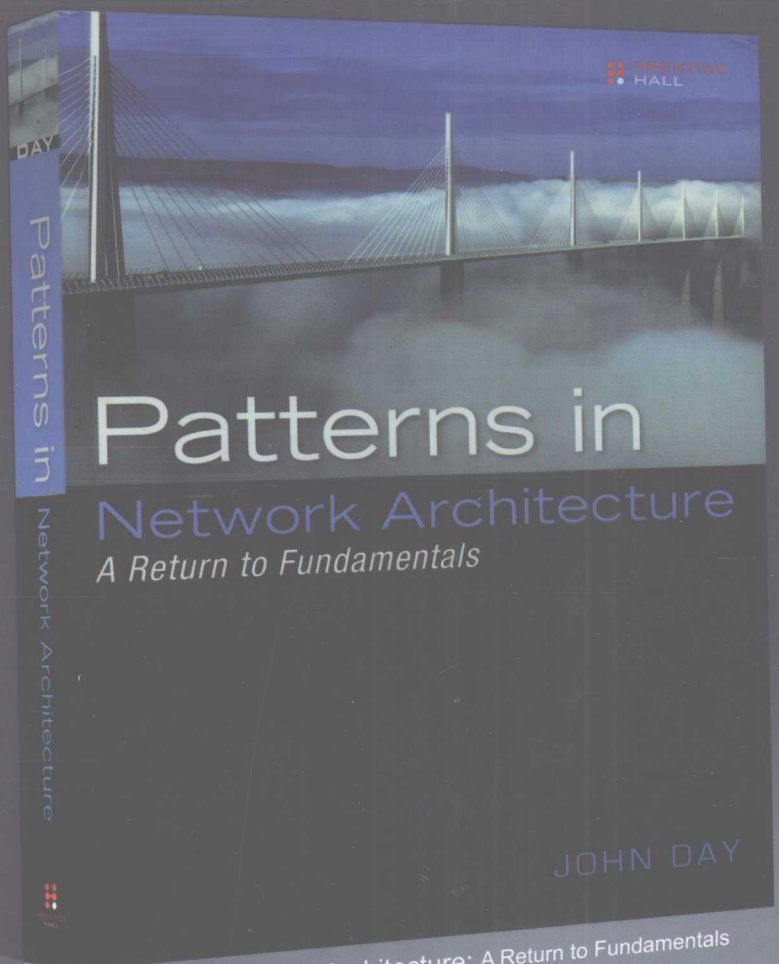
国外计算机科学经典教材

# 网络体系结构模式

(美) John Day 著　　付勇 译




清华大学出版社

国外计算机科学经典教材

# 网络体系结构模式

(美) John Day 著

付勇 译

清华大学出版社

北 京

图书在版编目(CIP)数据

网络体系结构模式/(美)戴(Day, J.)著；付勇 译. —北京：清华大学出版社，2009.6

(国外计算机科学经典教材)

书名原文：Patterns in Network Architecture: A Return to Fundamentals

ISBN 978-7-302-20180-9

I. 网… II.①戴… ②付… III.计算机网络—网络结构 IV.TP393.02

中国版本图书馆 CIP 数据核字(2009)第 071381 号

**责任编辑**：王 军 于 平
**装帧设计**：孔祥丰
**责任校对**：成凤进
**责任印制**：杨 艳

**出版发行**：清华大学出版社 **地 址**：北京清华大学学研大厦 A 座
http://www.tup.com.cn **邮 编**：100084
**社 总 机**：010-62770175 **邮 购**：010-62786544
**投稿与读者服务**：010-62776969，c-service@tup.tsinghua.edu.cn
**质 量 反 馈**：010-62772015，zhiliang@tup.tsinghua.edu.cn

**印 刷 者**：清华大学印刷厂
**装 订 者**：三河市新茂装订有限公司
**经 销**：全国新华书店
**开 本**：185×230 **印 张**：19.25 **字 数**：396 千字
**版 次**：2009 年 6 月第 1 版 **印 次**：2009 年 6 月第1 次印刷
**印 数**：1～4000
**定 价**：39.00 元

本书如存在文字不清、漏印、缺页、倒页、脱页等印装质量问题，请与清华大学出版社出版部联系调换。联系电话：(010)62770177 转 3103 产品编号：028732-01

# 出 版 说 明

近年来，我国的高等教育特别是计算机学科教育，进行了一系列大的调整和改革，亟需一批门类齐全、具有国际先进水平的计算机经典教材，以适应我国当前计算机科学的教学需要。通过使用国外优秀的计算机科学经典教材，可以了解并吸收国际先进的教学思想和教学方法，使我国的计算机科学教育能够跟上国际计算机教育发展的步伐，从而培养出更多具有国际水准的计算机专业人才，增强我国计算机产业的核心竞争力。为此，我们从国外多家知名的出版机构 Pearson、McGraw-Hill、John Wiley & Sons、Springer、Cengage Learning 等精选、引进了这套“国外计算机科学经典教材”。

作为世界级的图书出版机构，Pearson、McGraw-Hill、John Wiley & Sons、Springer、Cengage Learning 通过与世界级的计算机教育大师携手，每年都为全球的计算机高等教育奉献大量的优秀教材。清华大学出版社和这些世界知名的出版机构长期保持着紧密友好的合作关系，这次引进的“国外计算机科学经典教材”便全是出自上述这些出版机构。同时，为了组织该套教材的出版，我们在国内聘请了一批知名的专家和教授，成立了专门的教材编审委员会。

教材编审委员会的运作从教材的选题阶段即开始启动，各位委员根据国内外高等院校计算机科学及相关专业的现有课程体系，并结合各个专业的培养方向，从上述这些出版机构出版的计算机系列教材中精心挑选针对性强的题材，以保证该套教材的优秀性和领先性，避免出现“低质重复引进”或“高质消化不良”的现象。

为了保证出版质量，我们为该套教材配备了一批经验丰富的编辑、排版、校对人员，制定了更加严格的出版流程。本套教材的译者，全部由对应专业的高校教师或拥有相关经验的 IT 专家担任。每本教材的责编在翻译伊始，就定期不间断地与该书的译者进行交流与反馈。为了尽可能地保留与发扬教材原著的精华，在经过翻译、排版和传统的三审三校之后，我们还请编审委员或相关的专家教授对文稿进行审读，以最大程度地弥补和修正在前面一系列加工过程中对教材造成的误差和瑕疵。

由于时间紧迫和受全体制作人员自身能力所限，该套教材在出版过程中很可能还存在一些遗憾，欢迎广大师生来电来信批评指正。同时，也欢迎读者朋友积极向我们推荐各类优秀的国外计算机教材，共同为我国高等院校计算机教育事业贡献力量。

清华大学出版社

# 国外计算机科学经典教材

## 编审委员会

# 前　言

科学使人着迷。在其中我们可以通过对少量事实进行研究而获得大量的猜想。

—Mark Twain, *Life on the Mississippi*

## 0.1　七个未解的问题

我们并不打算用一本书的篇幅来进行阐述。这里只是要提炼我们对网络的认识。我们能够提炼出哪些原理、经验法则和指导原则呢？如何用尽可能少的限制描述它们？可以引入一些更宽松的约束吗？什么实际上没有变？

这些年来，我看到有些思想没有得到发展，而且发展的方向也不太正确，但是这些思想很可能会成为发现更重要观点的转折点。人们常常抱怨说“某些地方简化只会增加其他地方的复杂性。”真的是这样吗？

在我进行这种近似堂吉诃德式的追求时，模式却以我前所未见的形式发展起来。它使网络的复杂性大大降低，其结构比我们想象的要简单。事实证明像多链路、移动性和可伸缩性这样的功能是最终结构的产物，它们不会增加复杂性，而且不需要复杂的机制。该结构也让其他问题的解决方案变得更简单、更直接。从表面上看，出现的情况似乎与我们的想法相同，因此有人只凭第一印象就说“是的，我们都知道。”但随着研究的深入，情况则完全不同，这时就需要进行认知转换。要进行这种转换很难，因为不是所有与转换相关的关键概念都为大家所熟知。

除了系统化原理和经验法则之外，还有一些未解决的重要问题，理解这些问题需要一种自由的思想，但这种思想在产品开发或标准审议中是不可能出现的。

我经常半开玩笑地说“ARPANET 最大的问题就是我们一开始就准备得太充分”。意思就是说，对于以前没有经验的项目而言，对于能否运行还存在大量质疑的项目而言，有一些杰出的观点，这就“足够”了，不需要解决它存在的所有问题(也就是说不要钻牛角尖)。早期 ARPANET 最显著的现象之一就是多次出现“油和水”的问题，他们找到了一种完美的简单综合，同时没有走极端。在这种综合中，极端“只是”一般情况(同时告诉我们以前

不理解的东西)[1]。

人们会作各种初次尝试，有些是错误的，有的走了一些捷径，有些领域没有研究到等等。但即使是这样，网络发展的还是比预期要好。ARPANET 几乎立即从研究课题变成了必需且有用的资源[2]。

在网络的开发过程中，不变的导向隐喻是操作系统。我们总是通过研究操作系统，来了解问题的解决方案和应该构建什么(许多人将 ARPANET 的成功主要归结为它是由人们使用操作系统构建的这一点，而不是通信，遗憾的是情况不再如此)。1974 年，随着网络的运行，人们对 Net 实现的功能欢欣鼓舞[3](20 世纪 90 年代早期有些人看到 Net 时，还是有些兴奋)。然而，我们知道它有一些突出问题，也采取了一些仍需进一步完善的权宜之计(每个人在现实项目中都会这么做)。它们是：

- **取代 NCP** 提到 ARPANET，很可能在我们脑海中首先出现的是这样一种认识：主机-主机协议(Host-Host Protocol)不适用于“大型”网络，这里“大型”指的是包含数千台主机。所有主机共享的单独控制信道就是瓶颈所在。协议非常复杂，它与 IMP 子网的联系过于紧密。我们应该使用哪种类型的协议取代它？
- **清除结构** 考虑到操作系统在早期思想中已经出现，所以使用分层来实现功能。然而，很难说 ARPANET 最初实现方式的分层是很清楚的。在设计中仍然有许多串珠[4]。主机-主机协议和 IMP 子网的交互并不清楚。但是到了 1974 年，物理层、数据链路层、网络层和传输层的思想——可能在 CYCLADES 的实现方式中体现得更加明显，它能够清楚区分 CIGALE 和 TS——被广泛接受。后来，我们发现下面四层不是很“合适”，而且有点复杂。但我们仍然不能说已经很好地理解了层的定义[5]。对于异类资源共享网络而言，采用什么体系结构更合适呢？
- **上层** 利用操作系统为向导，我们认为有三种基本应用程序。我们只是在网络中复制了操作系统的结构：固定一个(Telnet)、一个需要进一步研究(FTP)、放弃了一个(RJE)。结果还不坏。人们通常认为上层中有更多我们尚未了解的“结构”。即

---

1 我说“他们”，是因为我当时只是“资历较浅”的毕业生，我也在其中，但我并不想因为这些认识而居功，只是希望能从中学到什么。

2 本书不是 Net 的历史(尽管有大量内容与此有关)。我发现，没有人能够只依据技术论点就能解释事情为什么是这样。

3 与我最近看到的将 Net 用作“对话”的功能相比，事情无非如此。我们将它看作异类资源共享设备，它是试验和生产分布式系统的推动力，这些系统有 Englebart 的 NLS、National Software Works、CCA's Datacomputer, NARIS 陆地使用管理系统，它们利用对用户不可见的美国分布式数据库，通过多系统处理 ERTS 卫星图像，重点使用美国粒子物理学家使用的 Rutherford High Energy Lab 和 UCLA 360/91 等，这些都在 1976 年以前出现。

4 在科学中，这是常见的事务状态，从一种范例转换到另一种范例的第一步跟两者都有挂勾。“串珠”表示电话公司网络模型，例如 X.25、ISDN、ATM 和 MPLS，它们在 1970 年之前就存在，今天仍然存在。

5 实际上我们还没有，就像课本作者喜欢指出的那样(如果对体系结构讨论有任何暗示)。

使有些人认为有 Telnet 和 FTP 就足够了[6]，但还有人对其他应用程序有各种各样的想法。我们需要更好地理解哪些应用程序有用且如何构造上层，以及它们怎样与系统的其他部分一起运行。事实证明，这就是操作系统模型不适用的地方，这三个应用程序都是特例。奇怪的是，在 Net 开发中这可能是重要的接合点。那么上层是什么样子？

- **应用程序名称和目录**　在开发早期，操作系统模型告诉我们，应该有应用程序名称和网络地址。和操作系统一样，应用程序名应该与位置无关，而地址应该依赖于位置。但实际上，当宣布著名的套接字被用作权宜之计而不是用来定义应用程序和目录时，我感到有点失望，这一点可以理解。提出命名计划并构建目录需要大量时间。我们只有三个应用程序，在各主机中只有各应用程序的一个实例。应用程序名和目录并不是目前急需的。最后，在大量应用程序出现之前，我们必须重新定义它。在网络中命名和寻址是什么样子？
- **多链路**　通过让主机地址变成 IMP 端口号(例如命名接口而不是节点)，路由选择算法不能说明这两个地址是同一个地方，这是我们的第一个错误[7]。但是解决方案也显而易见!如果使用操作系统模型，很明显我们需要逻辑地址空间而非物理地址空间，对于节点和接口需要单独的地址空间。问题是不知道这些地址空间是什么样子。从操作系统中可以知道，命名和寻址是一个难题。如果对这个难题理解正确，许多事情就会变得很简单；如果理解错误，事情就会变得更难。我们知道正确和错误只有一线之隔，因此必须小心谨慎。这种“逻辑”寻址的本质是什么？
- **位置相关的地址**　我们根本不知道位置相关对网络地址意味着什么。在操作系统中这是一个简单的问题，内存地址的位置相关很容易理解，在网格上构建城市也很好理解。但数据网络通常不是规则的网格。在没有路由相关的普通网状网络中，位置相关的意义并不清晰。它与网络图没有联系，因为它经常改变。它应该是图的某种抽象化，表示“在哪里”而不是表示“如何到那里”。但不清楚如何抽象图。在网络中，位置相关是什么意思？
- **采用无连接**　ARPANET 起初是面向连接的网络。与 IP 相比，ARPANET IMP 子网与 X.25 有更多共同点，这是首次尝试的合理选择，因为当时我们不知道它实际上如何运行，也不知道如何构建网络。重组、流程控制的经验显示严格控制的确定性网络有严重问题。控制越少(越不可靠)就越有效，这种观点虽然奇怪，但很有意义。CYCLADES 在网络中使用无连接数据报，用不可靠机制创建可靠通信，这

6　邮件是 FTP 中的两个命令。

7　是的，不一定。如果需要支持多余的连接，这就是一个错误，但它没有。很难构建可以移动数据的网络。但这是如何将 Net 作为“产品”的暗示。

非常完美、简单而又令人信服[8]。然而，还需要更好地理解无连接模型的行为。因为这只是在相对较小的网络中以较低的带宽使用它，随着范围的增大，还要更好地理解其运行原理。毕竟，任何事物的单纯形式在现实世界中能正常运行的情况还是很少见。无连接的新范例很简单，并且为取代主机-主机协议提供了一些概念。我们需要深入理解连接模型和无连接模型之间的差别。尽管我们对无连接很有兴趣，但也不得不承认连接有时也很有意义。然而，我们有些人(包括我自己在内)需要很长时间才会承认这一点。无连接模型的属性是什么？它与连接有什么关系？它如何满足生产系统要求？有没有一种模型能够将二者合并为一般情况？

这些是我们从 ARPANET 转向 Internet 时网络存在的一些主要问题。下面讨论 Internet 如何解决这些问题。

## 0.2 聚焦 TCP

主要有 4 个竞争者想取代 NCP：

(1) XNS–序列分组　它与(2)CYCLADES TS 协议类似，是一种分组序列化的动态窗口传输协议，包含多种 PDU 类型、建立、释放、确认和流程控制。XNS SeqPkt 和 CYCLADES 都分离了传输函数和网络函数，这与 TCP 和 IP 类似。

(2) Delta-t　它由 Lawrence Livermore 实验室开发，是协议的全新思想，包含更健壮的基于计时器的同步机制，这种机制基本消除了连接建立，将单独的 PDU 类型用于确认和流程控制。Delta-t 也分离传输函数和网络函数。

(3) TCP　字节序列化的动态窗口传输协议，包含一种 PDU 格式和控制位来区分各种状态变更。它还允许单独释放两个单信道。在最初的版本中，TCP 没有分开传输函数和网络函数。

TCP 还有一些特殊功能：

- 用单个 PDU 格式来简化处理，而不是用额外的代码分析一些不同的 PDU 类型。我们希望这样可以简化各分组处理并且节省代码空间。如果处理器速度低下，这一点就值得关注[9]。这种方法似乎向简化迈进了一步，但更好地理解协议之后，我们发现事实并非如此。另外，将这些控制位作为实现方式中的控制位，会增加代码的复杂性。我们建议当前实现方式将它们作为 opcode 看待。实际上，通过 Net

8　面向连接的分组交换模型是解决问题的简单甚至是显而易见的方法，而无连接模型是思想上充满灵感的转变。

9　这令人难以置信，但在 1974 年，设计的数据协议非常少，它们各不相同，与今天相比更是这样。

中的通信量统计就会发现，很明显 syns、fins 和 acks 被看作不同的 PDU 类型(例如 40 个字节的数据包数量)。

- 单个 PDU 格式也比搭载确认强大。计算显示搭载将系统开销减少 35%到 40%。之所以会出现这种情况，是因为当时 Net 上的大多数通信量是由 BBN Tenexes(Net 上的当时主导系统)实现的一次一字符回应的 Telnet 通信量。然而，因为现在的 Net 上没有太多 Tenexes，因此节余在 10%以下，可以忽略不计[10]。
- 1974 年，TCP 在某些领域有用，在其他领域则是沉重的负担。例如，带宽限制仍然很常见，有时标题大小也会成为问题。今天它的优势已经消失了，它不能很好地适应更大范围的操作，不能满足现代网络的要求。Delta-t 或者 TS 可能是更好的选择。它们不仅适合于当时的环境(delta-t 在 DoE 中已使用了好多年)，而且不需要显著改变其结构就可以适应现代网络的要求。

如第 3 章“协议中的模式”所述，这类协议的一般结构通常为一个管道数据传输部分与多种用途的计算部分松散耦合——而这部分需要将同步用于与簿记关联的错误和流程控制。单个 PDU 格式使这种结构变得复杂，也使协议难以符合不同应用程序的要求。因此单个 PDU 格式意义不大。所以 TCP 最符合 20 世纪 70 年代中期的特点。

为什么选择 TCP？理由有很多。当时，除了 delta-t 同步机制之外，其余四种协议之间的差别不是很大，每种协议都不可能得出压倒性的结论，也就是说，没有哪个协议是必然选择。人们希望不管做出哪种选择，都能够在研究工作中使用几年，然后用别的选择取代。毕竟，NCP 第一个尝试构建网络，TCP 就是在这种新方向上的首次尝试，人们并不指望第一次就能做好。至少，需要更多的尝试才可能“做好”。然而，选择 TCP 的首要因素是，Internet 是 DoD(美国国防部)项目，而 TCP 由 DoD 出资赞助。这只不过反映了大型机构部门内部相互竞争的实际情况，因为大多数评论家都是 DARPA 承包商。

**分离 IP(对于寻址而言没有什么新内容)**　从 TCP 中分离 IP 似乎是一种必然，它因为传输协议和 IP 实现不同的功能(请参考第 6 章“探究层”)。创建 IP 时唯一遗憾的是，它对多链路问题毫无帮助。IP 继续命名接口，这可以理解，它在 1975 年分离出来之后不久，人们就意识到了这个问题。虽然我们知道多链路问题和这个问题理论上的解决方案，但对于寻址还是不太清楚。然而，Internet 地址命名子网附着点，这的确让我们感觉不舒服。

NCP 渐渐被淘汰。最终经过 8 年的研发，到 1982 年部署 TCP。Internet 首次(可能也是最后一次)从 NCP 转向 TCP。同时(20 世纪 70 年代晚期，20 世纪 80 年代早期)，为了支持标准接口，人们逐步淘汰了“著名的”BBN 1822 Host-IMP 硬件接口。对于连接分组交换的主机而言，在大多数情况下，选择 IP over X.25；在其他情况下，选择新发明的 Ethernet。NCP 服务已超过十年，比所有人预期的时间都长。

---

10　做数学。20 个字符输入和 40 个字符输出是当时可接受的平均终端通信量。

**Saltzer 论寻址** 1982 年，MIT 的 Jerry Saltzer 发表了一篇论文，该论文讨论了计算机网络中的命名和寻址问题。Saltzer(1982)概述了网络为什么必须有应用程序名，而该名称仿效并映射节点地址，节点地址又映射附着点地址，附着点地址又映射路由，这些都是完整的寻址体系结构必不可少的元素[11]。剩下的就是要弄清楚位置相关在图中的含义。所有人都引用这篇论文并认为它是正确的，但都没有实现它。公平地说，关于他提出的抽象化如何应用于现有 Internet 以及位置相关在图中是什么含义，Saltzer 并没有给出答案。

**主机表不适用——采用 DNS 但没有应用程序名或者目录** 从 ARPANET 开始，网络信息中心(Network Information Center，NIC)就保留了一份当前主机列表及其相应 IMP 地址的文本文件。每隔数周，就会下载文件的最新版本。然后数周变成了每周，再然后变成每隔几天，大约到了 1980 年，它已经变得很难作为简单文本文件来进行手动管理了，这与 Internet 的不断增长紧密相关。因此，现在正是适当安排应用程序名称方案和目录、着手解决寻址问题的时候。但在 Net 中仍然只有三个应用程序，每个主机只有一个，此时仍然不需要目录。所有人都很满意过去 15 年的这种做事方法[12]。因此，DNS 只是作为分布式数据库的层次结构来解决 IP 地址的同义词问题，并取代原来的主机表。采用这种方法部分原因是由于强加了从 ARPANET 开始的命名主机思想(仔细分析网络中的命名就可以知道，命名主机与通信所需的寻址无关)。只要有著名的套接字，而且每个主机中只出现一个应用程序，那么 DNS 就是所需的“目录”：这是保持用户友好形式的 IP 地址的一种方法。虽然从 20 世纪 70 年代早期就开始讨论目录，但一直没有定论。我们一直采取的态度是引入的变更不要超出解决当前问题所需的变更。是这种谨慎的做法在维持现状吗？

**拥塞崩溃** 1986 年，Internet 遇到了最严重的危机，网络遭遇了拥塞崩溃。吞吐量先急剧上升后陡降的典型拥塞曲线每天都会出现。拥塞导致的长时间延迟造成了超时，进而导致重传，这让问题变得更糟糕。虽然无连接模型在 20 世纪 70 年代早期就闹得满城风雨，但 ARPANET 基本上还是面向连接的网络(除非显式使用 Type 3 消息)。甚至是在转向 IP 之后，到分组交换机和路由器的许多主机附件仍然使用 BBN 1822 或者 X.25，两者都是流程控制主机。因为越来越多的主机附件没有流程控制的无连接 LAN。1822 和 X.25 渐渐被淘汰，所以网络中的流程控制越来越少，只有 TCP 中才有流程控制。TCP 流程控制会阻止发送的应用程序超出目标应用程序，但不会阻止网络中的拥塞，因此拥塞崩溃不可避免。没有人试验过无连接网络扩大时的属性[13]。

对于这种主要危机，必须做点什么，而且要快。由于危机的存在，Internet 基本上不能

---

11 当 Saltzer 写这篇论文时，只有一种我们需要的改进，因此，他没有考虑到这一点并不奇怪。

12 有人假设这样完成，这并不奇怪。在计算中，15 年前几乎是 10 代——古老的历史!

13 需要实验网络，构造一些小型网络，但它们不够大，无法研究这些问题。它实在太昂贵。没有人愿意资助模拟大型网络，更不用说还有些恶意批评者质疑这种模拟是否有意义。

用，但处理危机比维持和运行网络还要困难。根据 Weiner 的控制理论，应该使用被控制的资源定位反馈。但网络中所有交换机上都可能出现拥塞。包含拥塞控制就意味着使用连接模型，而不是无连接模型。首先，面向连接的设计不能很好运行，并且存活性较差。第二，过去 15 年，网络团体在与电话公司的连接和无连接之争中战胜了对手(参见第 3 章)。我们无法承认失败，我们认为自己没有错[14]。许多人相信有中间立场，但到目前为止还没有人找到它，所有提议都不是走向这个极端就是那个极端。

Van Jacobson 提议在 TCP 中插入拥塞避免方案。它由现在著名的慢启动组成，每个往返都加倍拥塞窗口，直到检测到拥塞为止(然后呈指数下降)。实际上，拥塞避免创造了拥塞，但随后消除它。这种解决方案维护无连接模型，能够快速解决拥塞问题，而研究人员试图理解如何进行拥塞控制，并保持无连接网络的基本属性。而且，此时改变 TCP 实现方式比重新设计所有环节更容易。也许重要的是，这个接合点也标志网络中从离散计算缓冲区的流程控制到连续控制理论机制的质变。然而，这个危机过后，没有人再去寻找完整的解决方案。因为有太多人为因素，所以基本原理似乎是为了证明这为什么是“正确的解决方案”[15]。但因为拒绝拥塞控制，所以今天我们有一种共识，那就是拥塞控制属于 TCP。但这不是权宜之计吗？导致拥塞崩溃的条件还会出现吗？会采取什么措施？也许是一个产生大量通信量的优秀应用程序，但不使用 TCP 吗？如果 Net 上的通信量没有使用 TCP，情况又如何？像视频一样吗?

**SNMP**　ARPANET 有好的网络管理，但它是运行 Net 的 BBN 内部的功能[16]。在 20 世纪 80 年代早期，随着建立的公司网络越来越多，网络管理逐渐变成了人们重点关注的主题。到 20 世纪 80 年代中期，使用 IEEE 802.1 管理协议的经验表明，元素化“图灵机”方法[17]尽管简单直接——但是并不够。当时人们知道网络管理的关键就是管理更少的协议和更多的系统对象模型。Internet 团体使用两种方法：简单的类图灵机、轮询[18]协议、不包含面向对象特性的 SNMP；更复杂且可扩展的面向对象、事件驱动协议、HEMS。重要的是，和针对问题提供创新式解决方案的 ARPANET 不一样，20 世纪 80 年代末期的 Internet 未进行创新，而是采用 SNMP。当时重点强调的是透明简化，假设会产生更少的代码，避

---

14　他们不是。

15　当时，很少有网络人员具备扎实的控制理论背景，很少的人熟悉这些问题，因此对于改变运行都保持沉默。

16　这些故事是传说：BBN 打电话给 Pacific Bell，告诉他们说从 Santa Barbara 到 Menlo Park 的 T1 线路出现了问题，但 Pacific Bell 相信他们是从 Santa Barbara 或者 Menlo Park 打的电话，而不是从 Boston 打的电话。

17　所有事情都使用 Set 和 Get 属性完成。

18　在 SNMP 中使用轮询总是很复杂。在 ARPANET 中，轮询被看作一种不能伸缩的强力方法，并表示大型机思想。这是一种禁锢思想。从没有考虑这一点，当时提出轮询的人会赚大钱。

免使用看起来过于深奥的概念[19]。事实证明，SNMP 实现方式要大于 HEMS 或者 CMIP[20]。而其根本结构和缺乏面向对象支持，以及我们将在第 4 章“上层体系结构研究 ”中讨论的转移注意力的内容，是 Internet 中开发管理的主要障碍。

**Web** 在 20 世纪 90 年代早期，Web 开始广泛使用。它存在了一段时间，但在伊利诺斯州大学的 NCSA 为其扩充浏览器之前，它只是 Gopher 的另一种版本。该超级计算机中心的主要任务之一是研究如何更有效地提供数据。作为该任务的一部分，有一位编程人员突然想到将 GUI 放在 Web 上，这样让页面上的所有对象都变得“可单击”。Web 的广泛使用，对 Net 提出了新的要求。

Web 变成了 20 年来网络上一个新的应用程序，也正如人们所料，它产生了许多新问题。首先，这是第一个不是来自操作系统隐喻的应用程序。对于 Web 而言，协议和应用程序不是一回事。可能有多个使用 Web 协议的应用程序，在相同主机上可能同时有相同应用程序的多个实例。因为没有合适的应用程序命名结构，所以 Web 必须开发自己的命名方案——现在普遍存在的 URL。然而，这也不会让人们深入思考命名要求所需的结构。相反，人们对使用通用资源名称(Universal Resource Names)来扩充现有方案更感兴趣。

进行网络管理时，我们关注的是短期如何解决特定问题，很少关注它对一般问题有什么启示。

**Ipng** 20 世纪 90 年代早期，Internet 飞速发展。由于发展迅速，因此出现了 IP 地址短缺，尽管人们更关注的是不断增长的路由器表的大小。IAB 启动了一个项目来确定行动路线。在全面考虑开发新协议或采用现有协议的利弊之后，他们采用了双管齐下的保守方法，用称为 CLNP 的 OSI 版本取代 IP。该保守方法由固定分发的地址数的 IANA 构成，使用私有地址，创立 CIDR 促进路由聚集，要求大多数地址请求通过主要提供商实现到 CIDR 的转换。

提议采用 OSI 协议引起了巨大的骚动，导致 IAB 推翻了自己，IPng 进程开始选择新协议。人们起草了可接受 IPng 的要求，其中要求地址继续命名接口，而不是节点(尽管自从 1972 年，我们就知道网络地址(更不用说内部网络地址)不应该命名子网附着点)。从根本上说，IPv6 解决的唯一问题是延长了地址。而它对阻止路由器表的增长无济于事，对于寻址体系结构中的缺陷也无济于事[21]，它只会让事情变得更糟糕。而且，过渡到 IPv6 计划需要网络地址转换(Network Address Translation，NAT)。事实证明，网络的所有者因为其他

19 下推式自动操作、面向对象等。有一种强烈的反理性态度(在某种程度上)：真正的编程人员“需要书本知识”。他们天生就知道如何设计和编写代码。

20 OSI 管理协议，它是事件驱动和面向对象的协议。

21 看到 IETF 采取延长 IPv6 地址的方法来弥补自己的不足，我很伤心。

原因喜欢 NAT。有了 NAT 和私有地址空间之后，就没有理由采用 IPv6。如果选择 IPv6 组来解决寻址问题，并且正视 IPv4 不是 Internet 协议这一事实，就可能解决问题并避免使用 NAT。

IETF 为何无法解决这个已经存在 20 年的问题？原因如下：

(1) CLNP 的确能够解决这个问题，但有一种偏执思想认为，如果 OSI 能够解决这个问题，Internet 就不能解决这个问题[22]。

(2) 在 IETF 中很少有人(可能 1000 个人中有 12 个左右)理解这个问题[23]。大学里的老师没有教过在网络体系结构中应该命名什么。实际上，时至今日也很难找到一本讨论这个主题的网络教科书。

(3) 有一个信条是，所有多链路应该属于不同提供商[24]，要么没有对等点，要么它们隔得太远，没有必要使路由选择复杂化(如果可能的话)。关于基于提供商的地址也有一些辩解，但它忽略了 Saltzer 论文的要点——附着点地址是“物理地址”，而节点地址是“逻辑地址”。

**Internet 通信量自相似**　1994 年，Bellcore 小组发布的一篇论文指出各个 Ethernet 上 Internet 通信量的测量方法表现为自相似性。这首次暗示通信量不是 Poisson 分布。其实，自从 20 世纪 70 年代开始，人们就知道这一点[25]。虽然论文中没有说明，但很让人怀疑的是，这并不是 Internet 通信量本身或者 Ethernet 通信量自相似，而是自相似性是 TCP 拥塞控制的人为产物，这一点后面会验证。TCP 通信量比 UDP 通信量更加自相似，Web 通信量不如 TCP 通信量自相似。Web 通信量的低自相似性更像是“大象和老鼠”现象的结果。但有意思的是，TCP 拥塞控制导致混乱行为的结果没有形成如何处理拥塞控制的观点。该团体的一般观点是，这只是无法更改的事实。部分原因在于当前正在流行的观点和有些人的争论——即大型系统都显示自相似行为，因此没有什么可做的。

这将我们带回到了 20 世纪 90 年代早期，当时我刚刚开始这种练习，就像 IPng 刚刚热起来一样[26]。开头提出的 7 个未解答的问题仍然没有答案，仍然浮现在我们脑海中，我没有打算解决这些问题。这些都是令人畏惧的问题，每种出现的模式都可以用是否有助于解决这些问题来衡量。我正在寻找一种清晰的理解，然而，有三个问题必须考虑。经验显示，其中两个问题如果没有解决，就会破坏体系结构。我们已经接触过它们：找到连接和无连接的有效综合、解决命名和寻址(特别是位置相关的含义)问题。连接和无连接之争是许多

---

22　当然，有一些不改变它的逻辑原理，但它不改变基本反应。

23　每隔几个月这种争论就在 IETF 列表上展开。有些人仍然争论说它们应该使用地址。过去 15 年他们什么也没有学到。

24　现实世界中有时是这种情况。

25　问题是爆发的通信量需要新的方法来建模。没有人知道这种方法。

26　我记得有一次出席 IETF 会议，会上重点讨论 IPng，只有基本观点，但还没有完全形成。

灾难的根源，我们迫切需要真正的综合。当然，考虑这 7 个未解答的问题，就会发现深入理解命名和寻址之后就可以解决许多问题。第三个问题来自我 20 年来数百次协议设计的经验，我想分离机制和策略，就像在操作系统中那样——看看会出现什么情况[27]。

分离机制和策略揭示了我以前没有见过的模式，恢复了我 15 年前对模式的兴趣(但当时还没有多大兴趣)。到了 1994 年，这里提出的模型的轮廓就已经很清晰。不是 7 层或者 5 层，只有两个协议的单层以及递归的可选信息。复杂性的减少立即解决了一长串问题。

虽然有些关键问题还有待解决，但绝不只是要寻找解决这些问题的方法，它们只是解开我们所面临的症结的线索。只找到某些有效的方法还不够，这种解决方案必须适应更广泛的“理论”。如果它不能，就要改变解决方案或者改变理论。我知道更重要的是理解问题给我的启示，而不是做我自己认为最好的(有些读者认为我完全遗漏了这一点；其他有经验的人会知道我的意思)。

然而，在 20 世纪 90 年代中期，没有人觉得有必要研究“新的体系结构”，反正我没有这样做。有时我知道问题显示的模式，但它与常规的方向不一致，所以我不能接受。但也不断有暗示告诉我，遵循问题显现的内容可能更好。最后，阻力没有了，问题得到进一步简化，我也获得了新的认识[28]。

出现的是更简单的网络模型，完全隐藏了复杂性。我们很早就知道寻址的大概内容。Jerry Saltzer 在 1982 年就提供了它的基础知识。但对 Saltzer 的理论稍作扩充就会产生一种结果，这种结果与新兴的协议结构完全吻合(例如重复)。这些结果互为补充，这样的情况多次出现。有人说有些事情很难完成，但事实证明在这种模型中完成这些事情非常简单。如果支持的功能不是专门设计的，那么这就说明研究的方向是正确的。

位置相关的问题更难。我们已经知道地址必须依赖于位置，但独立于路由。后来我得出结论：要让地址以某种有意义的方式变得依赖于位置，必须依据抽象化的网络图来定义它们。研究抽象图的数学工具，这就产生了拓扑，并得出结论：地址空间具有拓扑结构。整个 20 世纪 90 年代，我都在和别人讨论这个问题，到 20 世纪 90 年代末，我终于找到了问题的解决方法。

后来，关于协议设计细节的一个临时教学问题让我不得不回顾自己所知道的基本原理，这使我对结构和进一步简化有了新的认识。

那么，这本书能够解决所有问题吗？几乎不可能。但它阐述了基本结构，依据这种基本结构，就可以构建一般的网络理论。它让我们能够跳出目前所处的框框，找到自己的不

---

27 按照这种方法，我遇到了第四次失败，尽管我们着迷于网络通信量剧增的思想，但我们要尽量排除这种剧增。我们似乎遗漏了一点：关于确认和流程控制策略我们有许多著作，但关于多路复用却不多(除了物理层现象之外)。虽然我在这个主题上取得了重大进步，但本书没有讨论，因为它不是“体系结构”问题。

28 这种情况与错误和流程控制协议的结构有关。

足。事实证明遗漏的东西不是很多，但它们是找到简单解决方案的关键。我努力在可读性和形式上寻求平衡，但我的目标之一是找出表示问题所需的最少概念。这种模型非常接近，是一种基本模型，过去 30 年我们所做的大多数工作仍然适用。该模型为我们推理独立于特殊网络或技术的网络奠定了良好的基础。我希望该模型能够启发其他人的观点和思想，并期待着看到新的观点和思想。

如前面所述，有些概念是理解这个模型的关键，但这些概念还不为人所知。我们很大程度上依赖于 Seymour Papert 所说的唯一概念[29]，这些概念值得计算机科学研究：将问题分解、抽象化、递归。抽象化已经被废弃几十年，但这里我们仍然要好好利用它。而且，我们认识体系结构需要大量认知转换。因此，本书就是要将读者从我们所知道的方式转变为一种新的研究事物的方式。即使这样，读者要理解本书也不容易，先必须进行深入思考。

首先我们回顾了基础知识，重申了通信所需的最低假设条件和进行抽象化所需的工具。第 2 章和第 3 章讨论了我们所熟悉的协议以及分离机制和策略。其中，出现的新模式表明可能只有三类协议，后来我们发现其中一种更像是“公共标题”而不是协议。我们还在解决连接和无连接之间的冲突方面取得了许多进展[30]。

第 4 章回顾了有关“上层”的认识，罗列了我们做得对的事情和需要避免的事情。奇怪的是，我们发现了一些关键概念，它们对于构造基本模型非常有用，但同时又得出结论说没有“上层体系结构”。第 5 章“命名和寻址”详细讨论了非常难以理解、非常深奥的主题——命名和寻址。我们特别提到了 Saltzer 在 1982 年发表的论文，说明为什么当前痴迷于“位置 / id 分离”问题没有出路。到了第 6 章，我们已对问题和所需元素有了比较好的理解，并且可以考虑如何将这些元素组合成系统。在该章中，我们做一个简单练习(我们中的任何人在过去 30 年间的任何时候都可以做这种练习)，只是发现它产生了我们一直在寻找的结构。本章是一切问题的关键所在。

在第 7 章“网络 IPC 模型”中，我们完成了一项艰难的任务，将前面 6 章讨论的所有部分组合为新模型，并讨论这种模型的运行。当我们定义所需的概念时，必须仿效 Johnson 所称的“无害苦工”。这是一项棘手的工作，但必须完成。我们讨论了新节点如何连接网络、如何启动通信。第 8 章“让地址拓扑化”重新回到命名和寻址，讨论位置相关的含义以及如何搞清这个概念的含义。在第 9 章“多链路、多播和移动性”中讨论了在这种模型中如何表示多链路、移动性和多播/任意播，并讨论了这种模型的一些新结果。第 10 章“退出死胡同”是回顾，讨论了导致这 7 个基本问题 250 多年来未得到解决的原因，并对未来做了展望。

---

29　我希望可以引用它。Seymour 确认他说过，但不记得在哪里说过，我找不到!

30　我们没有解决无连接缩放的问题，因为严格来说这不是体系结构问题，尽管这里提出的结构对解决方案有帮助。

# 目　　录


第 1 章　网络体系结构基础 …… 1
1.1　导言 …… 1
1.2　起源 …… 3
1.3　抽象级别 …… 5
1.3.1　模型 …… 7
1.3.2　服务 …… 7
1.3.3　协议与接口 …… 9
1.3.4　实现方式 …… 10
1.4　指定协议 …… 10
1.4.1　非正式规范 …… 11
1.4.2　形式描述技术 …… 11
1.5　后续章节内容 …… 13

第 2 章　协议元素 …… 15
2.1　简介 …… 15
2.2　协议体系结构 …… 15
2.3　数据单元 …… 22
2.3.1　构造协议 …… 25
2.3.2　PDU 的大小 …… 27
2.3.3　机制和策略 …… 28
2.3.4　QoS 与 NoS …… 31
2.4　数据传输机制的简略类别 …… 31
2.4.1　定界 …… 32
2.4.2　初始状态同步 …… 32
2.4.3　策略选择 …… 33
2.4.4　寻址 …… 33
2.4.5　流或连接标识符 …… 33