。

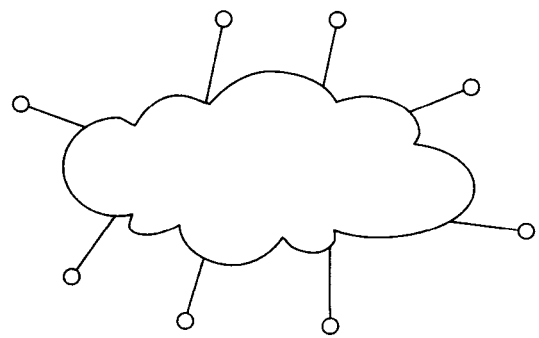


图2-1 网络云图

基本上，一个LAN（或者广域网）可被看做是各个站点能连接到的“云”（图2-1）。如果

一个站点连接到这个“云”，它就可以向其他所有连接的站点发送和接收数据包。

2.1.2 轮转

在共享的媒体上，一次只有一个站点能成功地进行传输。必须存在某种在站点间分配带宽的机制，使得

- 每个站点都能公平地共享带宽。
- 每个站点在合理的时间内都能获得访问媒体的权利。
- 由于仲裁机制而浪费的带宽必须最小化。

LAN上最流行的两种仲裁机制是令牌方案和竞争方式。

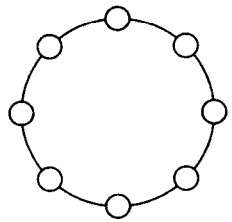


图2-2 令牌环

在令牌方案中，每个站点按某种循环的方式被准许传输。在令牌环（token ring）中，一个特殊的位序列——令牌（token）在环上循环（图2-2），当一个站点得到令牌的时候才被允许进行传输。

在令牌总线（token bus）中，令牌是从站点发送到站点的一个特殊的包（图2-3)。每个站点都需要知道它将把令牌传送给的那个站点。

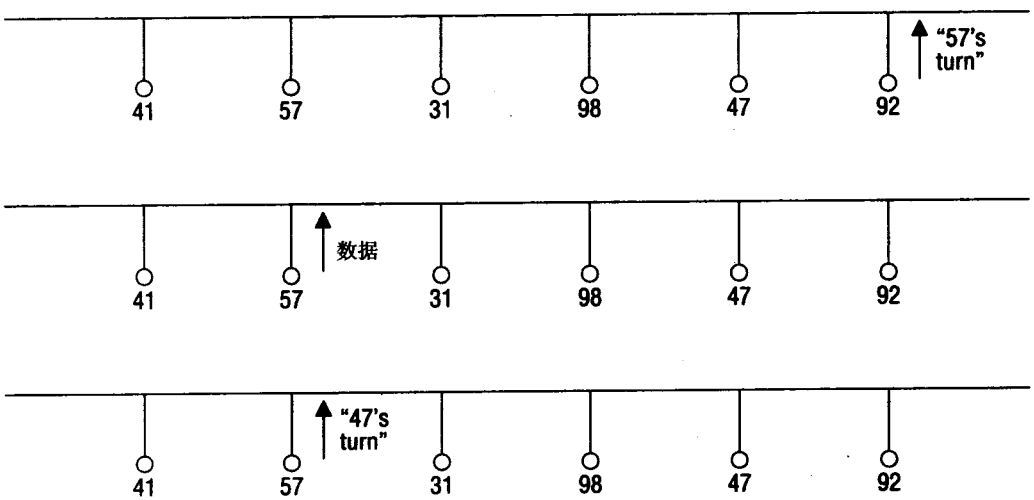


图2-3 令牌总线

在竞争方式中，站点按照自己的意愿进行发送，如果两个站点同时进行传输，冲突（collision）就发生了，而且它们的传输都不会成功。竞争机制是为了最大可能地减少冲突而建立的。竞争方式是“概率上公平的”，这就是说理论上有些站点将永远也不能进行传输，因为每次它进行传输时都有其他节点同时传输。但竞争方式的LAN是经过小心设计的，使得这种情形不太可能出现。

依赖于可能性而非一个能保证所期望结果的方案，看起来很令人不安。但是你必须认识到我们每天都是依赖于可能性的。房间里有足够你呼吸的氧气存在其实也是一种可能性，理论上