- 配置。该报文允许控制器查询并设置交换机的配置参数。
- 修改状态。该报文由控制器所使用，以增加/删除或修改交换机流表中的表项，并且设置交换机端口特性。
- 读状态。该报文被控制器用于从交换机的流表和端口收集统计数据和计数器值。
- 发送分组。该报文被控制器用于在受控交换机从特定的端口发送出一个特定的报文。

从受控交换机到控制器流动的重要报文有下列这些：

- 流删除。该报文通知控制器已删除一个流表项，例如由于超时，或作为收到“修改状态”报文的结果。
- 端口状态。交换机用该报文向控制器通知端口状态的变化。
- 分组入。4.4 节讲过，一个分组到达交换机端口，并且不能与任何流表项匹配，那么这个分组将被发送给控制器进行额外处理。匹配的分组也被发送给控制器，作为匹配时所采取的一个动作。“分组入”报文被用于将分组发送给控制器。另外的 OpenFlow 报文定义在［OpenFlow 2009；ONF 2016］中。

实 践 原 则

谷歌的软件定义全球网络

2.6 节中的学习案例中讲过，谷歌部署了专用的广域网（WAN），该网互联了它（在 IXP 和 ISP 中）的数据中心和服务器集群。这个称为 B4 的网络有一个谷歌基于 OpenFlow 设计的 SDN 控制平面。谷歌网络能够在长途线路上以接近 70% 的利用率运行 WAN 链路（超过典型的链路利用率的 2～3 倍），并且基于应用优先权和现有的流需求在多条路径之间分割应用流［Jain 2013］。特别是，谷歌 B4 网络适合于 SDN：①从IXP 和 ISP 中的边缘服务器到网络核心中的路由器，谷歌控制了所有的设备；②带宽最密集的应用是场点之间的大规模数据拷贝，这种数据拷贝在资源拥塞期间能够“服从”较高优先权的交互应用；③由于仅连接了几十个数据中心，集中式控制是可行的。

谷歌的 B4 网络使用定制的交换机，每台交换机实现了 OpenFlow 稍加扩充的版本，具有一个本地 OpenFlow 代理（OFA），该 OFA 方法上类似于我们在图 5-2 中遇到的控制代理。每个 OFA 又与网络控制服务器（NCS）中的 OpenFlow 控制器（OFC）相连，使用一个分离的“带外”网络，与在数据中心之间承载数据中心流量的网络截然不同。该 OFC 因此提供由 NCS 使用的服务以与它的受控交换机通信，方法上类似于显示在图 5-15 中的 SDN 体系结构的最低层。在 B4 中，OFC 也执行状态管理功能，在网络信息库（NIB）中保持节点和链路状态。OFC 的谷歌实现基于 ONIX SDN 控制器［Koponen 2010］。实现了两种路由选择协议：用于数据中心之间路由选择的 BGP 和用于数据中心内部路由选择的 IS-IS（非常接近 OSPF）。Paxos［Chandra 2007］用于执行 NCS 组件的热复制，以防止故障。

一种逻辑上置于网络控制服务器集合之上的流量工程网络控制应用，与这些服务器交互，以为一组应用流提供全局、网络范围的带宽。借助于 B4，SDN 一举跨入全球网络提供商的运行网络的行列。B4 的详细描述请参见［Jain 2013］。

5.5.3　数据平面和控制平面交互的例子

为了具体地理解 SDN 控制的交换机与 SDN 控制器之间的交互，我们考虑图 5-16 中所示的例子，其中使用了 Dijkstra 算法（该算法我们已在 5.2 节中学习过）来决定最短路径路由。图 5-16 中的 SDN 场景与前面 5.2.1 节和 5.3 节中描述的每路由器控制场景有两个重要差异，Dijkstra 算法是实现在每台路由器中并且在所有网络路由器中洪泛链路状态更新：

- Dijkstra 算法作为一个单独的程序来执行，位于分组交换机的外部。
- 分组交换机向 SDN 控制器发送链路更新并且不互相发送。

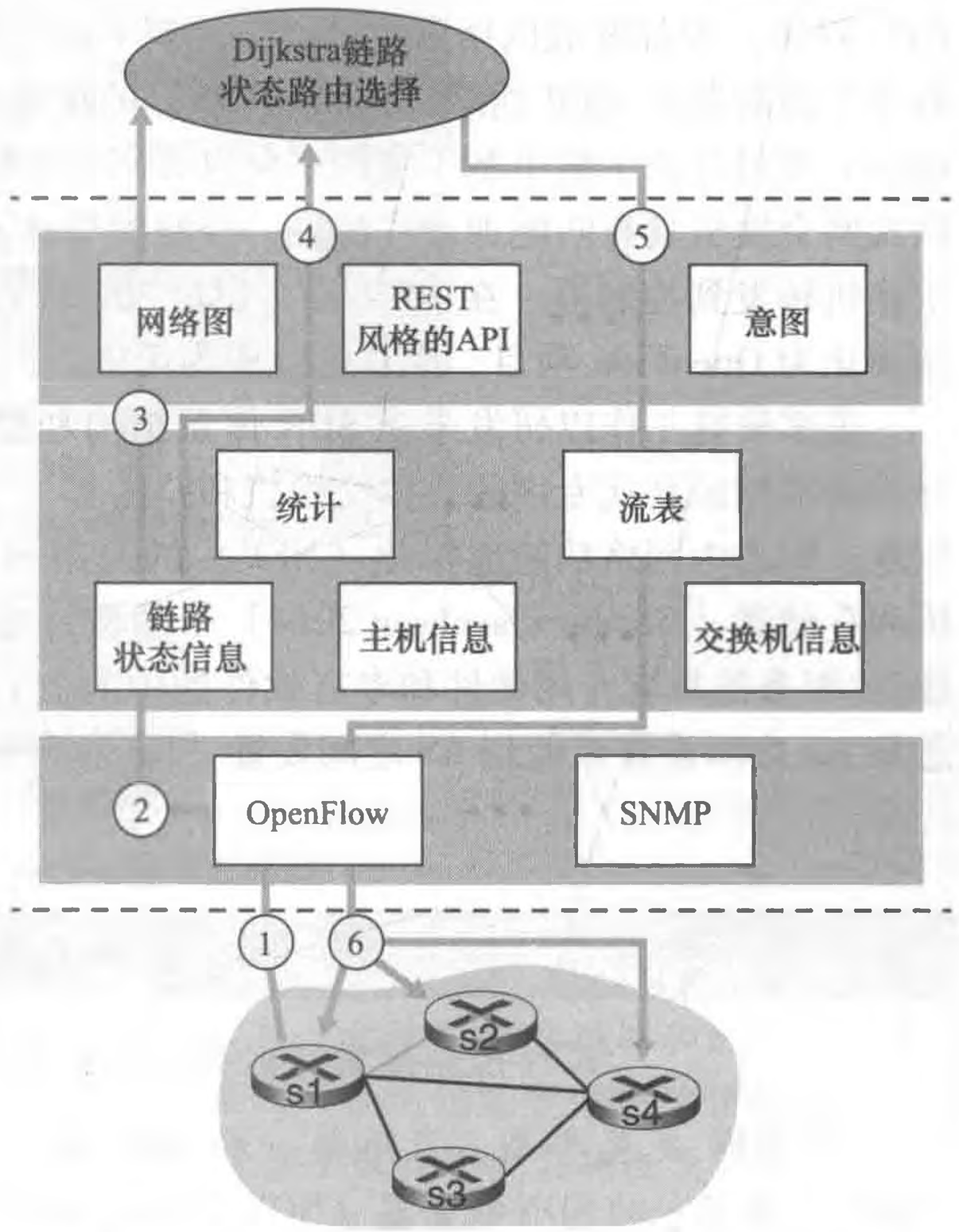


图 5-16　SDN 控制器场景：链路状态更新

在这个例子中，我们假设交换机 s1 和 s2 之间的链路断开；实现了最短路径路由选择，因此，除了 s2 操作未改变外，s1、s3 和 s4 的入和出流转发规则都受到影响。我们也假定 OpenFlow 被用作通信层协议，控制平面只执行链路状态路由选择而不执行其他功能。

1）交换机 s1 经历了自己与 s2 之间的链路故障，使用 OpenFlow “端口状态”报文向 SDN 控制器通报该链路状态的更新。

2）SDN 控制器接收指示链路状态更新的 OpenFlow 报文，并且通告链路状态管理器，由管理器更新链路状态库。

3）实现 Dijkstra 链路状态路由选择的网络控制应用程序先前进行了注册，当链路状态更新时将得到通告。应用程序接收该链路状态更新的通告。

4）链路状态路由选择应用程序与链路状态管理器相互作用，以得到更新的链路状态；它也会参考状态管理层中的其他组件。然后它计算新的最低开销路径。

5）链路状态路由选择应用则与流表管理器交互，流表管理器决定更新的流表。

6）流表管理器则使用 OpenFlow 协议更新位于受影响的交换机 s1、s2 和 s4 的流表项，其中 s1 此时将经 s4 将分组的目的地指向 s2，s2 此时将经中间交换机 s4 开始接收来自 s1 的分组，s4 此时必须转发来自 s1 且目的地为 s2 的分组。

这个例子虽简单，但图示了 SDN 控制平面如何提供控制平面服务（此时为网络层路由选择），而该服务以前是以每路由器控制在每台路由器中实现的。我们现在能够容易地体会到，SDN 使能的 ISP 能够容易地将最低开销路径的路由选择转变为更加定制的路由选择方法。因为控制器的确能够随心所欲地定制流表，因此能够实现它喜欢的任何形式的转发，即只是通过改变它的应用控制软件。这种改变的便利性与传统的每路由器控制平面的

情况形成对照，传统的情况必须要改变所有路由器中的软件，而这些路由器可能是由多个不同厂商提供给 ISP 的。

5.5.4　SDN 的过去与未来

尽管对 SDN 的强烈兴趣是相对近期的现象，但 SDN 的技术根源，特别是数据平面和控制平面的分离，可追溯到相当久远。在 2004 年，文献［Feamster 2004；Lakshman 2004；RFC 3746］中都赞成网络数据平面与控制平面分离。［van der Merwe 1998］描述了用于具有多个控制器的 ATM 网络［Black 1995］的控制框架，每台控制器控制若干 ATM 交换机。Ethane 项目开拓了简单基于流的多台以太网交换机和一台集中式控制器的网络概念，其中以太网交换机具有匹配加动作流表，控制器管理流准入和路由选择，而未匹配的分组将从交换机转发到控制器。在 2007 年，超过 300 台 Ethane 交换机的网络投入运行。Ethane 迅速演化为 OpenFlow 项目，而其他的成为历史！

很多研究工作以研发未来 SDN 体系结构和能力为目标。如我们所见，SDN 革命正在导致颠覆性地替代专用的整体交换机和路由器（它们同时具有数据平面和控制平面）。类似地，称之为网络功能虚拟化（NFV）的通用 SDN 的目标是用简单的商用服务器、交换机和存储器［Gember-Jacobson 2014］来颠覆性地替代复杂的中间盒（例如用于媒体高速缓存/服务的具有专用硬件和专有软件的中间盒）。第二个重要研究领域是寻求将 SDN 概念从 AS 内部设置扩展到 AS 之间设置［Gupta 2014］。

实 践 原 则

SDN 控制器学习案例：OpenDaylight 和 ONOS 控制器

在 SDN 发展早期，采用单一的 SDN 协议（OpenFlow［McKeown 2008；OpenFlow 2009］）和单一的 SDN 控制器（NOX［Gude 2008］）。此后，尤其是 SDN 控制器的数量有了很大增长［Kreutz 2015］。某些 SDN 控制器是公司特有的和专用的，例如 ONIX［Koponen 2010］、瞻博网络的 Contrail［Juniper Contrail 2016］以及谷歌用于其 B4 广域网的控制器［Jain 2013］。而更多控制器是开源的并以各种各样的编程语言实现［Erickson 2013］。最近，OpenDaylight 控制器［OpenDaylight Lithium 2016］和 ONOS 控制器［ONOS 2016］在产业界得到广泛支持。它们都是开源的，并且是与 Linux 基金会合作开发的。

OpenDaylight 控制器

图 5-17 呈现了 OpenDaylight Lithium SDN 控制器平台的简化视图［OpenDaylight Lithium 2016］。ODL 控制器组件的主要部分与我们在图 5-15 中给出的那些组件严格对应。**网络服务应用程序**用于决定数据平面转发和其他服务（如防火墙和负载均衡）如何在受控交换机中完成。与图 5-15 中规范的控制器不同，ODL 控制器具有**两个**接口，通过这两个接口，应用程序可以与原生的控制器服务通信以及彼此之间相互通信：外部应用程序使用 REST 请求 - 响应 API 与控制器模块通信，通信运行在 HTTP 上。内部应用程序经过服务抽象层（SAL）互相通信。至于控制器应用程序是在外部还是内部实现，这是由应用程序设计者选择决定的，图 5-17 中显示的特定应用程序配置只是作为一个例子。

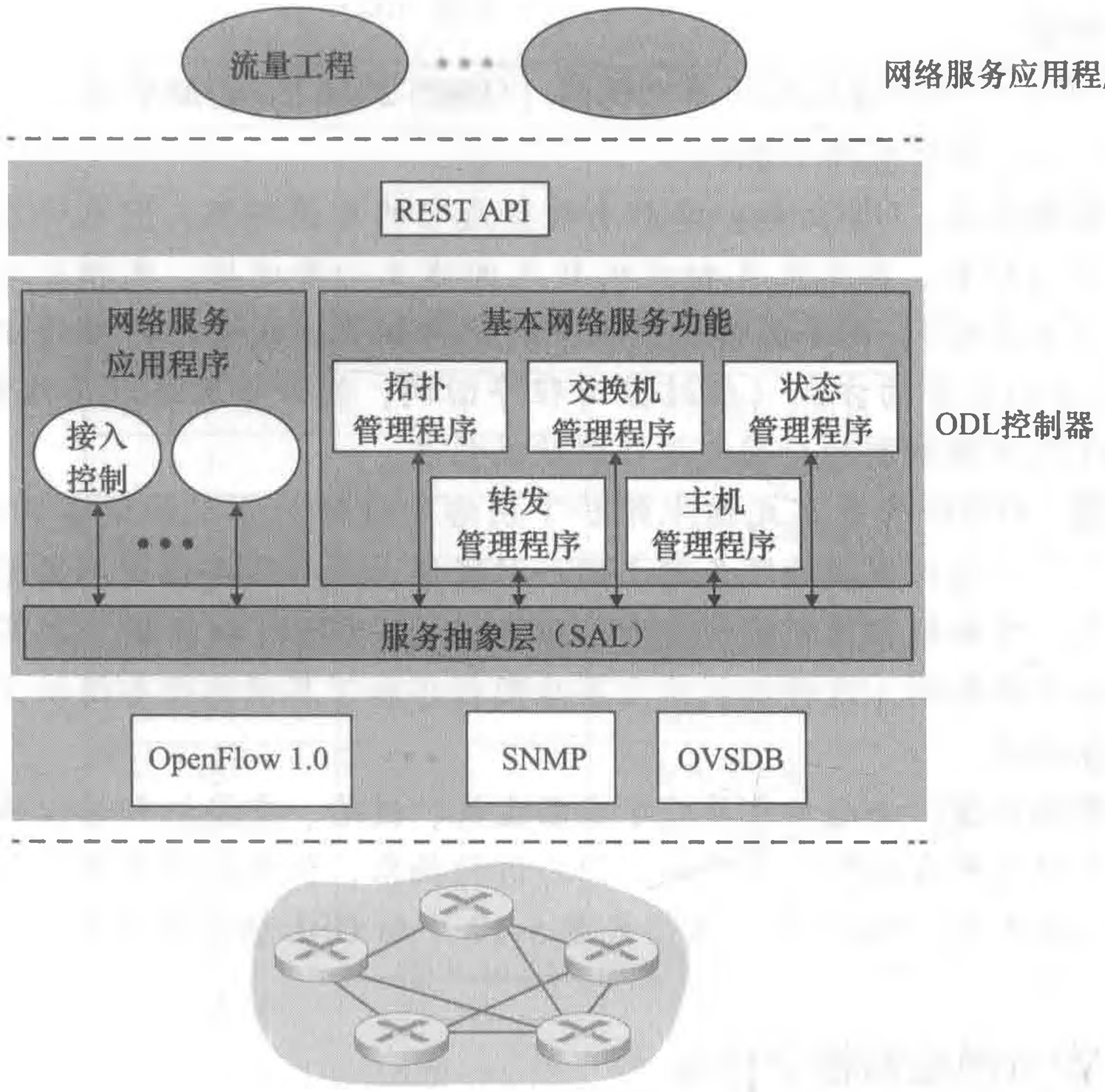


图 5-17　OpenDaylight 控制器

ODL 的**基本网络服务功能**是该控制器的核心，它们密切对应于我们在图 5-15 中遇到的网络范围状态管理能力。SAL 是控制器的神经中枢，允许控制器组件和应用程序互相调用服务并且订阅它们产生的事件。它也在通信层次对特定的**底层通信协议**提供了统一的抽象接口，包括 OpenFlow 和 SNMP（简单网络管理协议，即我们将在 5.7 节中涉及的一种网络管理协议）。OVSDB 是用于管理数据中心交换的协议，而数据中心交换是 SDN 技术的一个重要应用领域。我们将在第 6 章中介绍数据中心网络。

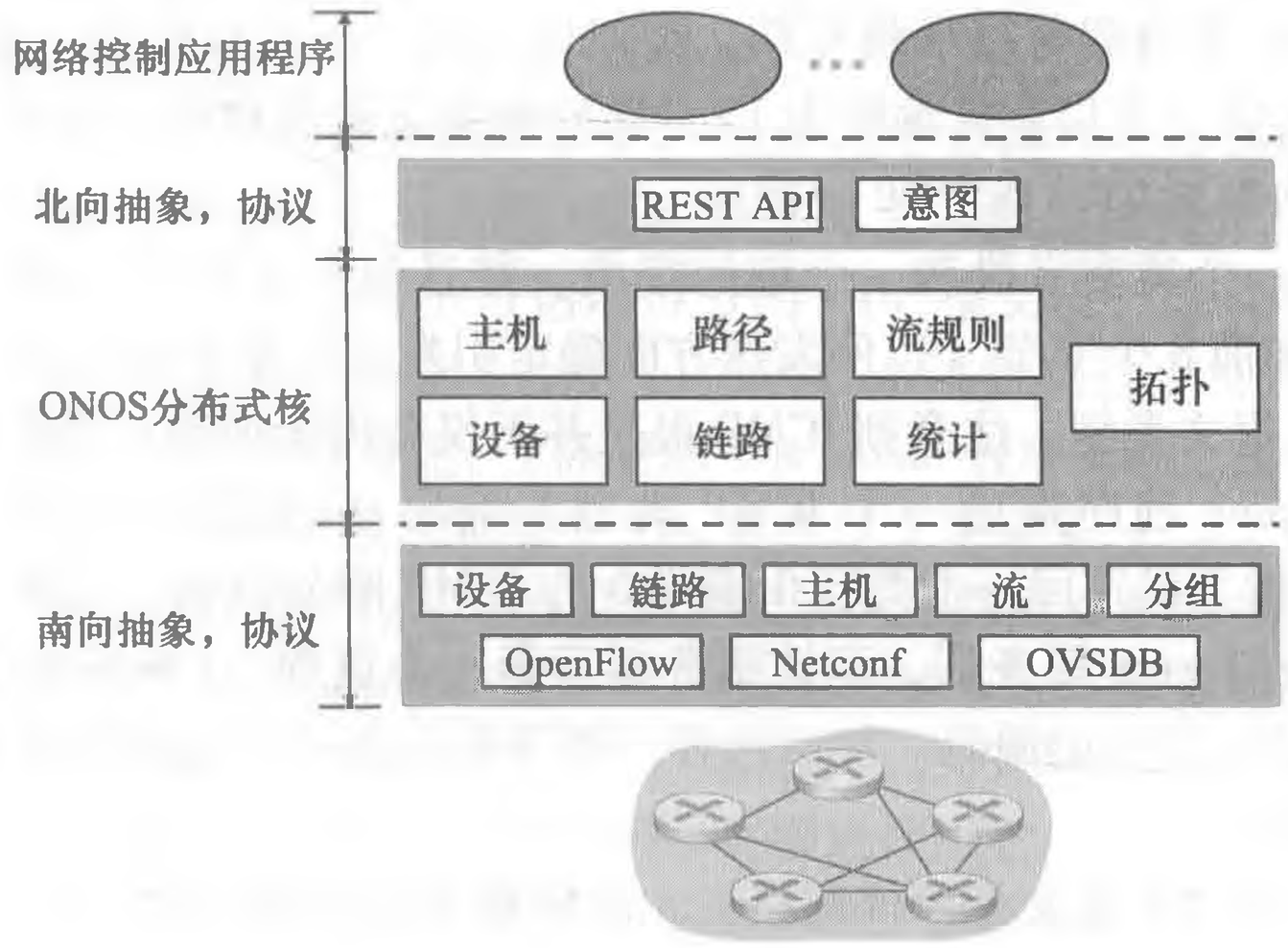


图 5-18　ONOS 控制器体系结构

ONOS 控制器

图 5-18 呈现了 ONOS 控制器的简化视图［ONOS 2016］。类似于图 5-15 中的规范控制器，在 ONOS 控制器中有三个层次：

- **北向抽象和协议**。ONOS 的一个独有特征是它的意图框架，它允许应用程序请求高层服务（例如，在主机 A 和主机 B 之间建立一条连接，或相反地不允许主机 A 和主机 B 通信），而不必知道该服务的执行细节。或者以同步的方式（经过请求），或者以异步的方式（经过监听程序回调，例如等网络状态改变时），经过北向 API 向网络控制的应用程序提供状态信息。
- **分布式核**。ONOS 的分布式核中维护了网络的链路、主机和设备的状态。ONOS 被部署为在一系列互联的服务器上的一种服务，每台服务器运行着 ONOS 软件的相同副本，增加服务器数量就提升了服务能力。ONOS 核提供了在实例之间服务复制和协同的机制，这种机制为上层应用程序和下层网络设备提供了逻辑上集中的核服务抽象。
- **南向抽象和协议**。南向抽象屏蔽了底层主机、链路、交换机和协议的异构性，允许分布式核对设备和协议不可知。因为这种抽象，位于分布式核下方的南向接口逻辑上比图 5-14 中的规范控制器或图 5-17 中的 ODL 控制器更高。

5.6　ICMP：因特网控制报文协议

由［RFC 792］定义的因特网控制报文协议（ICMP），被主机和路由器用来彼此沟通网络层的信息。ICMP 最典型的用途是差错报告。例如，当运行一个 HTTP 会话时，你也许会遇到一些诸如“目的网络不可达”之类的错误报文。这种报文就来源于 ICMP。在某个位置，IP 路由器不能找到一条通往 HTTP 请求中所指定的主机的路径，该路由器就会向你的主机生成并发出一个 ICMP 报文以指示该错误。

ICMP 通常被认为是 IP 的一部分，但从体系结构上讲它位于 IP 之上，因为 ICMP 报文是承载在 IP 分组中的。这就是说，ICMP 报文是作为 IP 有效载荷承载的，就像 TCP 与 UDP 报文段作为 IP 有效载荷被承载那样。类似地，当一台主机收到一个指明上层协议为 ICMP 的 IP 数据报时（上层协议编码为 1），它分解出该数据报的内容给 ICMP，就像分解出一个数据报的内容给 TCP 或 UDP 一样。

ICMP 报文有一个类型字段和一个编码字段，并且包含引起该 ICMP 报文首次生成的 IP 数据报的首部和前 8 个字节（以便发送方能确定引发该差错的数据报）。在图 5-19 中显示了所选的 ICMP 报文类型。注意到 ICMP 报文并不仅是用于通知差错情况。

众所周知的 ping 程序发送一个 ICMP 类型 8 编码 0 的报文到指定主机。看到回显（echo）请求，目的主机发回一个类型 0 编码 0 的 ICMP 回显回答。大多数 TCP/IP 实现直接在操作系统中支持 ping 服务器，即该服务器不是一个进程。［Stevens 1990］的第 11 章提供了有关 ping 客户程序的源码。注意到客户程序需要能够指示操作系统产生一个类型 8 编码 0 的 ICMP 报文。

另一个有趣的 ICMP 报文是源抑制报文。这种报文在实践中很少使用。其最初目的是执行拥塞控制，即使得拥塞的路由器向一台主机发送一个 ICMP 源抑制报文，以强制该主机减小其发送速率。我们在第 3 章已看到，TCP 在运输层有自己的拥塞控制机制，不需要

利用网络层中的反馈信息，如 ICMP 源抑制报文。

ICMP 类型	编码	描述
0	0	回显回答（对 ping 的回答）
3	0	目的网络不可达
3	1	目的主机不可达
3	2	目的协议不可达
3	3	目的端口不可达
3	6	目的网络未知
3	7	目的主机未知
4	0	源抑制（拥塞控制）
8	0	回显请求
9	0	路由器通告
10	0	路由器发现
11	0	TTL 过期
12	0	IP 首部损坏

图 5-19　ICMP 报文类型

在第 1 章中我们介绍了 Traceroute 程序，该程序允许我们跟踪从一台主机到世界上任意一台主机之间的路由。有趣的是，Traceroute 是用 ICMP 报文来实现的。为了判断源和目的地之间所有路由器的名字和地址，源主机中的 Traceroute 向目的地主机发送一系列普通的 IP 数据报。这些数据报的每个携带了一个具有不可达 UDP 端口号的 UDP 报文段。第一个数据报的 TTL 为 1，第二个的 TTL 为 2，第三个的 TTL 为 3，依次类推。该源主机也为每个数据报启动定时器。当第 n 个数据报到达第 n 台路由器时，第 n 台路由器观察到这个数据报的 TTL 正好过期。根据 IP 协议规则，路由器丢弃该数据报并发送一个 ICMP 告警报文给源主机（类型 11 编码 0）。该告警报文包含了路由器的名字和它的 IP 地址。当该 ICMP 报文返回源主机时，源主机从定时器得到往返时延，从 ICMP 报文中得到第 n 台路由器的名字与 IP 地址。

Traceroute 源主机是怎样知道何时停止发送 UDP 报文段的呢？前面讲过源主机为它发送的每个报文段的 TTL 字段加 1。因此，这些数据报之一将最终沿着这条路到达目的主机。因为该数据报包含了一个具有不可达端口号的 UDP 报文段，该目的主机将向源发送一个端口不可达的 ICMP 报文（类型 3 编码 3）。当源主机收到这个特别的 ICMP 报文时，知道它不需要再发送另外的探测分组。（标准的 Traceroute 程序实际上用相同的 TTL 发送 3 个一组的分组，因此 Traceroute 输出对每个 TTL 提供了 3 个结果。）

以这种方式，源主机知道了位于它与目的主机之间的路由器数量和标识，以及两台主机之间的往返时延。注意到 Traceroute 客户程序必须能够指令操作系统产生具有特定 TTL 值的 UDP 数据报，当 ICMP 报文到达时，也必须能够由它的操作系统进行通知。既然你已明白了 Traceroute 的工作原理，你也许想返回去更多地使用它。

在 RFC 4443 中为 IPv6 定义了 ICMP 的新版本。除了重新组织现有的 ICMP 类型和编码定义外，ICMPv6 还增加了新型 IPv6 功能所需的新类型和编码。这些包括“分组太大”类型和一个“未被承认的 IPv6 选项”差错编码。

5.7　网络管理和 SNMP

此时我们的网络层学习已经走到了结尾，我们前面仅有链路层了，我们都熟知网络是由许多复杂、交互的硬件和软件部件组成的，既包括构成网络的物理部件的链路、交换机、路由器、主机和其他设备，也包括控制和协调这些设备的许多协议。当一个机构将数以百计或数以千计的这种部件拼装在一起形成一个网络时，保持该网络“运行良好”对网络管理员无疑是一种挑战。我们在 5.5 节中看到，SDN 环境中逻辑上集中的控制器能够有助于这种过程。但是网络管理的挑战在 SDN 出现前很久就已如影相随了，网络管理员使用丰富的网络管理工具和方法来监视、管理和控制该网络。在本节中我们将学习这些工具和技术。

一个经常被问到的问题是：什么是网络管理？我们用一个构思缜密的单句（虽然它相当冗长）来概括网络管理的定义［Saydam 1996］：

> “网络管理包括了硬件、软件和人类元素的设置、综合和协调，以监视、测试、轮询、配置、分析、评价和控制网络及网元资源，用合理的成本满足实时性、运营性能和服务质量的要求。”

给定了这个宽泛的定义，本节我们将仅涉及网络管理的入门知识，即网络管理员在执行其任务中所使用的体系结构、协议和信息库。我们将不涉及网络管理员的决策过程，其中故障标识［Labovitz 1997；Steinder 2002；Feamster 2005；Wu 2005；Teixeira 2006］、异常检测［Lakhina 2005；Barford 2009］、满足约定的服务等级约定（Service Level Agreements，SLA）的网络设计/工程［Huston 1999a］等主题会加以考虑。因此我们有意识地收窄关注点，有兴趣的读者应当参考这些文献：由 Subramanian 撰写的优秀的网络管理教科书［Subramanian 2000］，以及本书 Web 网站上可用的详尽的网络管理材料。

5.7.1　网络管理框架

图 5-20 显式了网络管理的关键组件。

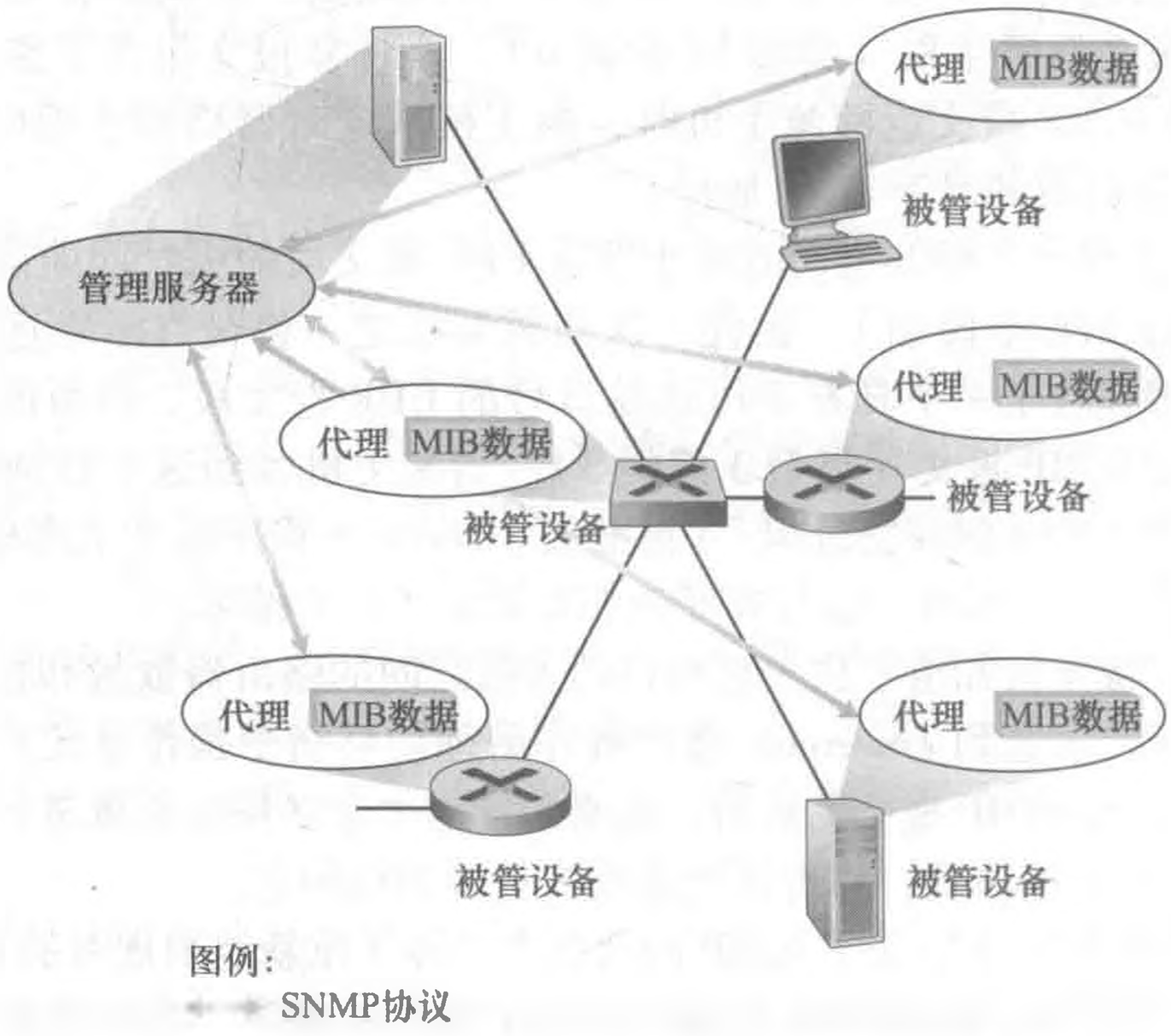


图 5-20　网络管理的组件：管理服务器，被管设备，MIB 数据，远程代理，SNMP

- **管理服务器**（managing server）是一个应用程序，通常有人的参与，并运行在网络运营中心（NOC）的集中式网络管理工作站上。管理服务器是执行网络管理活动的地方，它控制网络管理信息的收集、处理、分析和/或显示。正是在这里，发起控制网络行为的动作，人类网络管理员与网络设备打交道。
- **被管设备**（managed device）是网络装备的一部分（包括它的软件），位于被管理的网络中。被管设备可以是一台主机、路由器、交换机、中间盒、调制解调器、温度计或其他联网的设备。在一个被管设备中，有几个所谓**被管对象**（managed object）。这些被管对象是被管设备中硬件的实际部分（例如，一块网络接口卡只是一台主机或路由器的一个组件）和用于这些硬件及软件组件的配置参数（例如，像OSPF 这样的 AS 内部路由选择协议）。
- 一个被管设备中的每个被管对象的关联信息收集在**管理信息库**（Management Information Base，MIB）中，我们将看到这些信息的值可供管理服务器所用（并且在许多场合下能够被设置）。一个 MIB 对象可以是：一个计数器，例如由于 IP 数据报首部差错而由路由器丢弃的 IP 数据报的数量，或一台主机接收到的 UDP 报文段的数量；运行在一台 DNS 服务器上的软件版本的描述性信息；诸如一个特定设备功能是否正确的状态信息；或诸如到一个目的地的路由选择路径的特定协议的信息。MIB 对象由称为 SMI（Structure of Management Information）［RFC 2578；RFC 2579；RFC 2580］的数据描述语言所定义。使用形式化定义语言可以确保网络管理数据的语法和语义是定义良好的和无二义性的。相关的 MIB 对象被收集在 MIB 模块（module）中。到2015 年年中，RFC 定义了大约400 个 MIB 模块，还有大量厂商特定的（专用的）MIB 模块。
- 在每个被管设备中还驻留有**网络管理代理**（network management agent），它是运行在被管设备中的一个进程，该进程与管理服务器通信，在管理服务器的命令和控制下在被管设备中采取本地动作。网络管理代理类似于我们在图 5-2 中看到的路由选择代理。
- 网络管理框架的最后组件是**网络管理协议**（network management protocol）。该协议运行在管理服务器和被管设备之间，允许管理服务器查询被管设备的状态，并经过其代理间接地在这些设备上采取行动。代理能够使用网络管理协议向管理服务器通知异常事件（如组件故障或超过了性能阈值）。重要的是注意到网络管理协议自己不能管理网络。恰恰相反，它为网络管理员提供了一种能力，使他们能够管理（“监视、测试、轮询、配置、分析、评价和控制”）网络。这是一种细微但却重要的区别。在下节中，我们将讨论因特网的 SNMP 协议。

5.7.2　简单网络管理协议

简单网络管理协议（Simple Network Management Protocol）版本 2（SNMPv2）［RFC 3416］是一个应用层协议，用于在管理服务器和代表管理服务器执行的代理之间传递网络管理控制和信息报文。SNMP 最常使用的是请求响应模式，其中 SNMP 管理服务器向 SNMP 代理发送一个请求，代理接收到该请求后，执行某些动作，然后对该请求发送一个回答。请求通常用于查询（检索）或修改（设置）与某被管设备关联的 MIB 对象值。SNMP 第二个常被使用的是代理向管理服务器发送的一种非请求报文，该报文称为**陷阱报文**（trap message）。陷阱报文用于通知管理服务器，一个异常情况（例如一个链路接口启

动或关闭）已经导致了 MIB 对象值的改变。

表 5-2 中显示了 SNMPv2 定义的 7 种类型的报文，这些报文一般称为协议数据单元（PDU）。图 5-21 显示了这些 PDU 的格式。

表 5-2　SNMPv2 PDU 类型

SNMPv2 PDU 类型	发送方 - 接收方	描述
GetRequest	管理者到代理	取得一个或多个 MIB 对象实例值
GetNextRequest	管理者到代理	取得列表或表格中下一个 MIB 对象实例值
GetBulkRequest	管理者到代理	以大数据块方式取得值，例如大表中的值
InformRequest	管理者到管理者	向不能访问的远程管理实体通知 MIB 值
SetRequest	管理者到代理	设置一个或多个 MIB 对象实例的值
Response	代理到管理者或管理者到管理者	对 GetRequest，GetNextRequest，GetBulkRequest，SetRequest PDU，或 InformRequest 产生的响应
SNMPv2 - Trap	代理到管理者	向管理者通知一个异常事件

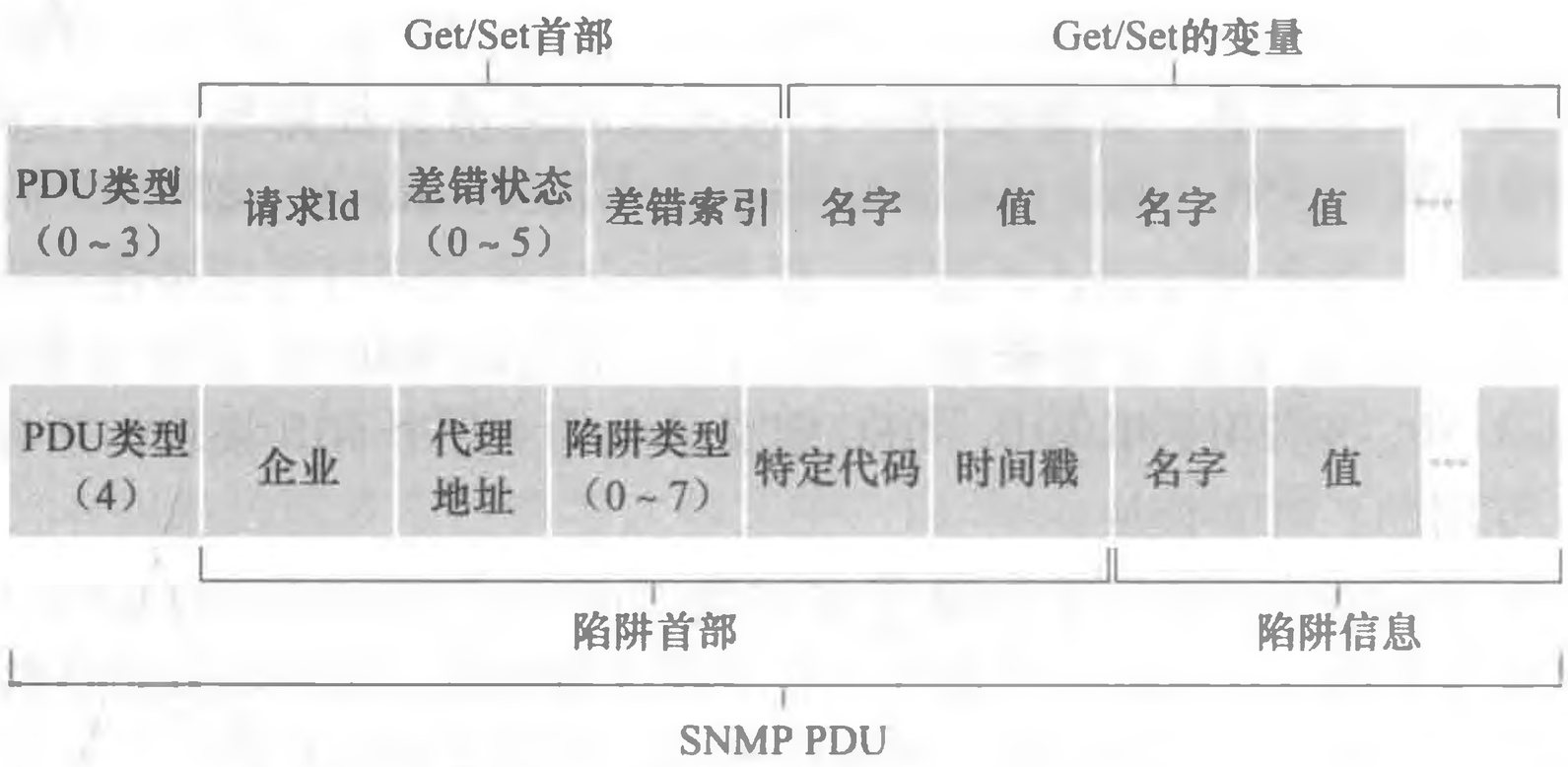


图 5-21　SNMP PDU 格式

- GetRequest、GetNextRequest 和 GetBulkRequest PDU 都是管理服务器向代理发送的，以请求位于该代理所在的被管设备中的一个或多个 MIB 对象值。其值被请求的 MIB 对象的对象标识符定义在该 PDU 的变量绑定部分。GetRequest、GetNextRequest 和 GetBulkRequest 的差异在于它们的数据请求粒度。GetRequest 能够请求 MIB 值的任意集合；多个 GetNextRequest 能用于顺序地读取 MIB 对象的列表或表格；GetBulkRequest 允许读取大块数据，能够避免因发送多个 GetRequest 或 GetNextRequest 报文可能导致的额外开销。在所有这三种情况下，代理用包括该对象标识符和它们相关值的 Response PDU 进行响应。
- 管理服务器使用 SetRequest PDU 来设置位于被管设备中的一个或多个 MIB 对象的值。代理用带有“noError”差错状态的 Response PDU 进行应答，以证实该值的确已被设置。
- 管理服务器使用 InformRequest PDU 来通知另一个 MIB 信息管理服务器，后者对于接收服务器是远程的。
- Response PDU 通常从被管设备发送给管理服务器，以响应来自该服务器的请求报文，返回所请求的信息。

- SNMPv2 PDU 的最后一种类型是陷阱报文。陷阱报文是异步产生的，即它们不是为了响应接收到的请求而产生的，而是为了响应管理服务器要求通知的事件而产生的。RFC 3418 定义了周知的陷阱类型，其中包括设备的冷启动或热启动、链路就绪或故障、找不到相邻设备，或鉴别失效事件。接收到的陷阱请求不要求从管理服务器得到响应。

考虑到 SNMPv2 请求响应性质，这时需要注意到尽管 SNMP PDU 能够通过许多不同的运输协议传输，但 SNMP PDU 通常是作为 UDP 数据报的载荷进行传输的。RFC 3417 的确表明 UDP 是“首选的运输映射”。然而，由于 UDP 是一种不可靠的运输协议，因而不能确保一个请求或它的响应能够被它希望的目的地接收到。管理服务器用该 PDU 的请求 ID 字段为它向代理发送的请求编号；该代理的响应从接收到的请求中获取它的请求 ID。因此，该请求 ID 字段能被管理服务器用来检测丢失的请求或回答。如果在一定时间后还没有收到对应的响应，由管理服务器来决定是否重传一个请求。特别是，SNMP 标准没有强制任何特殊的重传过程，即使初始进行重传。它只是要求管理服务器“需要根据重传的频率和周期做出负责任的动作”。当然，这使人想知道一个“负责任的”协议应当如何动作！

SNMP 经历了 3 个版本的演变。SNMPv3 的设计者说过“SNMPv3 能被认为是具有附加安全性和管理能力的 SNMPv2”［RFC 3410］。SNMPv3 无疑在 SNMPv2 基础上有改变，而没有什么比在管理和安全领域的变化更为明显。在 SNMPv3 中，安全性的中心地位特别重要，因为缺乏适当的安全性导致 SNMP 主要用于监视而不是控制（例如，在 SNMPv1 中很少使用 SetRequest）。我们再一次看到安全性是重要的关注点（安全性是第 8 章详细学习的主题），尽管认识到它的重要性也许有些迟了，但“亡羊补牢，犹未为晚”。

5.8　小结

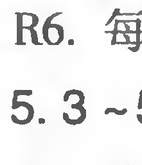
我们现在已经完成了进入网络核心的两章旅程，即开始于第 4 章的网络层数据平面的学习和本章完成的网络层控制平面的学习。我们知道了控制平面是网络范围的逻辑，它不仅控制从源主机到目的主机沿着端到端路径在路由器之间如何转发数据报，而且控制网络层组件和服务器如何配置和管理。

我们学习了构建控制平面有两大类方法：传统的每路由器控制（其中在每台路由器中运行算法，并且路由器中的路由选择组件与其他路由器中的路由选择组件通信）和软件定义网络（SDN）控制（其中一个逻辑上集中的控制器计算并向每台路由器分发转发表为它们所用）。我们在 5.2 节中学习了两种基本的路由选择算法，即链路状态和距离矢量，用于计算图中的最小开销路径；这些算法在每路由器控制和 SDN 控制中都有应用。这些算法是两种广泛部署的因特网路由选择协议 OSPF 和 BGP 的基础，我们在 5.3 节和 5.4 节中讨论了这两种协议。我们在 5.5 节中讨论了网络层控制平面的 SDN 方法，研究了 SDN 网络控制应用程序、SDN 控制器，以及控制器和 SDN 控制设备之间通信所使用的 OpenFlow 协议。在 5.6 节和 5.7 节中，我们包括了管理 IP 网络的某些技术细节：ICMP（互联网控制报文协议）和 SNMP（简单网络管理协议）。

在完成了网络层学习之后，我们的旅行此时沿着协议栈向下走了一步，即到了链路层。像网络层一样，链路层是每台网络连接的设备的一部分。但我们将在下一章中看到，链路层的任务是在相同链路或局域网之间更局域化地移动分组。尽管这种任务从表面上看

可能比网络层任务简单得多，但我们将看到，链路层涉及许多重要和引人入胜的问题，这些问题会花费我们不少时间。

课后习题和问题



复习题

5.1 节

R1. 基于每路由器控制的控制平面意味着什么？在这种情况下，当我们说网络控制平面和数据平面是“整体地”实现时，是什么意思？

R2. 基于逻辑上集中控制的控制平面意味着什么？在这种有情况下，数据平面和控制平面是在相同的设备或在分离的设备中实现的吗？请解释。

5.2 节

R3. 比较和对照集中式和分布式路由选择算法的性质。给出一个路由选择协议的例子，该路由选择协议采用分布式方法和集中式方法。

R4. 比较和对照链路状态和距离矢量这两种路由选择算法。

R5. 在距离矢量路由选择中的“无穷计数”是什么意思？

R6. 每个自治系统使用相同的 AS 内部路由选择算法是必要的吗？说明其原因。

5.3 ~5.4 节

R7. 为什么在因特网中用到了不同的 AS 间与 AS 内部协议？

R8. 是非判断题：当一台 OSPF 路由器发送它的链路状态信息时，它仅向那些直接相邻的节点发送。解释理由。

R9. 在 OSPF 自治系统中区域表示什么？为什么引入区域的概念？

R10. 定义和对比下列术语：子网、前缀和 BGP 路由。

R11. BGP 是怎样使用 NEXT-HOP 属性的？它是怎样使用 AS-PATH 属性的？

R12. 描述一个较高层 ISP 的网络管理员在配置 BGP 时是如何实现策略的。

R13. 是非判断题：当 BGP 路由器从它的邻居接收到一条通告的路径时，它必须对接收路径增加上它自己的标识，然后向其所有邻居发送该新路径。解释理由。

5.5 节

R14. 描述在 SDN 控制器中的通信层、网络范围状态管理层和网络控制应用程序层的主要任务。

R15. 假定你要在 SDN 控制平面中实现一个新型路由选择协议。你将在哪个层次中实现该协议？解释理由。

R16. 什么类型的报文流跨越 SDN 控制器的北向和南向 API？谁是从控制器跨越南向接口发送的这些报文的接收者？谁是跨越北向接口从控制器发送的这些报文的接收者？

R17. 描述两种从受控设备到控制器发送的 OpenFlow 报文（由你所选）类型的目的。描述两种从控制器到受控设备发送的 OpenFlow 报文（由你所选）类型的目的。

R18. 在 OpenDaylight SDN 控制器中服务抽象层的目的是什么？

5.6 ~5.7 节

R19. 列举 4 种不同类型的 ICMP 报文。

R20. 在发送主机执行 Traceroute 程序，收到哪两种类型的 ICMP 报文？

R21. 在 SNMP 环境中定义下列术语：管理服务器、被管设备、网络管理代理和 MIB。

R22. SNMP GetRequest 和 SetRequest 报文的目的是什么？

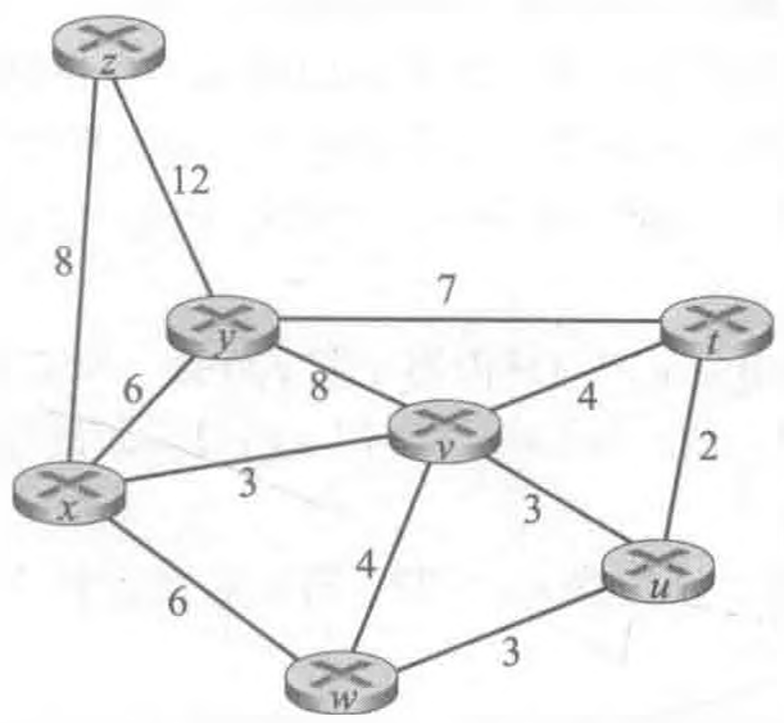
R23. SNMP 陷阱报文的目的是什么？

习题

P1. 观察图 5-3，列举从 y 到 u 不包含任何环路的路径。

P2. 重复习题 P1，列举从 x 到 z、z 到 u 以及 z 到 w 的不包含任何环路的路径。

P3. 考虑下面的网络。对于标明的链路开销，用 Dijkstra 的最短路算法计算出从 x 到所有网络节点的最短路径。通过计算一个类似于表 5-1 的表，说明该算法是如何工作的。



P4. 考虑习题 P3 中所示的网络。使用 Dijkstra 算法和一个类似于表 5-1 的表来说明你做的工作：

a. 计算出从 t 到所有网络节点的最短路径。

b. 计算出从 u 到所有网络节点的最短路径。

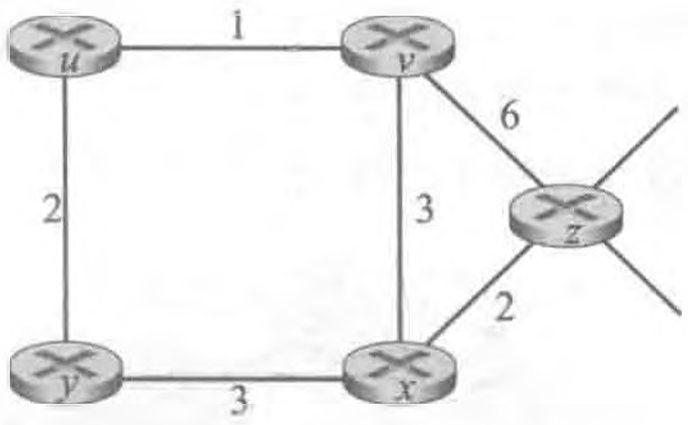
c. 计算出从 v 到所有网络节点的最短路径。

d. 计算出从 w 到所有网络节点的最短路径。

e. 计算出从 y 到所有网络节点的最短路径。

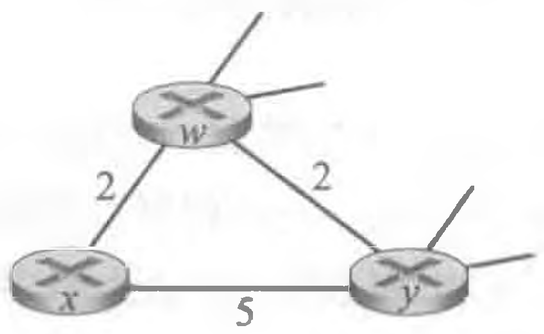
f. 计算出从 z 到所有网络节点的最短路径。

P5. 考虑下图所示的网络，假设每个节点初始时知道到它的每个邻居的开销。考虑距离向量算法，并显示在节点 z 中的距离表表项。



P6. 考虑一个一般性拓扑（即不是以上所显示的特定网络）和一个同步版本的距离向量算法。假设每次迭代时，一个节点与其邻居交换其距离向量并接收它们的距离向量。假定算法开始时，每个节点只知道到其直接邻居的开销，在该分布式算法收敛前所需的最大迭代次数是多少？评估你的答案。

P7. 考虑下图所示的网络段。x 只有两个相连邻居 w 与 y。w 有一条通向目的地 u（没有显示）的最低开销路径，其值为 5，y 有一条通向目的地 u 的最低开销路径，其值为 6。从 w 与 y 到 u（以及 w 与 y 之间）的完整路径未显示出来。网络中所有链路开销皆为正整数值。



a. 给出 x 对目的地 w、y 和 u 的距离向量。

b. 给出对于 $c(x, w)$ 或 $c(x, y)$ 的链路开销的变化，使得执行了距离向量算法后，x 将通知其邻居有一条通向 u 的新最低开销路径。

c. 给出对 $c(x, w)$ 或 $c(x, y)$ 的链路开销的变化，使得执行了距离向量算法后，x 将不通知其邻居有一条通向 x 的新最低开销路径。

P8. 考虑如图 5-6 中所示 3 个节点的拓扑。不使用显示在图 5-6 中的开销值，链路开销值现在是 $c(x, y)=3$，$c(y, z)=6$，$c(z, x)=4$。在距离向量表初始化后和在同步版本的距离向量算法每次迭代后，计算它的距离向量表（如我们以前对图 5-6 讨论时所做的那样）。

P9. 考虑距离向量路由选择中的无穷计数问题。如果我们减小一条链路的开销，将会出现无穷计数问题吗？为什么？如果我们将没有链路的两个节点连接起来，会出现什么情况？

P10. 讨论图 5-6 中的距离向量算法，距离向量 $D(x)$ 中的每个值不是递增的并且最终将在有限步中稳定下来。

P11. 考虑图 5-7。假定有另一台路由器 w，与路由器 y 和 z 连接。所有链路的开销给定如下：$c(x, y)=4$，$c(x, z)=50$，$c(y, w)=1$，$c(z, w)=1$，$c(y, z)=3$。假设在距离向量路由选择算法中使用了毒性逆转。

a. 当距离向量路由选择稳定时，路由器 w、y 和 z 向 x 通知它们之间的距离。它们告诉彼此什么样的距离值？

b. 现在假设 x 和 y 之间的链路开销增加到 60。如果使用了毒性逆转，将会存在无穷计数问题吗？为什么？如果存在无穷计数问题，距离向量路由选择需要多少次迭代才能再次到达稳定状态？评估你的答案。

c. 如果 $c(y, x)$ 从 4 变化到 60，怎样修改 $c(y, z)$ 使得不存在无穷计数问题。

P12. 描述在 BGP 中是如何检测路径中的环路的。

P13. BGP 路由器将总是选择具有最短 AS 路径长度的无环路由吗？评估你的答案。

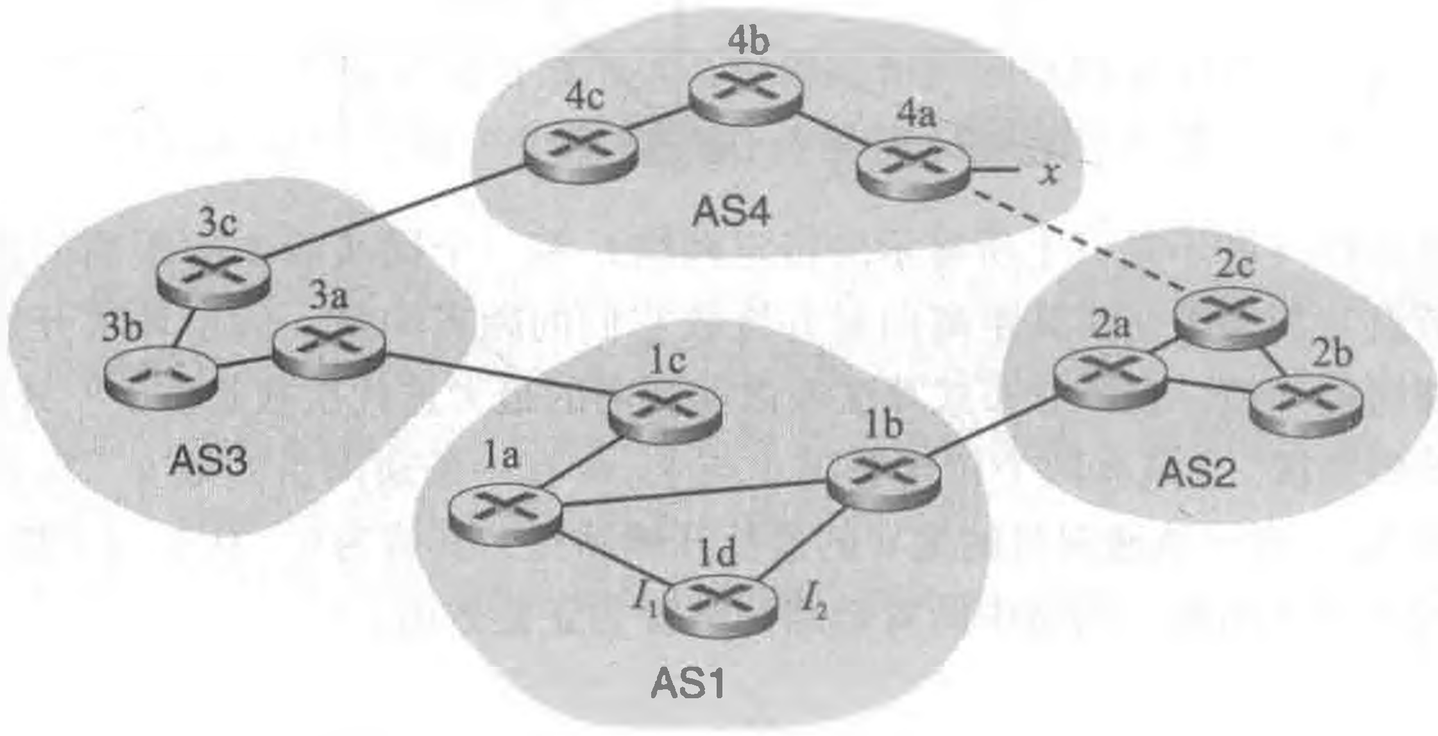
P14. 考虑下图所示的网络。假定 AS3 和 AS2 正在运行 OSPF 作为其 AS 内部路由选择协议。假定 AS1 和 AS4 正在运行 RIP 作为其 AS 内部路由选择协议。假定 AS 间路由选择协议使用的是 eBGP 和 iBGP。假定最初在 AS2 和 AS4 之间不存在物理链路。

a. 路由器 3c 从下列哪个路由选择协议学习到了前缀 x：OSPF、RIP、eBGP 或 iBGP？

b. 路由器 3a 从哪个路由选择协议学习到了前缀 x？

c. 路由器 1c 从哪个路由选择协议学习到了前缀 x？

d. 路由器 1d 从哪个路由选择协议学习到了前缀 x？



P15. 参考前面习题 P14，一旦路由器 1d 知道了 x 的情况，它将一个表项（x，I）放入它的转发表中。

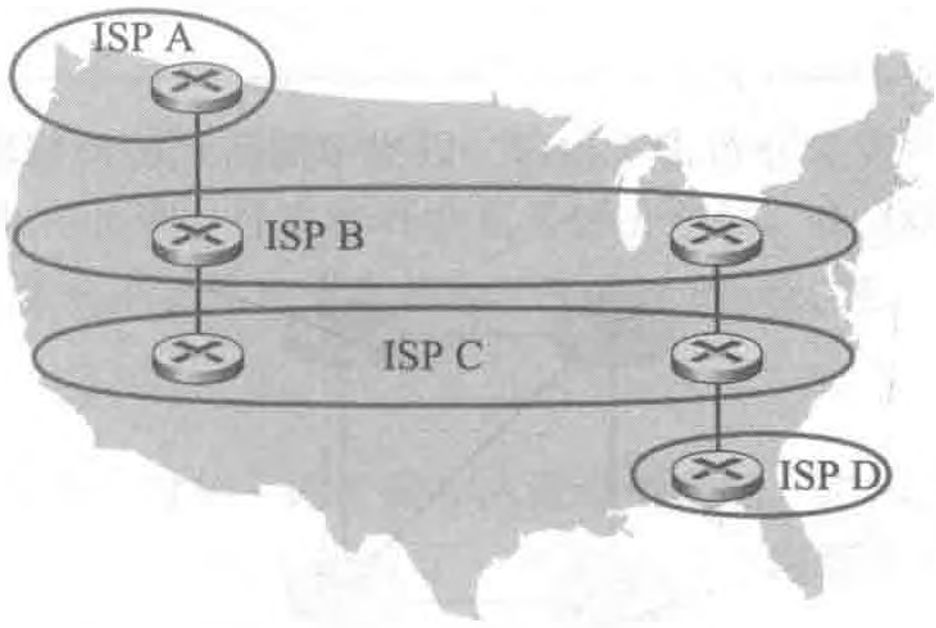
a. 对这个表项而言，I 将等于 I_1 还是 I_2？用一句话解释其原因。

b. 现在假定在 AS2 和 AS4 之间有一条物理链路，显示为图中的虚线。假定路由器 1d 知道经 AS2 以及经 AS3 能够访问到 x。I 将设置为 I_1 还是 I_2？用一句话解释其原因。

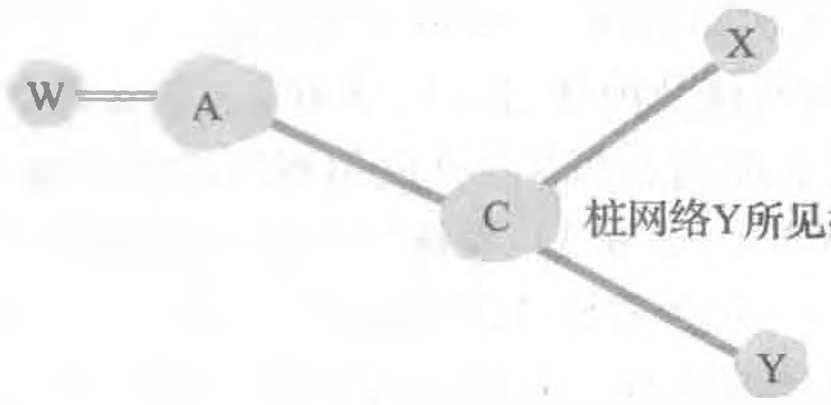
c. 现在假定有另一个 AS，它称为 AS5，其位于路径 AS2 和 AS4 之间（没有显示在图中）。假定路由

器 1d 知道经 AS2 AS5 AS4 以及经过 AS3 AS4 能够访问到 x。I 将设置为 I_1 还是 I_2？用一句话解释其原因。

P16. 考虑下面的网络。ISP B 为地区 ISP A 提供国家级主干服务。ISP C 为地区 ISP D 提供国家级主干服务。每个 ISP 由一个 AS 组成。B 和 C 使用 BGP，在两个地方互相对等。考虑从 A 到 D 的流量。B 愿意将流量交给 C 传给西海岸（使得 C 将承担承载跨越整个国家的流量开销），而 C 愿意经其东海岸与 B 对等的站点得到这些流量（使得 B 将承载跨越整个国家的流量）。C 可能会使用什么样的 BGP 机制，使得 B 将通过东海岸对等点传递 A 到 D 的流量？要回答这个问题，你需要钻研 BGP 规范。



P17. 在图 5-13 中，考虑到达桩网络 W、X 和 Y 的路径信息。基于 W 与 X 处的可用信息，它们分别看到的网络拓扑是什么？评估你的答案。Y 所见的拓扑视图如下图所示。



桩网络Y所见拓扑视图

P18. 考虑图 5-13。B 将不会基于 BGP 路由选择，经过 X 以 Y 为目的地转发流量。但是有某些极为流行的应用程序，其数据分组先朝向 X，然后再流向 Y。指出一种这样的应用程序，描述数据分组是如何沿着这条未由 BGP 路由选择所给定的路径流动的。

P19. 在图 5-13 中，假定有另一个桩网络 V，它为 ISP A 的客户。假设 B 和 C 具有对等关系，并且 A 是 B 和 C 的客户。假设 A 希望让发向 W 的流量仅来自 B，并且发向 V 的流量来自 B 或 C。A 如何向 B 和 C 通告其路由？C 收到什么样的 AS 路由？

P20. 假定 AS X 和 Z 不直接连接，但与 AS Y 连接。进一步假定 X 与 Y 具有对等协定，Y 与 Z 具有对等协定。最后，假定 Z 要传送所有 Y 的流量但不传送 X 的流量。BGP 允许 Z 实现这种策略吗？

P21. 考虑在管理实体和被管设备之间发生通信的两种方式：请求响应方式和陷阱方式。从以下方面考虑这两种方式的优缺点：①开销；②当异常事件出现时通知的时间；③对于管理实体和设备之间丢失报文的健壮性。

P22. 在 5.7 节中我们看到，用不可靠的 UDP 数据报传输 SNMP 报文是更可取的方式。请考虑 SNMP 设计者选择 UDP 而不是 TCP 作为 SNMP 运输协议的理由。

套接字编程作业

在第 2 章结尾给出了 4 个套接字编程作业。下面给出第 5 个应用 ICMP 的作业（ICMP 的是本章讨论的协议）。

作业 5：ICMP ping

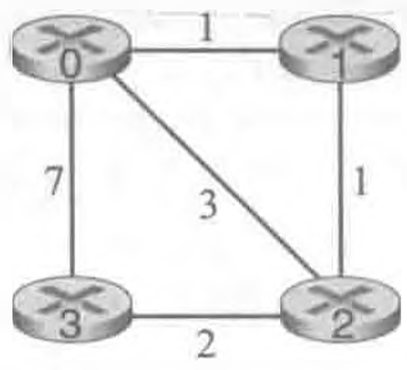
ping 是一种流行的网络应用程序，用于测试位于远程的某个特定的主机是否开机和可达。它也经常

用于测量客户主机和目标主机之间的时延。它的工作过程是：向目标主机发送 ICMP“回显请求”分组（即 ping 分组），并且侦听 ICMP“回显响应”应答（即 pong 分组）。ping 测量 RRT、记录分组丢失和计算多个 ping-pong 交换（往返时间的最小、平均、最大和标准差）的统计汇总。

在本实验中，读者将用 Python 语言编写自己的 ping 应用程序。该应用程序将使用 ICMP。但为了保持程序的简单，将不完全遵循 RFC 1739 中的官方规范。注意到仅需要写该程序的客户程序，因为服务器侧所需的功能构建在几乎所有的操作系统中。读者能够在 Web 站点 http://www.pearsonhighered.com/cs-resources 找到本作业的全面细节，以及该 Python 代码的重要片段。

编程作业

在本编程作业中，需要写一组“分布式”程序，以为下图所示的网络实现一个分布式异步距离向量路由选择算法。



要写出下列例程，这些例程将在为该作业提供的模拟环境中异步“执行”。对于节点 0，将要写出这样的例程：

- rtinit0()。在模拟开始将调用一次该例程。rtinit0()无参数。它应当初始化节点 0 中的距离表，以反映出到达节点 1、2 和 3 的直接开销分别为 1、3 和 7。在上图中，所有链路都是双向的，两个方向的开销皆相同。在初始化距离表和节点 0 的例程所需的其他数据结构后，它应向其直接连接的邻居（在本情况中为节点 1、2 和 3）发送它到所有其他网络节点的最低开销路径的开销信息。通过调用例程 tolayer2()，这种最低开销信息在一个路由选择更新分组中被发送给相邻节点，就像在完整编程作业中描述的那样。路由选择更新分组的格式也在完整编程作业中进行描述。
- rtupdate0(struct rtpkt *rcvdpkt)。当节点 0 收到一个由其直接相连邻居之一发给它的路由选择分组时，调用该例程。参数 *rcvdpkt 是一个指向接收分组的指针。rtupdate0()是距离向量算法的“核心”。它从其他节点 i 接收的路由选择更新分组中包含节点 i 到所有其他网络节点的当前最短路径开销值。rtupdate0()使用这些收到的值来更新其自身的距离表（这是由距离向量算法所规定的）。如果它自己到另外节点的最低开销由于此更新而发生改变的话，则节点 0 通过发送一个路由选择分组来通知其直接相连邻居这种最低开销的变化。我们在距离向量算法中讲过，仅有直接相连的节点才交换路由选择分组。因此，节点 1 和节点 2 将相互通信，但节点 1 和节点 3 将不相互通信。

为节点 1、2、3 定义类似的例程。因此你总共将写出 8 个例程：rtinit0()、rtinit1()、rtinit2()、rtinit3()、rtupdate0()、rtupdate1()、rtupdate2()和 rtupdate3()。这些例程将共同实现一个分布式的、与图中所示拓扑和开销相关的距离表的异步计算。

读者可在网址 http://www.pearsonhighered.com/cs-resources 处找到该编程作业的全部详细资料，以及创建模拟硬件/软件环境所需的 C 程序代码。一个 Java 版的编程作业也可供使用。

Wireshark 实验

在本书配套的 Web 站点 www.pearsonhighered.com/cs-resource 上，将找到一个 Wireshark 实验作业，该作业考察了在 ping 和 traceroute 命令中 ICMP 协议的使用。