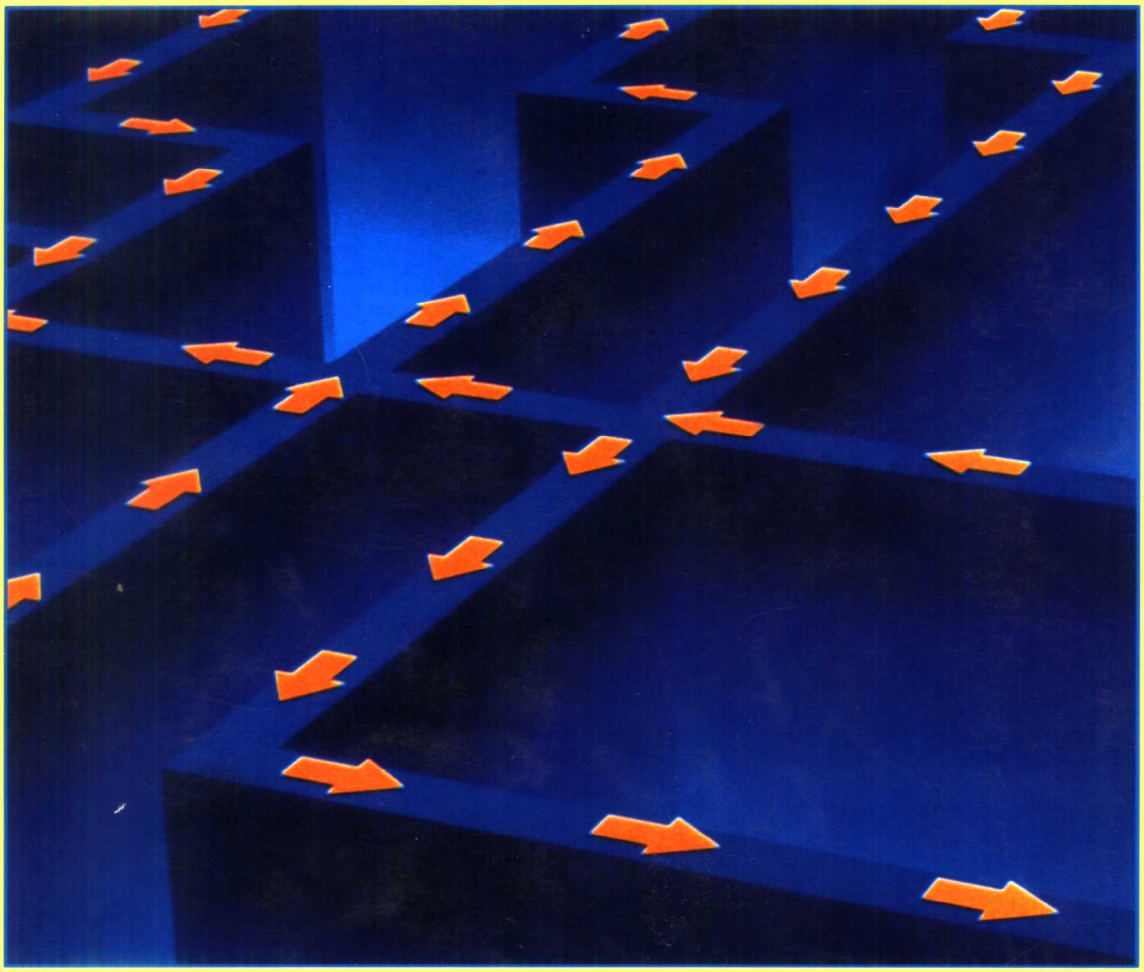
OSPF 协议完全实现

OSPF Complete Implementation



John T. Moy 著
闵春平 李建成 钱言琮 译

Addison-Wesley
附赠光盘
中国电力出版社
www.infopower.com.cn

OSPF

协议完全实现

OSPF Complete Implementation

John T. Moy 著
闵春平 李建成 钱言琮 译

中国电力出版社

内 容 提 要

OSPF 协议已成为目前广域网和内联网采用最多、应用最广泛的路由选择协议之一。本书由该协议的开发者编著，具有很高的权威性。书中以大量的实例详细介绍了具体实现的软件体系结构，深入阐述了 OSPF 的功能。主要内容包括 OSPF 层次结构、移植指导、IP 路由表、链路状态数据库、路由选择计算、MOSPF 实现、配置和监控以及主机路由侦听等。

本书适合 TCP/IP 网络管理员、协议设计者和网络应用的开发者阅读。

图书在版编目（CIP）数据

OSPF 协议完全实现/（美）莫艾著；闵春平等译 —北京：中国电力出版社，2002.7
ISBN 7-5083-1104-3

Ⅰ.O… Ⅱ.①莫…②闵… Ⅲ．计算机网络-路由选择-通信协议 Ⅳ. TN915.04

中国版本图书馆 CIP 数据核字（2002）第 041232 号

著作权合同登记号 图字：01-2002-0701 号

中国电力出版社出版、发行
（北京三里河路 6 号 100044 http://www.infopower.com.cn）
汇鑫印务有限公司印刷
各地新华书店经售
*
2002 年 8 月第一版 2002 年 8 月北京第一次印刷
787 毫米×1092 毫米 16 开本 20.5 印张 460 千字
定价 45.00 元

前　言

本书是《OSPF：Anatomy of an Internet Routing Protocol》（译注 1）的姊妹篇。为了与 Internet 的价值传统“达成一致，代码可用”保持一致，本书提供了一个完整的 OSPF 实现，并遵守其姊妹篇中阐述的 OSPF 协议规范。书中的 OSPF 实现采用 C++语言写成，可移植性好，而且提供了两个可移植实例：① OSPF 路由选择守护进程——**ospfd**，用于 Linux 操作系统；② OSPF 路由选择仿真器——**ospf_sim**，可在 linux 或 Windows 下运行。

本书正文提供了 OSPF 实现的设计文档、具体移植工作的指导说明以及两个移植实例的用户使用手册。OSPF 实现的数据流和主要的数据结构也在本书中给予阐述说明，并在必要时辅以代码块说明。OSPF 完全实现的代码附在本书附带的光盘内。

通过考查 OSPF 实现，本书对协议的细节进行了深入分析。同时本书还对优化 OSPF 实现的方法进行了阐释。书中所附习题供有兴趣的读者用来练习修改一个相当庞大的实时分布式软件系统，并从中获取经验。

另外值得一提的是，本人从多年编写网络软件的实践中还学习到一点，那就是做任何一种事情都不止一种方法。因此读者绝对不要认为本书所提供的 OSPF 协议实现方法是惟一可行的方法。实际上，读者从本书中应该学到的是实现网络软件的一些新技术和 OSPF 路由选择选择协议的优点。

读者对象

与《OSPF：Anatomy of an Internet Routing Protocol》一书一样，本书主要针对那些对 Internet 路由选择感兴趣的人们：数据通信专业的学生、TCP/IP 网络管理员、协议设计人员、路由选择协议软件开发人员以及其他致力于 TCP/IP 网络的设计、开发和管理的专业人员。另外，通过本书所附的练习，软件工程设计人员也可以在修改并强化一个相当庞大且复杂的实时软件系统的过程中获得很多经验。

因为本书还包含一个工作中的 OSPF 实现，它能够将一个 Linux 工作站转换为一个路由器或者一个 OSPF 网络仿真器，因此本书对那些从事与 OSPF 网络管理、设计和监控相关的非专业程序设计人员也很有参考价值。

本书写作时，认为读者已具有一定的 OSPF 基础知识。同时，读者也可以从本书的姊妹篇《OSPF：Anatomy of an Internet Routing Protocol》一书或 OSPF 协议规范中获得

译注 1：该书中文版《OSPF 协议剖析》已由中国电力出版社引进出版。详情请访问：http://www.infopower .com.con。

相关的基础知识。

本书的组织结构

读者可采用多种方式阅读本书。那些仅对如何使用 **ospfd** 完全实现或对 OSPF 仿真器感兴趣的读者，可以只阅读第 1，2，13，14 和 15 章；对于那些需要将 OSPF 移植到其他操作系统的读者，只需集中阅读第 4 章以及第 14，15 章中所描述的两个样例移植即可。

本书的其余章节，即第 3 章和第 5~12 章，详细叙述了 OSPF 完全实现及其一些扩展功能的实现。阅读这些章节，要求读者具有一定的 C++编程语言基础。这部分的每一章都讨论一个 OSPF 功能，例如 LSA 泛洪。每一章的开始首先对 OSPF 规范作一必要的阐明，描述 OSPF 实现所提供的新功能、提高效率的措施，然后通过考查代码样例来深入说明其功能实现。本书图 7.2，是《OSPF: Anatomy of an Internet Routing Protocol》中图 6.6 的拷贝，它贯穿在本书的所有例子中。本书每章结尾的练习用来强化该章所讨论的观点，并允许读者向 OSPF 实现中添加功能。本书没有提供练习的答案，实际上那些标记为 bug 补丁的习题的答案可以在 OSPF 实现的源代码中找到：**http://www.ospf.org/software/ospfd**。

第 1 章，功能说明。该章介绍了 OSPF 的功能以及已经由附带软件实现的和还未实现的扩展功能。另外，本章对两个移植样例进程做了简要介绍，即简要介绍了用于 Linux 操作系统的路由选择守护进程 **ospfd** 和 OSPF 路由选择仿真器 **ospf_sim** 这两个软件系统。

第 2 章，安装指导。本章阐述了如何在 Linux 操作系统上安装 OSPF 路由选择守护进程 **ospfd**，以及如何在 Linux 和 Windows 下安装 OSPF 路由选择仿真器 **ospf_sim**。

第 3 章，软件体系结构。本章首先详细介绍了 OSPF 完全实现的软件体系结构，包括输入、输出和贯穿 OSPF 实现的数据流。然后简单介绍了主要的数据结构及它们之间的相互关系。本章最后阐述了所附光盘中的源文件组织结构。

第 4 章，移植指导。该章展示了如何将 OSPF 软件移植到各种操作环境中去。本章还对 OSPF 完全实现与操作系统之间的软件层次作了解释。另外，移植时的一些特殊考虑，如如何处理各种类型的 CPU 芯片等，也在本章予以介绍。

第 5 章，构造模块。本章描述了 OSPF 软件使用的各种实用函数。这些实用函数是与软件一起提供的，包括 AVL 树、Patricia 树以及优先级队列。另外，计时器、日志消息和 IP 路由表的实现也在本章给予说明。

第 6 章，链路状态数据库。本章描述了 OSPF 实现中链路状态数据库的组织结构，以及对链路状态数据库的各种操作，如 LSA 的老化等。

第 7 章，创建 LSA。本章叙述了 OSPF 实现如何创建 LSA，包括每一个特定 OSPF LSA 类型的构建。然后讨论了 LSA 创建的速率限制、LSA 的更新以及从链路状态数据库中如何泛洪 LSA。

第 8 章，相邻路由器维护。本章描述了 OSPF 相邻关系的发现和保持过程。同时本章还对相邻路由器之间链路状态数据库的初始同步化以及接口状态变化的处理作了介绍。

第 9 章，泛洪。这一章详细介绍了通过可靠泛洪算法来保证 OSPF 链路状态数据库的持续同步机制。

第 10 章，OSPF 层次结构。本章首先讨论了 OSPF 区边界认证的限制。然后介绍了跨越区边界泛洪路由选择信息的方法，以及外部路由导入 OSPF 路由选择域的方法。

第 11 章，路由选择计算。本章介绍了生成 IP 路由表表项的基本的 OSPF 路由选择计算，接着阐述了触发路由表计算的各种事件，以及能够加速路由选择计算的各种链路状态数据库操作。路由表计算包括区间、区内以及外部路由的计算。

第 12 章，MOSPF 实现。本章介绍了 OSPF 组播扩展功能的实现，即 MOSPF 实现，包括 MOSPF 与 IGMP 的交互作用、group-membership-LSA 的生成以及 MOSPF 路由选择计算。

第 13 章，配置和监控。本章主要阐述 OSPF 完全实现是如何进行配置的，具体叙述了配置参数的完整列表，以及动态改变某一配置参数时所引起的相应后果。同时本章还介绍了配置请求的处理机制和关闭 OSPF 软件时的文明退出过程。

第 14 章，一个 Linux 路由选择守护进程。本章介绍了 OSPF 软件的第一个移植样例，即用于 Linux 操作系统的 OSPF 路由选择守护进程 **ospfd**。**ospfd** 路由选择守护进程与标准的 **routed** RIP 路由选择守护进程相类似，后者大多用在基于 unix 的操作系统上。如果希望用 OSPF 代替 RIP，则应该尝试使用 **ospfd**。另外，本章对 **ospfd** 的配置、监控和调试方法也作了介绍。

第 15 章，OSPF 仿真器。本章介绍了 OSPF 实现的另外一个移植样例，即运行在 Linux 和 Windows 下的 OSPF 路由选择仿真器 **ospf_sim**。同时本章还介绍了配置并运行该仿真的方法，它是一种基于窗口的 Tk / Tcl 应用程序。

书后还有一些附录。附录 A 是 OSPF 软件发行版中所包含程序的 UNIX 风格的用户手册页。OSPF 软件生成的日志消息见附录 B。读者感兴趣的 OSPF 软件扩展项目列表见附录 C。附录 D 提供了 GNU GPL（General Public License，通用公共许可证），其中涵盖 OSPF 完全实现。附录之后是按字母顺序排列的参考文献。例如在正文中，引用[75]对应参考文献中的第 75 条。

bug 修正

本书所提供 OSPF 软件是 Release0.1 版。尽管笔者已努力测试了尽可能多的功能，但是毫无疑问，软件中仍有许多 bug。如果发现了 bug，可以向**ospfd-bugs@ospf.org** 发送 bug 报告。同时可以访问**http://www.ospf.org/software/ospfd** 获得 bug 修正，但有时修正由于种种原因并不那么及时，请读者谅解。

源代码版权

致谢

首先要感谢本书的技术审校人员，是他们及时而细致的审稿使本书更趋完美。他们是 Robert Minnear，Patrick W. Murphy，Matthew G. Naugle，Mark J. S. Paton 和 John W. Stewart，III 等。

还要感谢在此书编写过程中一直给我帮助的 Addison Wesley 的编辑们：Karen Gettman，Mary Harrington，Sarah Weaver 和 Karen Wernholm。同时感谢 Tim Kinch 为本书绘制了所有的插图。

同时还想感谢那些自愿参加 OSPF 软件测试的人们，因为早期我们的软件有很多 bug，但是他们仍然耐心而热情地对软件进行了测试。他们是：Fredi Ammann，Edoardo Calia 以及 New Hampshire 大学 InterOperability 实验室的 Kimo Johnson，Ray LaRocca，William Lenharth 和 Danut Maftei。特别地，我要感谢 Kimo，如果没有他对该项目所进行的彻底、创造性的测试，该项目恐怕永远不能完成。

另外，特别感谢我的妻子 Sonya Keene，感谢她为本书设计了封面，并在本书写作的几年中给予我的极大的耐心。

J.M.

目　　录

第 1 章

功 能 说 明

本章介绍性地列出了所附的 OSPF（Open Shortest Path First，开放最短路径优先）实现的主要功能特点。同时也对 OSPF 实现的两个移植样例，即 Linux 下的 OSPF 路由选择守护进程与 OSPF 路由选择仿真器作了简要的介绍。

1.1 功能特点

本书的 OSPF 源代码完全实现了 OSPFv2 规范 RFC 2328[75]。所有的 OSPF 接口类型广播型（broadcast）、点到点型（point-to-point）（包括编号和未编号的）、非广播多路存取型（NBMA，nonbroadcast multiaccess）、点到多点型（point-to-multipoint）以及虚链路（virtual link）都在 OSPF 实现中得到支持。同时 OSPF 实现也完全支持 OSPF 各种区划分，这些区包括存根区（stub area）、区边界（area border）上可配置的聚合（aggregation）以及虚链路。在配置控制下，任意多的外部路由都能够通过 AS-external-LSA（AS，Autonomous System，自治系统；LSA，Link-state advertisement，链路状态通告）导入。当存在多个等代价路径（equal-cost path）通往某一个目的地址时，这些路径都会被发现，并被试图装入系统的 IP（Internet Protocol）路由表（routing table）（见 11.3 节）。LSA 内非零 TOS（Type of Service，服务类型）值的度量值（metric）将被正确解析（parse），但是会忽略度量值本身。

附录 C 的 RFC2328 中提到的所有配置参数都是可以动态配置的，同样 OSPF MIB[5] 内所有与 RFC2328 功能性质有关的可写 MIB（Management Information Base，管理信息库）变量也都是可以动态配置的。但有一个例外，即可配置参数 **RFC1583 Compatibility**，它总是被置为 **false**（见 11.5 节）。

所有的配置参数都可以动态配置，也就是说配置的改变会马上在 OSPF 执行中生效，并尽可能小地导致中断发生。例如，可以改变一个接口的 OSPF 区 ID 号（OSPF Area ID）。当发生这种情况时，接口状态将发生切换，从而该接口上的相邻路由器（neighbor）之间

将重新建立邻接关系（adjacency）。参见第 13 章中关于本 OSPF 完全实现对各种配置变化响应的行为的完整描述。

一个文明关闭进程（graceful shutdown procedure）也在本 OSPF 完全实现中得到实现。直接退出时，OSPF 完全实现会首先清除自己生成的所有 LSA，然后终止其所有的相邻关系（neighbor relationship）。上述处理将减少其他 OSPF 路由器的链路状态数据库的大小，并缩减了其他路由器重新路由的时间，详见 13.10 节。

每条 AS-external-LSA 都被指定一个惟一的链路状态 ID 号（Link State ID），甚至是当多个长度不同但地址相同的前缀（prefix）被同时导入时［这种情形发生在 CIDR（Classless Interdomain Routing，无类别域间路由选择）寻址条件下］，每一条 AS-external-LSA 仍然被指定一个惟一的链路状态 ID 号。但是后者分配链路状态 ID 号的算法不同于 RFC 2328 附录 E 中（见 10.3.2 小节）指定的算法。

OSPF 完全实现也可以用于主机（host computer），这时主机希望用 OSPF 来建立自己的路由表，而不希望像一个 IP 路由器那样转发分组（forward packet）。这种情况一般称作主机路由侦听（host wiretapping）。而当使用一般的 **routed** 路由选择守护进程（routing daemon）时，该进程执行的是 RIP（Routing Information Protocol，路由选择信息协议），这时路由侦听通过只接收 RIP 分组而不发送 RIP 分组的方式来完成。而在 OSPF 中路由侦听时，主机通过只收集 OSPF 链路状态数据库（Link State Database）但是不提供自己的链路状态数据库的方式来实现路由侦听，详见 13.12 节。

1.1.1 优化

OSPF 完全实现包含了大量的优化处理，以降低 OSPF 的控制通信量和 CPU 消耗。同样毫不例外，这些优化处理也要付出一定的代价，即 OSPF 完全实现必须提高内存使用率。本实现已努力在内存交换速度与内存容量之间作了权衡，从而不至于过于浪费内存资源。

优化实例包括以下内容：

- OSPF 相邻路由器会话快速达到双向状态（bidirectional state），而不会等待接收定期的 Hello 分组（见 8.2 节）。
- 为同时进行数据库交换的相邻路由器数目设置一个上限，从而保留一定的内存资源，以处理 CPU 请求（见 8.3 节）。
- LSA 的泛洪（flood）会建立发送到多个接口的单个链路状态更新分组（Link State Update Packet），从而避免对除了第一个接口之外的所有其他接口进行更新分组的建立和校验（见 9.3 节）。
- LSA 的重发使用类似于 TCP（Transmission Control Protocol，传输控制协议）慢启动的进程，从而使 LSA 重发速率与相邻路由器接收新 LSA 的速率相匹配（match）（见 9.5 节）。

- AS-external-LSA 的创建受一定的速率限制（见 10.3.2 小节）。
- OSPF 通过预解析（preparsing）LSA 来优化 OSPF 路由选择计算，这样可以避免在路由选择计算期间进行数据库查找、双向链路检验等等（见 11.4 节）。

许多优化可通过设置各个配置参数来调节，见第 13 章。

1.1.2 OSPF 扩展实现

本 OSPF 完全实现除了已经实现了 OSPFv2 基本规范之外，还实现了下面的可选 OSPF 扩展：

- MOSPF。本 OSPF 完全实现包括完整的 MOSPF（Multicast Extensions to OSPF，OSPF 组播扩展）协议[66]实现。MOSPF 实现包括组播路由表表项（multicast routing entry）计算、与 IGMPv2（Internet Group Membership Protocol Version 2，Internet 组成员身份协议版本 2）的交互作用、group-membership-LSA 的生成和区间组播。同时该扩展实现还包括与其他组播路由选择协议（如 DVMRP，Distance Vector Multicast Routing Protocol，距离向量组播路由选择协议）之间的相互作用（见 12.6 节），以及 IGMPv2 实现的路由器部分，即主机身份查询（Host Membership Query）的发送和主机身份报告（Host Membership Report）的接收（见 12.2 节）。
- 需求线路扩展（demand-circuit extension）。该项 OSPF 实现完成对 OSPF 需求线路扩展[64]的实现，它是一种通过拨号和低速链路来运行 OSPF 的有效途径。另外，在配置控制下，AS-external-LSA 创建时可以置位 DoNotAge 位，从而排除刷新这种 LSA 类型的必要性（见 13.1 节）。
- OSPF 数据库溢出（Database Overflow）。OSPF 数据库溢出方法是一种限制 OSPF 数据库大小的方法（在 RFC1765[68]详细说明）。同时该实现还限制非默认 AS-external-LSA 配置的数目和退出数据库溢出状态的时间（见 10.3.4 小节）。

1.1.3 练习与项目

为了使读者更多体验 OSPF 完全实现的方法，本书每一章的末尾都给出了一系列练习。这些练习，读者只需要作很少的代码修改就可实现，并且这些修改都可以通过本书所附带的 OSPF 路由选择仿真器来验证。路由选择仿真器在 Linux 或 Windows 环境下运行。对 Linux 来说，重新编译仿真器的所有必须的开发工具都包含在标准的 Linux 发行版中。对 Windows 来说，作者已经使用的是 Cygwin 工具，它是 GNU 开发工具和实用函数向 Windows 环境的一个移植。这些工具的拷贝已经包含在本书附带的光盘上，对这些工具更新可访问`www.cygnus.com/cygwin`。

更多的有关开发项目的列表在附录 C 中。在附录 C 中提供了比练习更多的代码增强，例如一个 MIB 接口和代替 IPV6 的 OSPF 实现都作为开发项目在附录 C 中予以列出。

1.2 实现机制

OSPF 完全实现几乎完全使用 C++写成，并使用了 GNU 开发工具（C++编译器 **g++** 和 GNU **make**）。这些开发工具与标准 Linux 发行版一起，已经由 Cygwin 项目（**www.cygnus.com/cygwin**）移植入 Windows 下。GNU 工具不仅免费，而且是作者所知道的最好的开发工具。并且作者也经常使用 GNU 调试器 **gdb**。

OSPF 实现中惟一没有使用 C++写成的是以下部分：

- 在 Linux **ospfd** 移植中，配置程序是用 Tcl 写成的（工具命令语言）。在这里作者认为运行时间不是要考虑的主要因素，而是考虑到 Tcl 可以减少很可观的程序开发时间。
- 用户接口（user interface）OSPF 仿真器（simulator）配置文件中的命令解析和图形用户界面都是用 Tk/Tcl 写成的，这可以加速开发速度。
- 作者使用的 MD5 软件是包含 RSA 数据安全（RSA Data Security）等在内的 MD5 消息摘要算法（MD5 Message Digest Algorithm）[91]，整个软件是在 C 中编译的。

作者最初之所以选择 C++为 OSPF 完全实现的其余部分的编程语言，是因为作者发现 C++的类组织结构更容易书写文档，同时 C++的类继承性思想也是一个很好的思想。实际上 C++已经获得的好名声有时并不名副其实，因为在有些方面，它并不适合实时嵌入式（real-time embedded）应用程序，而这点正是路由选择协议软件环境的典型需求。并且还应该记住的是，C++的 **new** 和 **delete** 内存分配（allcation）和释放（free）操作在许多系统上的消耗是十分昂贵的。作者也已经避免使用 C++构造函数，有的最新的编译器，如模板（template），也不执行 C++构造函数。同时，本文的 OSPF 完全实现也避免使用 C++流，因为流支持对编程不十分灵便。

本书并不要求读者是一个专业 C++程序员，因为本书是为那些想学习如何实现路由选择协议的人们而写，而不是为那些想学习 C++编程的人们而写。关于 C++编程，有许多优秀的文档，但不在本书内容之列。特别值得一提，而且作者也确信，纯粹的 C++主义者会恼怒本书中有关 C++使用方面的不规范，如公有和私有数据之间普遍缺乏本质区别、极少使用 C++**const** 结构以及其他许多对 C++的失礼。

1.3 OSPF 路由选择守护进程：ospfd

OSPF 路由选择守护进程（OSPF Routing Daemon）包括一个 OSPF 软件向 Linux 的移植。这个移植生成 **ospfd** 程序，是一个在概念上类似于大多数类 UNIX（unix-like）系统所提供的 **routed** 守护进程的路由选择守护进程。**ospfd** 程序将 Linux 工作站转换成一个 OSPF 路由器。像大多数的路由选择守护进程一样，**ospfd** 将在 Linux 网络初始

化脚本中启动。

ospfd 程序的安装指导放在 2.1 节叙述，读者可在附录 A 中所列的 **ospfd** 用户手册页中查找其使用信息。而关于 **ospfd** 的详细用法信息以及关于 **ospfd** 路由选择守护进程实现的讨论见 14 章。

ospfd 配置数据放在文件 **/etc/ospfd.conf** 中，该文件以人可以阅读的 ASCII 进行编码，并且可以使用任何文体编辑器对其编辑。文件的准确语法（syntax）在手册页和 14.1 节中均有描述。可通过编辑其配置文件对路由选择守护进程动态重新配置，以反映新配置，然后发送给 **ospfd** 进程一个 SIGUSR1 信号：**kill -s USR1** *pid*,其中 *pid* 是 **ospfd** 程序的 linux 进程 ID 号。动态重新配置的副作用的讨论见第 13 章。

在 Linux 系统接口中遭遇的错误，及一些致命错误都会被 Linux **Syslog** 工具记录到日志文件上。OSPF 软件也有自己详细的内部日志工具，该工具在第 5 章以及 14.5 节中讨论。这些日志消息被写入文件 **/var/log/Ospfd.log** 中；消息请求的服务在其写入文件之前是可以配置的。所有日志消息的描述，包括日志的级别（logging level），见附录B。同时 **ospfd** 中还提供两个监控程序：一个是命令行解释器（command line interpreter）**ospfd_mon**；另一个是基于 HTML 的监控器（monitor）**ospfd_browser**。它们能够转储（dump）OSPF 统计信息，如链路状态数据库（link-state database）、相邻路由器状态（neighbor status）等，其相应的手册页在附录 A 中。

ospfd 路由选择守护进程可以配置成运行在主机模式（host mode）下。当配置成主机模式时，路由表表项（routing table entry）将会由 OSPF 协议软件计算，但这时 **ospfd** 不创建 LSA，并且其他 OSPF 路由器也不会试图通过 Linux 工作站转发（forward）通信。这时工作模式等价于 **routed** 的只听模式（listen-only），详细内容见 13.12 节。

因为受 Linux 使用标准 Berkeley 套接接口（socket interface）的限制，OSPF 的某些标准功能，如未编号的点到点接口（unnumbered point-to-point interface），没有包含在 **ospfd** 中。限制的完全列表见 1.5 节。

MOSPF 启动时，**ospfd** 同时也将 Linux 工作站转换成一个组播（multicast）路由器，它运行 MOSPF，并采用组播路由选择协议。运行 MOSPF 的 **ospfd** 将提供类似于 **mrouted** 路由选择守护进程的服务，并用 MOSPF 代替 **mrouted** 的 DVMRP。实际上，在这种模式下，**ospfd** 是重用最初为 **mrouted** 开发的内核接口。

1.4 OSPF 路由选择仿真器

OSPF 代码已经被移植到 OSPF 仿真器（**ospf_sim**）内，OSPF 仿真器运行在 Liunx 或 Windows 下。该仿真器服务于多种功能。在大规模网络上如果不拥有对大量实际路由器的访问能力，则可使用 OSPF 仿真器在其上测试 OSPF 代码及对代码所作的任何修改。同时仿真器 **ospf_sim** 还可以用来测试网络配置的有效性和鲁棒性（robustness），尤其

是OSPF区（area）设计的有效性和鲁棒性。最后值得一提的是，仿真器还可以用作教学辅助工具，演示OSPF协议如何在非平凡网络中（nontrival network）操作。

ospf_sim程序安装指导见2.2节，其用法信息在附录A中列出的**ospf_sim**用户手册页上，详细的用法信息以及有关OSPF仿真器实现的讨论见第15章。

ospf_sim仿真器有一个Tk/Tcl实现的GUI（Graphical User Interface，图形用户界面），它用于显示正在仿真中的OSPF网络图。网络图中的颜色表示随着仿真时间的推移，仿真路由器之间数据库同步（synchronization）的状态，仿真时间精确到0.1s。从GUI上，读者可以修改网络图，添加或删除路由器（router）、网段（network segment）、接口、虚链路（virtual link）、区聚合（area aggregate）等等。在两次仿真运行之间，配置保存在由Tk/Tcl命令组成的ASCII码文件中。这个文件可在文本编辑器（text editor）内修改，以生成大规模网络的配置，并且比通过GUI修改要快得多。

来自仿真路由器的日志消息（logging message）被写到标准输出上，同时写到标准输出上的还有仿真时间和预先规定记录日志消息的路由器的OSPF路由器ID号（OSPF Router ID）。同样地，用于监控Linux **ospfd**路由选择守护进程的**ospfd_mon**程序也可以用来从仿真路由器中收集OSPF统计信息。

1.5 申明

作为版本0.1，OSPF软件以及它的两个移植样例都还不可避免的含有许多bug，其中有许多已经被发现。特别是New Hampshire大学Interoperability实验室[105]的专业测试员们为此付出了很多。但是软件中肯定还存在相当多的bug。正如GNU GPL（附录D）所声明的那样，本软件免费，且不承担任何责任。该声明适用本书附带的OSPF软件和两个样例。特别是对那些希望使用本书提供的**ospfd**路由选择守护进程的人们来说，可以在使用自己确切的网络配置之前，先在OSPF仿真器上测试其软件。

bug修正及对OSPF软件的增强（enhancement）将张贴在**http://www.ospf.org/software/ospfd**上。关于bug的报告请提交到**ospfd-bugs@ospf.org**。提交bug报告时，请尽量给出下面的数据：

- 正使用的OSPF软件的版本。
- 运行OSPF软件的平台，如Linux或Windows，以及操作系统的版本。
- 所遇问题的简洁说明。
- **ospfd**路由选择守护进程或仿真器的配置文件。
- 出现问题的OSPF网格框图。如果问题出在仿真器上，则不需发送网络框图，因为此时网络可完全由配置文件完整描述。
- OSPF软件运行失败之前及失败期间生成的日志消息，以及OSPF软件写入**syslog**中的所有信息。