- 经内存交换。最简单、最早的路由器是传统的计算机，在输入端口与输出端口之间的交换是在CPU（路由选择处理器）的直接控制下完成的。输入与输出端口的功能就像在传统操作系统中的I/O设备一样。一个分组到达一个输入端口时，该端口会先通过中断方式向路由选择处理器发出信号。于是，该分组从输入端口处被复制到处理器内存中。路由选择处理器则从其首部中提取目的地址，在转发表中找出适当的输出端口，并将该分组复制到输出端口的缓存中。在这种情况下，如果内存带宽为每秒可写进内存或从内存读出最多B个分组，则总的转发吞吐量（分组从输入端口被传送到输出端口的总速率）必然小于$B/2$。也要注意到不能同时转发两个分组，即使它们有不同的目的端口，因为经过共享系统总线一次仅能执行一个内存读/写。

  许多现代路由器通过内存进行交换。然而，与早期路由器的一个主要差别是，目的地址的查找和将分组存储（交换）进适当的内存存储位置是由输入线路卡来处理的。在某些方面，经内存交换的路由器看起来很像共享内存的多处理器，用一个线路卡上的处理将分组交换（写）进适当的输出端口的内存中。Cisco的Catalyst 8500系列的交换机［Cisco 8500 2016］是经共享内存转发分组的。
- 经总线交换。在这种方法中，输入端口经一根共享总线将分组直接传送到输出端口，不需要路由选择处理器的干预。通常按以下方式完成该任务：让输入端口为分组预先计划一个交换机内部标签（首部），指示本地输出端口，使分组在总线上传送和传输到输出端口。该分组能由所有输出端口收到，但只有与该标签匹配的端口才能保存该分组。然后标签在输出端口被去除，因为其仅用于交换机内部来跨越总线。如果多个分组同时到达路由器，每个位于不同的输出端口，除了一个分组外所有其他分组必须等待，因为一次只有一个分组能够跨越总线。因为每个分组必须跨过单一总线，故路由器的交换带宽受总线速率的限制；在环状交叉路的类比中，这相当于环状交叉路一次仅包含一辆汽车。尽管如此，对于运行在小型局域网和企业网中的路由器来说，通过总线交换通常足够用了。Cisco 6500路由器［Cisco 6500 2016］内部通过一个32Gbps背板总线来交换分组。
- 经互联网络交换。克服单一、共享式总线带宽限制的一种方法是，使用一个更复杂的互联网络，例如过去在多处理器计算机体系结构中用来互联多个处理器的网络。纵横式交换机就是一种由$2N$条总线组成的互联网络，它连接N个输入端口与N个输出端口，如图4-6所示。每条垂直的总线在交叉点与每条水平的总线交叉，交叉点通过交换结构控制器（其逻辑是交换结构自身的一部分）能够在任何时候开启和闭合。当某分组到达端口A，需要转发到端口Y时，交换机控制器闭合总线A和Y交叉部位的交叉点，然后端口A在其总线上发送该分组，该分组仅由总线Y接收。注意到来自端口B的一个分组在同一时间能够转发到端口X，因为A到Y和B到X的分组使用不同的输入和输出总线。因此，与前面两种交换方法不同，纵横式网络能够并行转发多个分组。纵横式交换机是**非阻塞的**（nonblocking），即只要没有其他分组当前被转发到该输出端口，转发到输出端口的分组将不会被到达输出端口的分组阻塞。然而，如果来自两个不同输入端口的两个分组其目的地为相同的输出端口，则一个分组必须在输入端等待，因为在某个时刻经给定总线仅能够发送一个分组。Cisco 12000系列交换机［Cisco 12000 2016］使用了一个互联网络；Cisco 7600系列能被配置为使用总线或者纵横式交换机

[Cisco 7600 2016]。

更为复杂的互联网络使用多级交换元素，以使来自不同输入端口的分组通过交换结构同时朝着相同的输出端口前行。对交换机体系结构的展望可参见[Tobagi 1990]。Cisco CRS 利用了一种三级非阻塞交换策略。路由器的交换能力也能够通过并行运行多种交换结构进行扩展。在这种方法中，输入端口和输出端口被连接到并行运行的 N 个交换结构。一个输入端口将一个分组分成 K 个较小的块，并且通过 N 个交换结构中的 K 个发送（“喷射”）这些块到所选择的输出端口，输出端口再将 K 个块装配还原成初始的分组。

4.2.3　输出端口处理

如图 4-7 中所示，输出端口处理取出已经存放在输出端口内存中的分组并将其发送到输出链路上。这包括选择和取出排队的分组进行传输，执行所需的链路层和物理层传输功能。

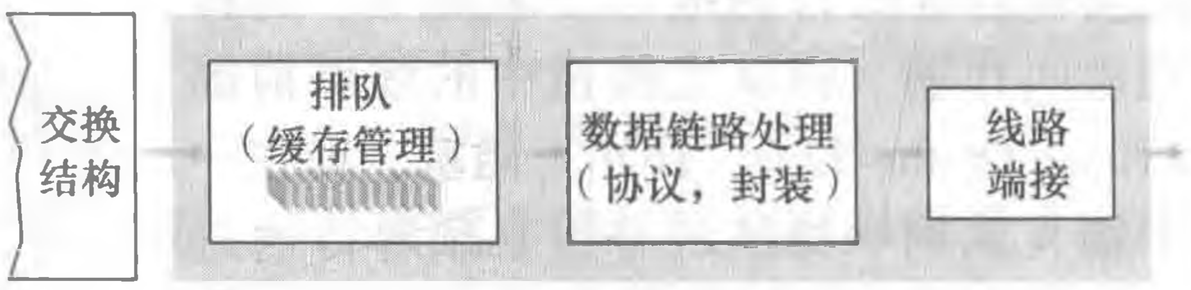

图 4-7　输出端口处理

4.2.4　何处出现排队

如果我们考虑显示在图 4-6 中的输入和输出端口功能及其配置，下列情况是一目了然的：在输入端口和输出端口处都可以形成分组队列，就像在环状交叉路的类比中我们讨论过的情况，即汽车可能等待在流量交叉点的入口和出口。排队的位置和程度（或者在输入端口排队，或者在输出端口排队）将取决于流量负载、交换结构的相对速率和线路速率。我们现在更为详细一点考虑这些队列，因为随着这些队列的增长，路由器的缓存空间最终将会耗尽，并且当无内存可用于存储到达的分组时将会出现**丢包**（packet loss）。回想前面的讨论，我们说过分组“在网络中丢失”或“被路由器丢弃”。正是在一台路由器的这些队列中，这些分组被实际丢弃或丢失。

假定输入线路速度与输出线路速度（传输速率）是相同的，均为 R_{line}（单位为每秒分组数），并且有 N 个输入端口和 N 个输出端口。为进一步简化讨论，假设所有分组具有相同的固定长度，分组以同步的方式到达输入端口。这就是说，在任何链路发送分组的时间等于在任何链路接收分组的时间，在这样的时间间隔内，在一个输入链路上能够到达 0 个或 1 个分组。定义交换结构传送速率 R_{switch} 为从输入端口到输出端口能够移动分组的速率。如果 R_{switch} 比 R_{line} 快 N 倍，则在输入端口处仅会出现微不足道的排队。这是因为即使在最坏情况下，所有 N 条输入线路都在接收分组，并且所有的分组将被转发到相同的输出端口，每批 N 个分组（每个输入端口一个分组）也能够在下一批到达前通过交换结构处理完毕。

1. 输入排队

如果交换结构不能快得（相对于输入线路速度而言）使所有到达分组无时延地通过它传送，会发生什么情况呢？在这种情况下，在输入端口也将出现分组排队，因为到达的分组必须加入输入端口队列中，以等待通过交换结构传送到输出端口。为了举例说明这种排队的重要后果，考虑纵横式交换结构，并假定：①所有链路速度相同；②一个分组能够以一条输入链路接收一个分组所用的相同的时间量，从任意一个输入端口传送到给定的输出

端口；③分组按 FCFS 方式，从一指定输入队列移动到其要求的输出队列中。只要其输出端口不同，多个分组可以被并行传送。然而，如果位于两个输入队列前端的两个分组是发往同一输出队列的，则其中的一个分组将被阻塞，且必须在输入队列中等待，因为交换结构一次只能传送一个分组到某指定端口。

图 4-8 显示了一个例子，其中在输入队列前端的两个分组（带深色阴影）要发往同一个右上角输出端口。假定该交换结构决定发送左上角队列前端的分组。在这种情况下，左下角队列中的深色阴影分组必须等待。但不仅该分组要等待，左下角队列中排在该分组后面的浅色阴影分组也要等待，即使右中侧输出端口（浅色阴影分组的目的地）中无竞争。这种现象叫作输入排队交换机中的**线路前部**（Head-Of-the-Line，HOL）**阻塞**，即在一个输入队列中排队的分组必须等待通过交换结构发送（即使输出端口是空闲的），因为它被位于线路前部的另一个分组所阻塞。[Karol 1987] 指出，由于 HOL 阻塞，只要输入链路上的分组到达速率达到其容量的 58%，在某些假设前提下，输入队列长度就将无限制地增大（不严格地讲，这等同于说将出现大量的丢包）。[McKeown 1997b] 讨论了多种解决 HOL 阻塞的方法。

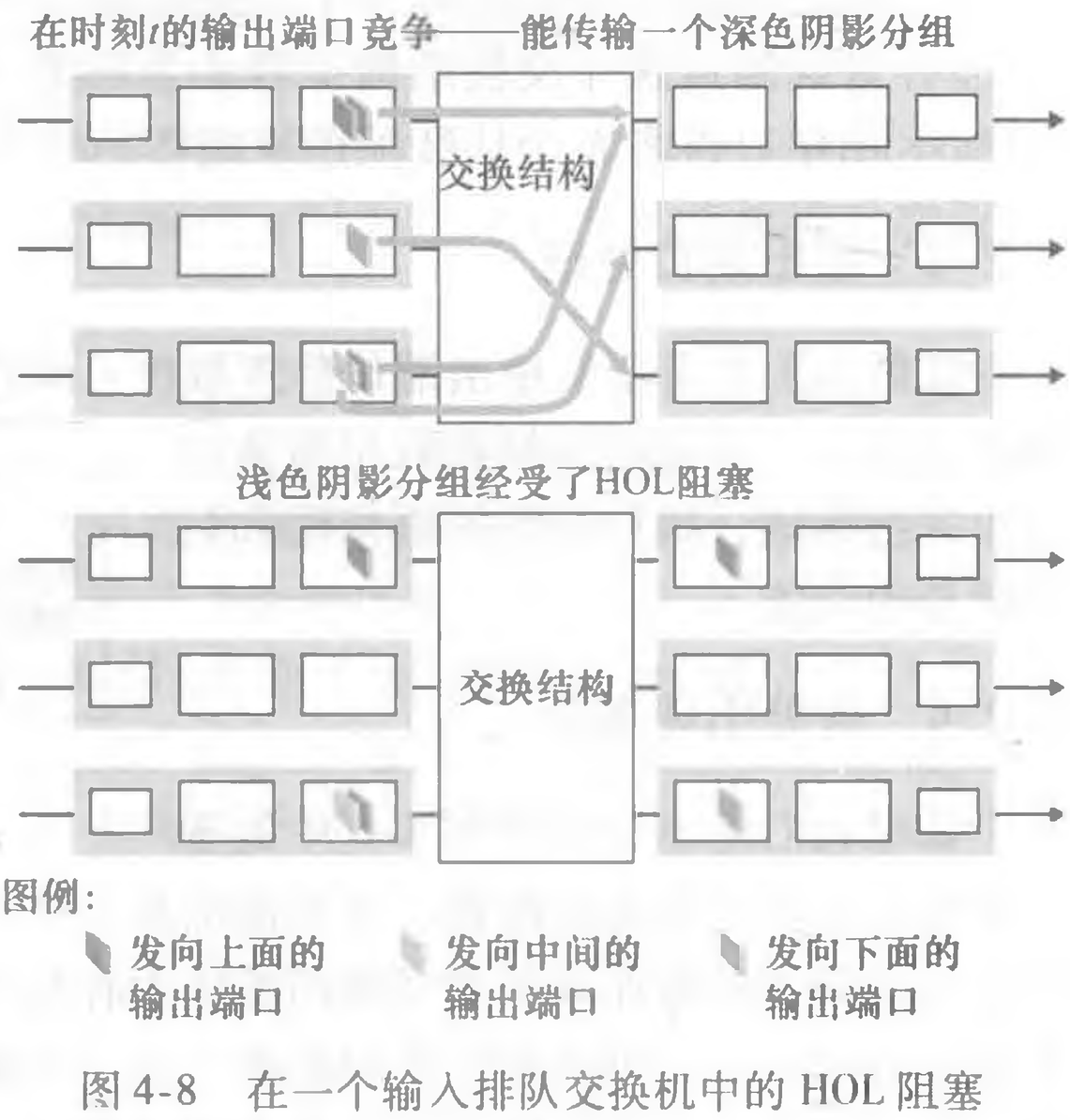

图 4-8　在一个输入排队交换机中的 HOL 阻塞

2. 输出排队

我们接下来考虑在交换机的输出端口是否会出现排队。再次假定 R_{switch} 比 R_{line} 快 N 倍，并且到达 N 个输入端口的每个端口的分组，其目的地是相同的输出端口。在这种情况下，在向输出链路发送一个分组的时间内，将有 N 个新分组到达该输出端口（N 个输入端口的每个都到达 1 个）。因为输出端口在一个单位时间（该分组的传输时间）内仅能传输一个分组，这 N 个到达分组必须排队（等待）经输出链路传输。在正好传输 N 个分组（这些分组是前面正在排队的）之一的时间中，可能又到达 N 个分组，等等。所以，分组队列能够在输出端口形成，即使交换结构比端口线路速率快 N 倍。最终，排队的分组数量能够变得足够大，耗尽输出端口的可用内存。

当没有足够的内存来缓存一个入分组时，就必须做出决定：要么丢弃到达的分组（采用一种称为**弃尾**（drop-tail）的策略），要么删除一个或多个已排队的分组为新来的分组腾出空间。在某些情况下，在缓存填满之前便丢弃一个分组（或在其首部加上标记）的做法是有利的，这可以向发送方提供一个拥塞信号。已经提出和分析了许多分组丢弃与标记策略 [Labrador 1999，Hollot 2002]，这些策略统称为**主动队列管理**（Active Queue Management，AQM）算法。**随机早期检测**（Random Early Detection，RED）算法是得到最广泛研究和实现的 AQM 算法之一 [Christiansen 2001；Floyd 2016]。

在图4-9中图示了输出端口的排队情况。在时刻t，每个入端输入端口都到达了一个分组，每个分组都是发往最上侧的输出端口。假定线路速度相同，交换机以3倍于线路速度的速度运行，一个时间单位（即接收或发送一个分组所需的时间）以后，所有三个初始分组都被传送到输出端口，并排队等待传输。在下一个时间单位中，这三个分组中的一个将通过输出链路发送出去。在这个例子中，又有两个新分组已到达交换机的入端；这些分组之一要发往最上侧的输出端口。这样的后果是，输出端口的**分组调度**（packet scheduler）在这些排队分组中选择一个分组来传输，这就是我们将在下节中讨论的主题。

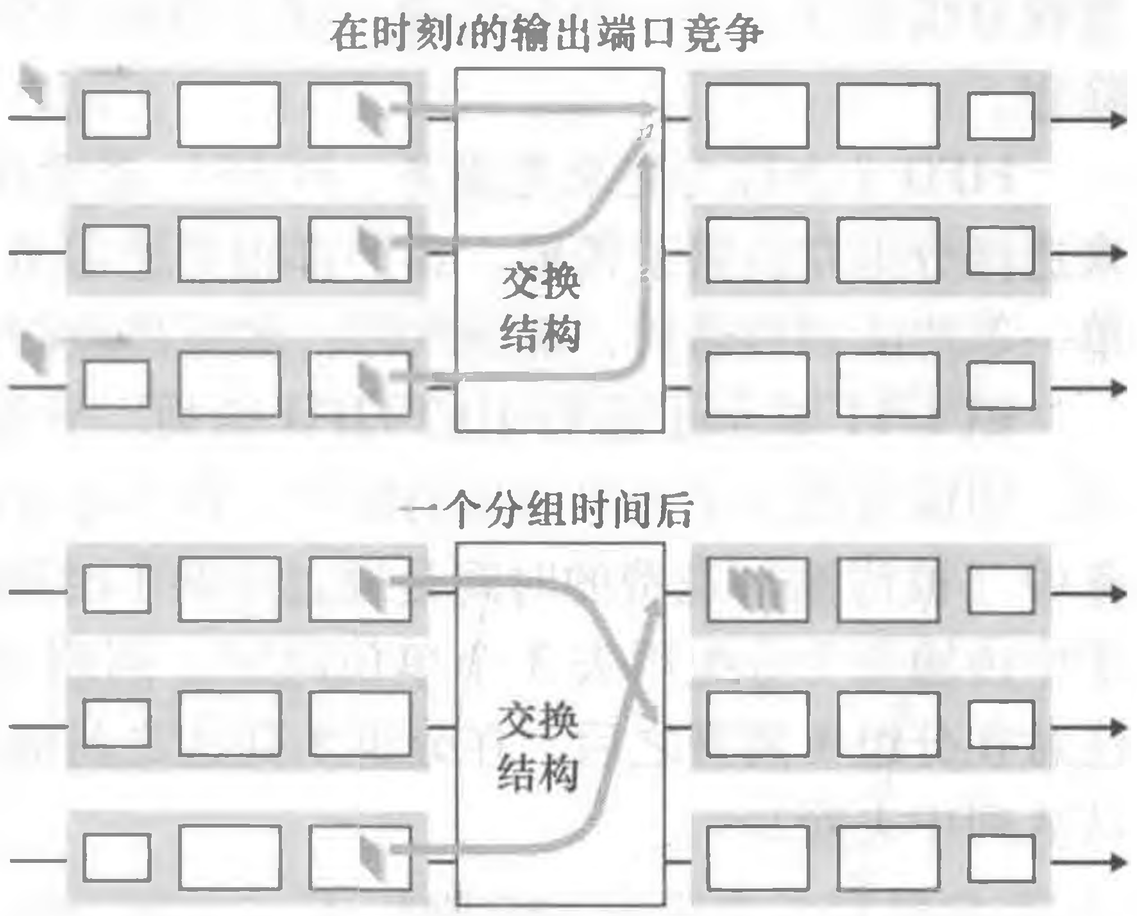

图4-9　输出端口排队

假定需要路由器缓存来吸收流量负载的波动，一个自然而然的问题就是需要多少缓存。多年以来，用于缓存长度的经验方法是［RFC 3439］，缓存数量（B）应当等于平均往返时延（RTT，比如说250ms）乘以链路的容量（C）。这个结果是基于相对少量的TCP流的排队动态性分析得到的［Villamizar 1994］。因此，一条具有250ms RTT的10Gbps链路需要的缓存量等于$B = \text{RTT} \cdot C = 2.5\text{Gb}$。然而，最近的理论和试验研究［Appenzeller 2004］表明，当有大量的TCP流（N条）流过一条链路时，缓存所需要的数量是$B = \text{RTT} \cdot C/\sqrt{N}$。对于通常有大量流经过的大型主干路由器链路（参见如［Fraleigh 2003］），N的值可能非常大，所需的缓存长度的减小相当明显。［Appenzeller 2004；Wischik 2005；Beheshti 2008］从理论、实现和运行的角度提供了可读性很强的有关缓存长度问题的讨论。

4.2.5　分组调度

现在我们转而讨论确定次序的问题，即排队的分组如何经输出链路传输的问题。以前你自己无疑在许多场合都排长队等待过，并观察过等待的客户怎样被服务，你无疑也熟悉路由器中常用的许多排队规则。有一种是先来先服务（FCFS，也称之为先进先出（FIFO））。这是英国人人共知的规则，用于病人就诊、公交车站和市场中的有序FCFS队列。（哦，你排队了吗?）有些国家基于优先权运转，即给一类等待客户超越其他等待客户的优先权服务。也有循环排队，其中客户也被划分为类别（与在优先权队列一样），但每类用户依次序提供服务。

1. 先进先出

图4-10显示了对于**先进先出**（First-In-First-Out，FIFO）链路调度规则的排队模型的抽象。如果链路当前正忙于传输另一个分组，到达链路输出队列的分组要排队等待传输。如果没有足够的缓存空间来容纳到达的分组，队列的分组丢弃策略则确定该分组是否将被丢弃（丢失）或者

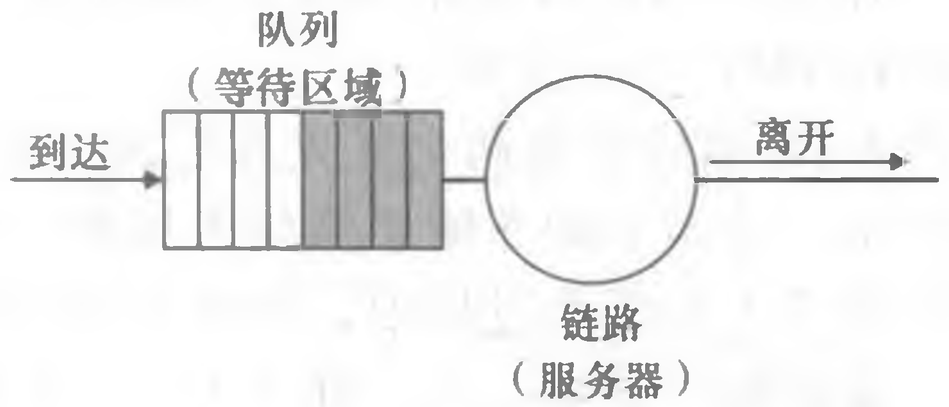

图4-10　FIFO排队抽象

从队列中去除其他分组以便为到达的分组腾出空间，如前所述。在下面的讨论中，我们将忽视分组丢弃。当一个分组通过输出链路完全传输（也就是接收服务）时，从队列中去除它。

FIFO（也称为先来先服务，FCFS）调度规则按照分组到达输出链路队列的相同次序来选择分组在链路上传输。我们都很熟悉服务中心的 FIFO 排队，在那里到达的顾客加入单一等待队列的最后，保持次序，然后当他们到达队伍的前面时就接受服务。

图 4-11 显示了运行中的 FIFO 队列。分组的到达由上部时间线上带编号的箭头来指示，用编号指示了分组到达的次序。各个分组的离开表示在下部时间线的下面。分组在服务中（被传输）花费的时间是通过这两个时间线之间的阴影矩形来指示的。假定在这个例子中传输每个分组用去 3 个单位时间。利用 FIFO 规则，分组按照到达的相同次序离开。注意在分组 4 离开之后，在分组 5 到达之前链路保持空闲（因为分组 1 ~ 4 已经被传输并从队列中去除）。

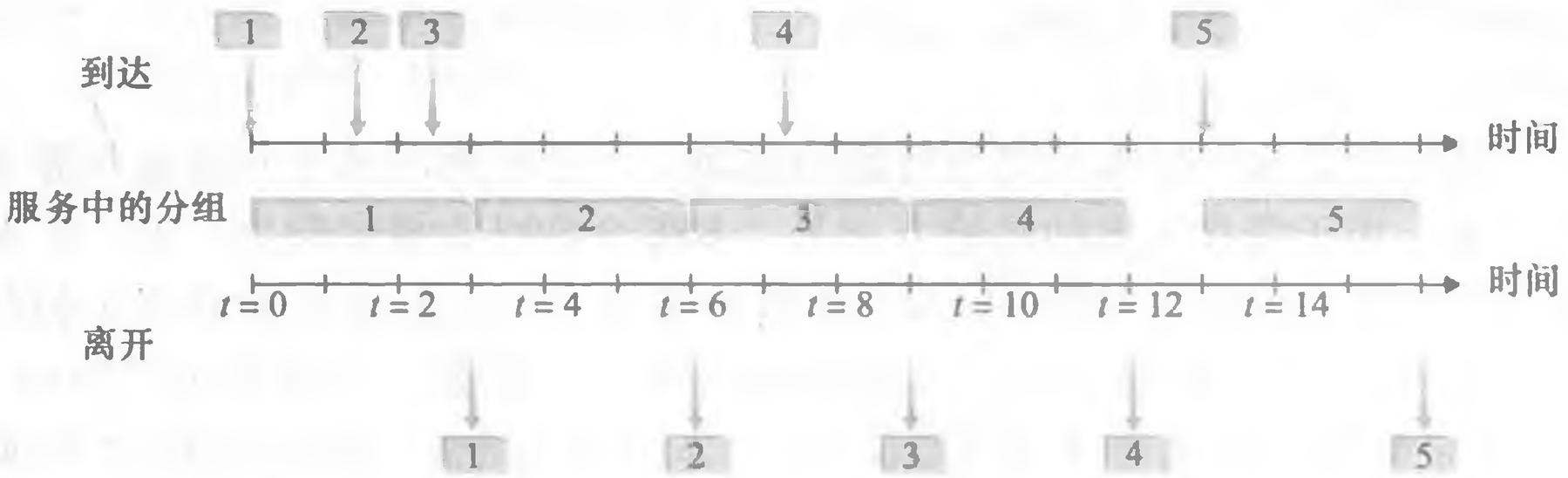

图 4-11　运行中的 FIFO 队列

2. 优先权排队

在**优先权排队**（priority queuing）规则下，到达输出链路的分组被分类放入输出队列中的优先权类，如图 4-12 所示。在实践中，网络操作员可以配置一个队列，这样携带网络管理信息的分组（例如，由源或目的 TCP/UDP 端口号所标识）获得超过用户流量的优先权；此外，基于 IP 的实时话音分组可能获得超过非实时流量（如 SMTP 或 IMAP 电子邮件分组）的优先权。每个优先权类通常都有自己的队列。当选择一个分组传输时，优先权排队规则将从队列为非空（也就是有分组等待传输）的最高优先权类中传输一个分组。在同一优先权类的分组之间的选择通常以 FIFO 方式完成。

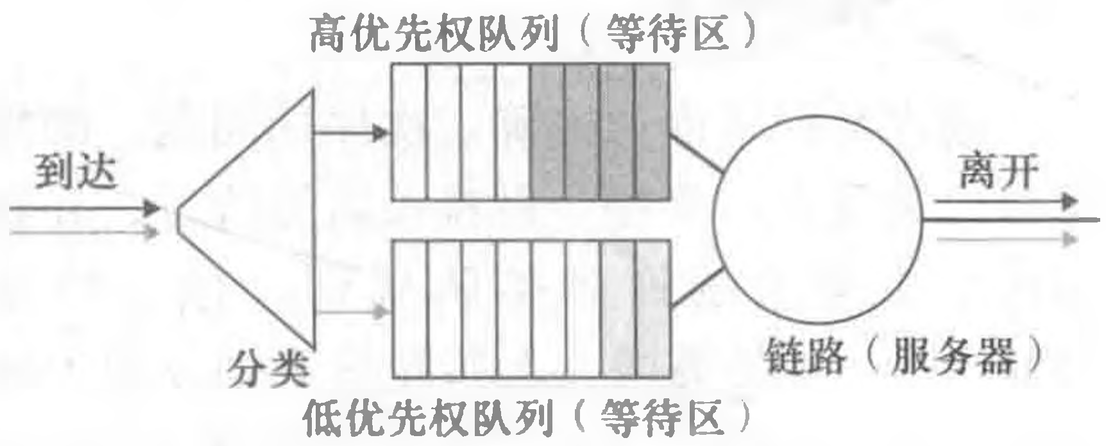

图 4-12　优先权排队模型

图 4-13 描述了有两个优先权类的一个优先权队列的操作。分组 1、3 和 4 属于高优先权类，分组 2 和 5 属于低优先权类。分组 1 到达并发现链路是空闲的，就开始传输。在分组 1 的传输过程中，分组 2 和 3 到达，并分别在低优先权和高优先权队列中排队。在传输完分组 1 后，分组 3（一个高优先权的分组）被选择在分组 2（尽管它到达得较早，但它是一个低优先权分组）之前传输。在分组 3 的传输结束后，分组 2 开始传输。分组 4（一个高优先权分组）在分组 2（一个低优先权分组）的传输过程中到达。在**非抢占式优先权排队**（non-preemptive priority queuing）规则下，一旦分组

开始传输，就不能打断。在这种情况下，分组 4 排队等待传输，并在分组 2 传输完成之后开始传输。

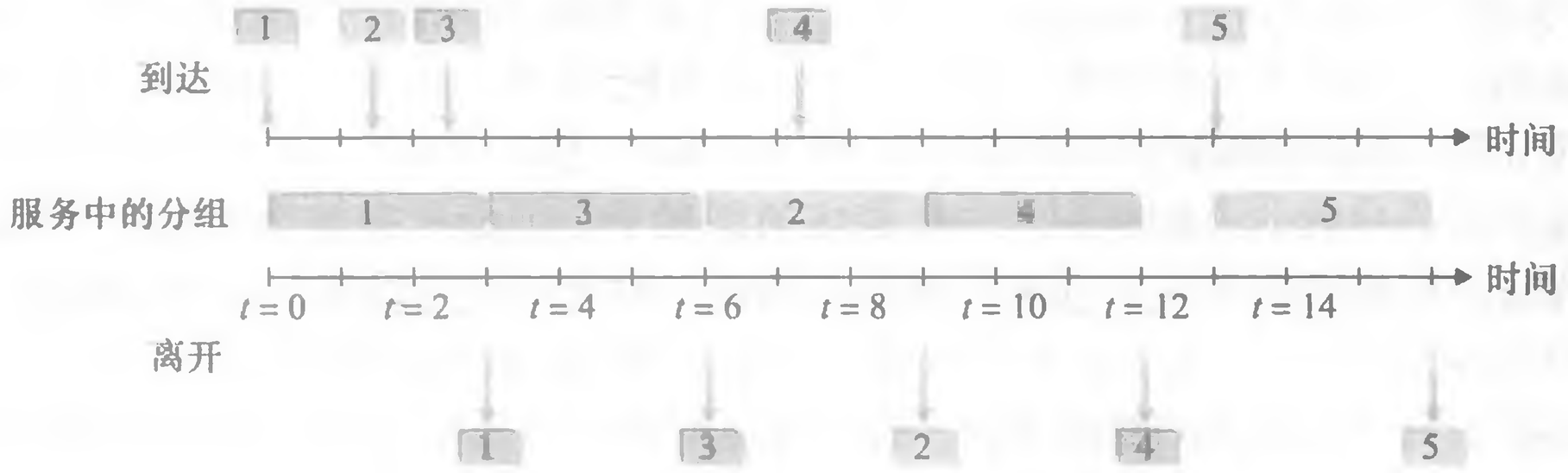

图 4-13　优先权队列的操作

3. 循环和加权公平排队

在**循环排队规则**（round robin queuing discipline）下，分组像使用优先权排队那样被分类。然而，在类之间不存在严格的服务优先权，循环调度器在这些类之间轮流提供服务。在最简单形式的循环调度中，类 1 的分组被传输，接着是类 2 的分组，接着又是类 1 的分组，再接着又是类 2 的分组，等等。一个所谓的**保持工作排队**（work-conserving queuing）规则在有（任何类的）分组排队等待传输时，不允许链路保持空闲。当寻找给定类的分组但是没有找到时，保持工作的循环规则将立即检查循环序列中的下一个类。

图 4-14 描述了一个两类循环队列的操作。在这个例子中，分组 1、2 和 4 属于第一类，分组 3 和 5 属于第二类。分组 1 一到达输出队列就立即开始传输。分组 2 和 3 在分组 1 的传输过程中到达，因此排队等待传输。在分组 1 传输后，链路调度器查找类 2 的分组，因此传输分组 3。在分组 3 传输完成后，调度器查找类 1 的分组，因此传输分组 2。在分组 2 传输完成后，分组 4 是唯一排队的分组，因此在分组 2 后立刻传输分组 4。

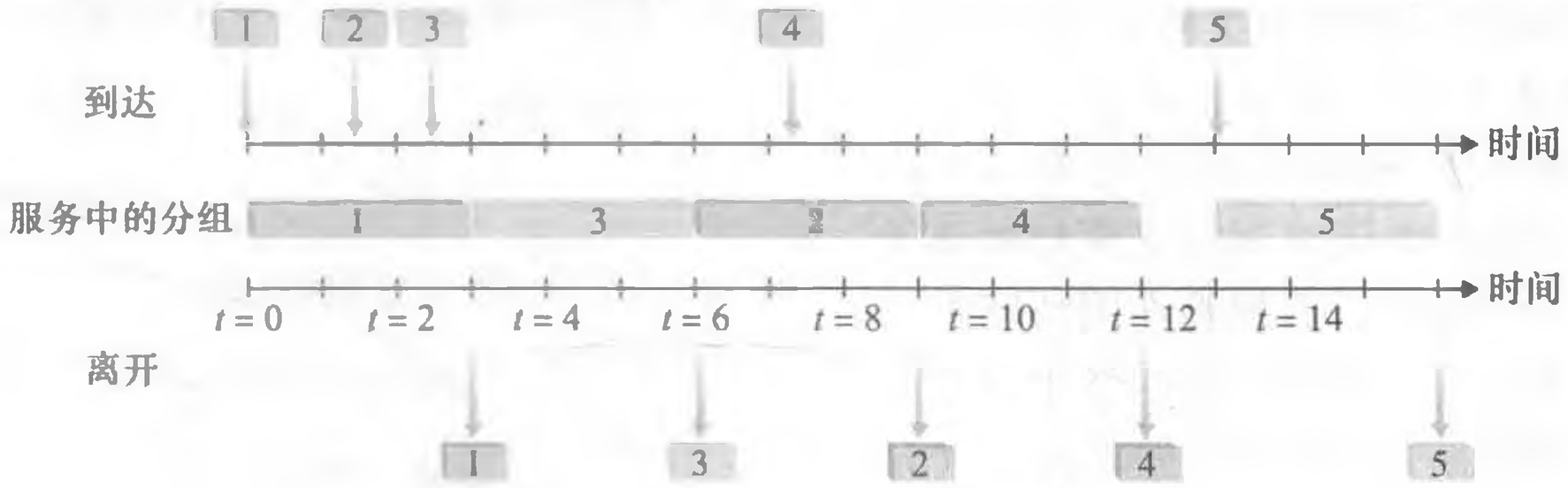

图 4-14　两类循环队列的操作

一种通用形式的循环排队已经广泛地实现在路由器中，它就是所谓的**加权公平排队**（Weighted Fair Queuing，WFQ）规则[Demers 1990；Parekh 1993；Cisco QoS 2016]。图 4-15 对WFQ 进行了描述。其中，到达的分组被分类并在合适的每个类的等待区域排队。与使用循环调度一样，WFQ 调度器也以循环的方式为各个类提供服务，即首先服务第 1 类，然后服务第 2 类，接着再服务第 3 类，然后

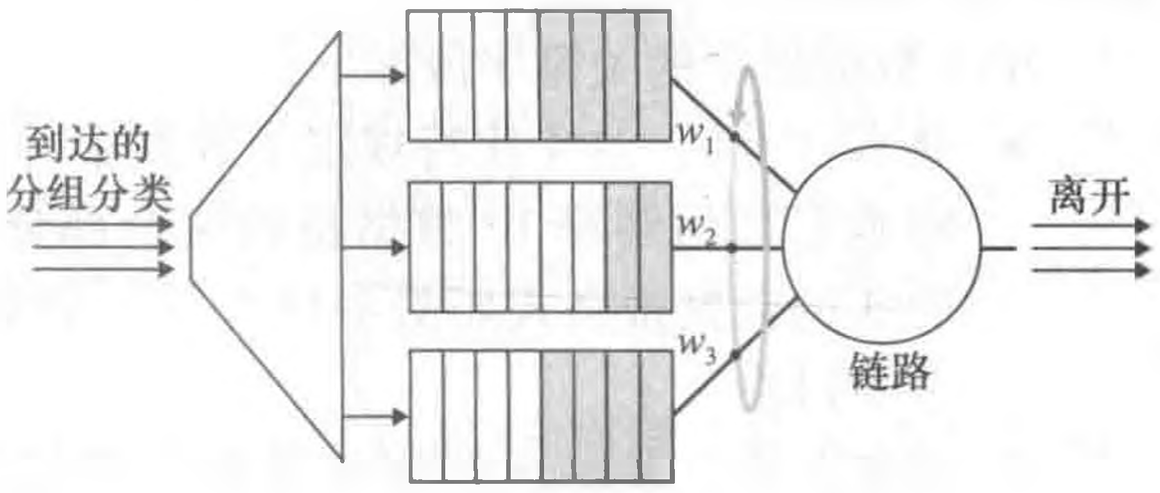

图 4-15　加权公平排队

（假设有3个类别）重复这种服务模式。WFQ也是一种保持工作排队规则，因此在发现一个空的类队列时，它立即移向服务序列中的下一个类。

WFQ和循环排队的不同之处在于，每个类在任何时间间隔内可能收到不同数量的服务。具体而言，每个类i被分配一个权w_i。使用WFQ方式，在类i有分组要发送的任何时间间隔中，第i类将确保接收到的服务部分等于$w_i/(\sum w_j)$，式中分母中的和是计算所有有分组排队等待传输的类别得到的。在最坏的情况下，即使所有的类都有分组排队，第i类仍然保证分配到带宽的$w_i/(\sum w_j)$部分。因此，对于一条传输速率为R的链路，第i类总能获得至少为$R \cdot w_i/(\sum w_j)$的吞吐量。我们对WFQ的描述理想化了，因为没有考虑这样的事实：分组是离散的数据单元，并且不能打断一个分组的传输来开始传输另一个分组；[Demers 1990；Parekh 1993]讨论了这个分组化问题。

4.3　网际协议：IPv4、寻址、IPv6及其他

到目前为止，我们在第4章中对网络层的学习，包括网络层的数据平面和控制平面组件概念，转发和路由选择之间的区别，各种网络服务模型的标识和对路由器内部的观察，并未提及任何特定的计算机网络体系结构或协议。在这节中，我们将关注点转向今天的因特网网络层的关键方面和著名的网际协议（IP）。

今天有两个版本的IP正在使用。在4.3.1节中，我们首先研究广泛部署的IP版本4，这通常简单地称为IPv4［RFC 791］。在4.3.5节中，我们将仔细考察IP版本6［RFC 2460；RFC 4291］，它已经被提议替代IPv4。在中间，我们将主要学习因特网编址，这是一个看起来相当枯燥和面向细节的主题，但是这对理解因特网网络层如何工作是至关重要的。掌握IP编址就是掌握因特网的网络层！

4.3.1　IPv4数据报格式

前面讲过网络层分组被称为数据报。我们以概述IPv4数据报的语法和语义开始对IP的学习。你也许认为没有什么比一个分组的比特的语法和语义更加枯燥无味的了。无论如何，数据报在因特网中起着重要作用，每个网络行业的学生和专业人员都需要理解它、吸收它并掌握它（只是理解协议首部的确能够使学习成为有趣的事，请查阅［Pomeranz 2010］）。IPv4数据报格式如图4-16所示。

图4-16　IPv4数据报格式

IPv4数据报中的关键字段如下：

- 版本（号）。这4比特规定了数据报的IP协议版本。通过查看版本号，路由器能够确定如何解释IP数据报的剩余部分。不同的IP版本使用不同的数据报格式。IPv4的数据报格式如图4-16所示。新版本的IP（IPv6）的数据报格式将在4.3.5节中讨论。
- 首部长度。因为一个IPv4数据报可包含一些可变数量的选项（这些选项包括在IPv4数据报首部中），故需要用这4比特来确定IP数据报中载荷（例如在这个数

据报中被封装的运输层报文段）实际开始的地方。大多数 IP 数据报不包含选项，所以一般的 IP 数据报具有 20 字节的首部。

- 服务类型。服务类型（TOS）比特包含在 IPv4 首部中，以便使不同类型的 IP 数据报（例如，一些特别要求低时延、高吞吐量或可靠性的数据报）能相互区别开来。例如，将实时数据报（如用于 IP 电话应用）与非实时流量（如 FTP）区分开也许是有用的。提供特定等级的服务是一个由网络管理员对路由器确定和配置的策略问题。我们在 3.7.2 节讨论明确拥塞通告所使用的两个 TOS 比特时也学习过。
- 数据报长度。这是 IP 数据报的总长度（首部加上数据），以字节计。因为该字段长为 16 比特，所以 IP 数据报的理论最大长度为 65 535 字节。然而，数据报很少有超过 1500 字节的，该长度使得 IP 数据报能容纳最大长度以太网帧的载荷字段。
- 标识、标志、片偏移。这三个字段与所谓 IP 分片有关，这是一个我们将很快要考虑的主题。有趣的是，新版本的 IP（即 IPv6）不允许在路由器上对分组分片。
- 寿命。寿命（Time-To-Live，TTL）字段用来确保数据报不会永远（如由于长时间的路由选择环路）在网络中循环。每当一台路由器处理数据报时，该字段的值减 1。若TTL字段减为 0，则该数据报必须丢弃。
- 协议。该字段通常仅当一个 IP 数据报到达其最终目的地时才会有用。该字段值指示了 IP 数据报的数据部分应交给哪个特定的运输层协议。例如，值为 6 表明数据部分要交给 TCP，而值为 17 表明数据要交给 UDP。对于所有可能值的列表，参见［IANA Protocol Numbers 2016］。注意在 IP 数据报中的协议号所起的作用，类似于运输层报文段中端口号字段所起的作用。协议号是将网络层与运输层绑定到一起的黏合剂，而端口号是将运输层和应用层绑定到一起的黏合剂。我们将在第 6 章看到，链路层帧也有一个特殊字段用于将链路层与网络层绑定到一起。
- 首部检验和。首部检验和用于帮助路由器检测收到的 IP 数据报中的比特错误。首部检验和是这样计算的：将首部中的每 2 个字节当作一个数，用反码算术对这些数求和。如在 3.3 节讨论的那样，该和的反码（被称为因特网检验和）存放在检验和字段中。路由器要对每个收到的 IP 数据报计算其首部检验和，如果数据报首部中携带的检验和与计算得到的检验和不一致，则检测出是个差错。路由器一般会丢弃检测出错误的数据报。注意到在每台路由器上必须重新计算检验和并再次存放到原处，因为TTL字段以及可能的选项字段会改变。关于计算因特网检验和的快速算法的有趣讨论参见［RFC 1071］。此时，一个经常问的问题是：为什么 TCP/IP 在运输层与网络层都执行差错检测？这种重复检测有几种原因。首先，注意到在 IP 层只对 IP 首部计算了检验和，而 TCP/UDP 检验和是对整个 TCP/UDP 报文段进行的。其次，TCP/UDP 与 IP 不一定都必须属于同一个协议栈。原则上，TCP 能够运行在一个不同的协议（如 ATM）上［Black 1995］，而 IP 能够携带不一定要传递给 TCP/UDP 的数据。
- 源和目的 IP 地址。当某源生成一个数据报时，它在源 IP 字段中插入它的 IP 地址，在目的 IP 地址字段中插入其最终目的地的地址。通常源主机通过 DNS 查找来决定目的地址，如在第 2 章中讨论的那样。我们将在 4.3.3 节中详细讨论 IP

编址。

- 选项。选项字段允许 IP 首部被扩展。首部选项意味着很少使用，因此决定对每个数据报首部不包括选项字段中的信息，这样能够节约开销。然而，少量选项的存在的确使问题复杂了，因为数据报首部长度可变，故不能预先确定数据字段从何处开始。而且还因为有些数据报要求处理选项，而有些数据报则不要求，故导致一台路由器处理一个 IP 数据报所需的时间变化可能很大。这些考虑对于高性能路由器和主机上的 IP 处理来说特别重要。由于这样或那样的原因，在 IPv6 首部中已去掉了 IP 选项，如 4.3.5 节中讨论的那样。
- 数据（有效载荷）。我们来看看最后也是最重要的字段，这是数据报存在的首要理由！在大多数情况下，IP 数据报中的数据字段包含要交付给目的地的运输层报文段（TCP 或 UDP）。然而，该数据字段也可承载其他类型的数据，如 ICMP 报文（在 5.6 节中讨论）。

注意到一个 IP 数据报有总长为 20 字节的首部（假设无选项）。如果数据报承载一个 TCP 报文段，则每个（无分片的）数据报共承载了总长 40 字节的首部（20 字节的 IP 首部加上 20 字节的 TCP 首部）以及应用层报文。

4.3.2　IPv4 数据报分片

在第 6 章中我们将看到，并不是所有链路层协议都能承载相同长度的网络层分组。有的协议能承载大数据报，而有的协议只能承载小分组。例如，以太网帧能够承载不超过 1500 字节的数据，而某些广域网链路的帧可承载不超过 576 字节的数据。一个链路层帧能承载的最大数据量叫作**最大传送单元**（Maximum Transmission Unit，MTU）。因为每个 IP 数据报封装在链路层帧中从一台路由器传输到下一台路由器，故链路层协议的 MTU 严格地限制着 IP 数据报的长度。对 IP 数据报长度具有严格限制并不是主要问题。问题在于在发送方与目的地路径上的每段链路可能使用不同的链路层协议，且每种协议可能具有不同的 MTU。

为了更好地理解这一转发问题，想象你是一台互联几条链路的路由器，且每条链路运行具有不同 MTU 的链路层协议。假定你从某条链路收到一个 IP 数据报，通过检查转发表确定出链路，并且该条出链路的 MTU 比该 IP 数据报的长度要小。此时你会感到慌乱，如何将这个过大的 IP 分组挤进链路层帧的有效载荷字段呢？解决该问题的方法是将 IP 数据报中的数据分片成两个或更多个较小的 IP 数据报，用单独的链路层帧封装这些较小的 IP 数据报，然后通过输出链路发送这些帧。每个这些较小的数据报都称为片（fragment）。

片在其到达目的地运输层以前需要重新组装。TCP 与 UDP 的确都希望从网络层收到完整的、未分片的报文。IPv4 的设计者感到在路由器中重新组装数据报会给协议带来相当大的复杂性并且影响路由器的性能。（如果你是一台路由器，你愿意将重新组装报文片放在你必须要做的各种各样工作的首位吗？）为坚持网络内核保持简单的原则，IPv4 的设计者决定将数据报的重新组装工作放到端系统中，而不是放到网络路由器中。

当一台目的主机从相同源收到一系列数据报时，它需要确定这些数据报中的某些是否是一些原来较大的数据报的片。如果某些数据报是这些片的话，则它必须进一步确定何时收到了最后一片，并且如何将这些接收到的片拼接到一起以形成初始的数据报。为了让目

的主机执行这些重新组装任务，IPv4 的设计者将标识、标志和片偏移字段放在 IP 数据报首部中。当生成一个数据报时，发送主机在为该数据报设置源和目的地址的同时贴上标识号。发送主机通常将它发送的每个数据报的标识号加 1。当某路由器需要对一个数据报分片时，形成的每个数据报（即片）具有初始数据报的源地址、目的地址与标识号。当目的地从同一发送主机收到一系列数据报时，它能够检查数据报的标识号以确定哪些数据报实际上是同一较大数据报的片。由于 IP 是一种不可靠的服务，一个或多个片可能永远到达不了目的地。因为这种原因，为了让目的主机绝对地相信它已收到了初始数据报的最后一个片，最后一个片的标志比特被设为 0，而所有其他片的标志比特被设为 1。另外，为了让目的主机确定是否丢失了一个片（且能按正确的顺序重新组装片），使用偏移字段指定该片应放在初始 IP 数据报的哪个位置。

图 4-17 图示了一个例子。一个 4000 字节的数据报（20 字节 IP 首部加上 3980 字节 IP 有效载荷）到达一台路由器，且必须被转发到一条 MTU 为 1500 字节的链路上。这就意味着初始数据报中 3980 字节数据必须被分配为 3 个独立的片（其中的每个片也是一个 IP 数据报）。

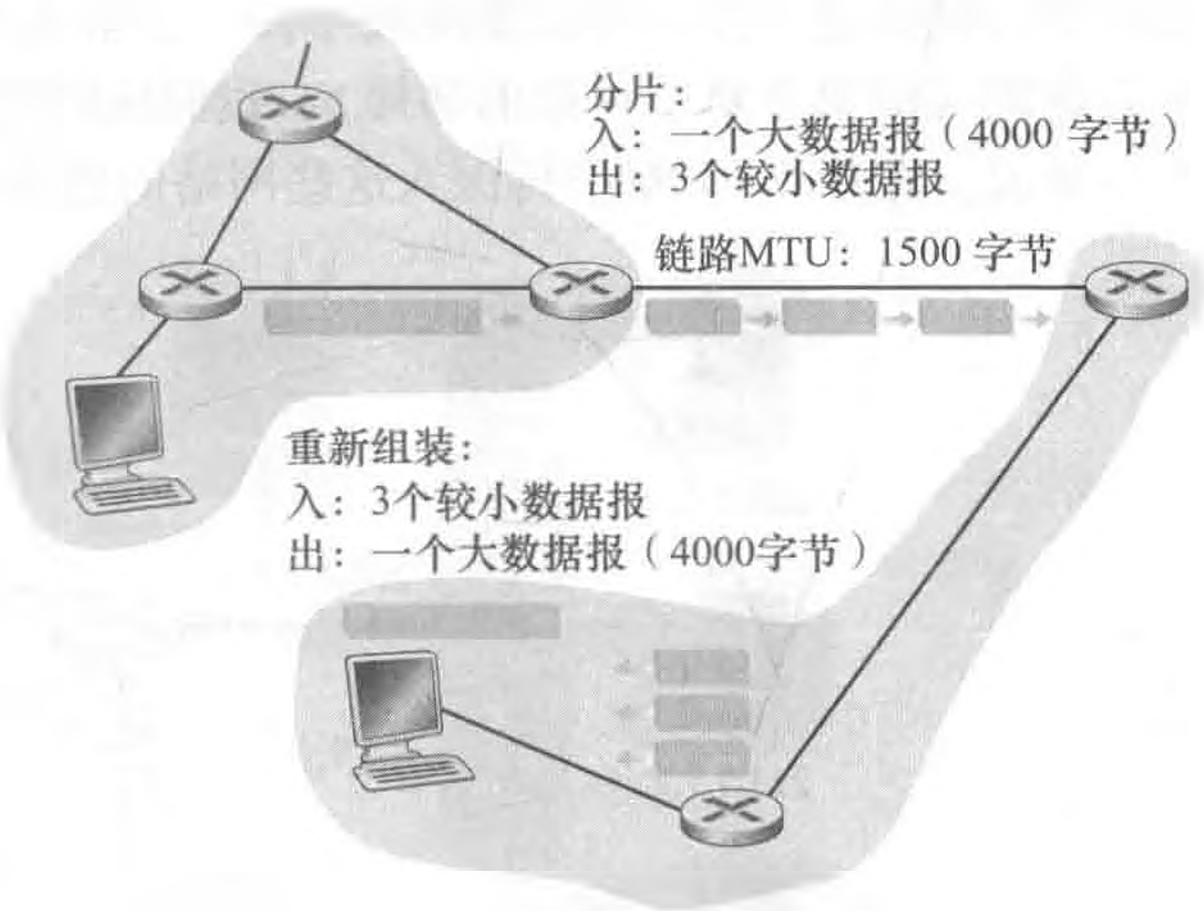

图 4-17　IP 分片与重新组装

本书的在线材料和本章后面的习题将使你能够详细探究分片。在本书的 Web 站点上，我们提供了一个 Java 小程序来产生片。提供入数据报长度、MTU 和入数据报标识，它就会自动为你产生片。参见 http://www.pearsonhighered.com/cs-resources/。

4.3.3　IPv4 编址

我们现在将注意力转向 IPv4 编址。尽管你可能认为编址是相当直接的主题，但我们希望通过本章的学习，你能认识到因特网编址不仅是一个丰富多彩、微妙和有趣的主题，而且也是一个对因特网极为重要的主题。[Stewart 1999] 的第 1 章是介绍 IPv4 编址的优秀读物。

然而，在讨论 IP 编址之前，我们需要简述一下主机与路由器连入网络的方法。一台主机通常只有一条链路连接到网络；当主机中的 IP 想发送一个数据报时，它就在该链路上发送。主机与物理链路之间的边界叫作**接口**（interface）。现在考虑一台路由器及其接口。因为路由器的任务是从链路上接收数据报并从某些其他链路转发出去，路由器必须拥有两条或更多条链路与它连接。路由器与它的任意一条链路之间的边界也叫作接口。一台路由器因此有多个接口，每个接口有其链路。因为每台主机与路由器都能发送和接收 IP 数据报，IP 要求每台主机和路由器接口拥有自己的 IP 地址。因此，从技术上讲，一个 IP 地址与一个接口相关联，而不是与包括该接口的主机或路由器相关联。

每个 IP 地址长度为 32 比特（等价为 4 字节），因此总共有 2^{32} 个（或大约 40 亿个）可能的 IP 地址。这些地址通常按所谓**点分十进制记法**（dotted-decimal notation）书写，即地址中的每个字节用它的十进制形式书写，各字节间以句点隔开。例如，考虑 IP 地址 193.32.216.9，193 是该地址的第一个 8 比特的十进制等价数，32 是该地址的第二个 8 比

特的十进制等价数，依次类推。因此，地址 193.32.216.9 的二进制记法是：

11000001 00100000 11011000 00001001

在全球因特网中的每台主机和路由器上的每个接口，都必须有一个全球唯一的 IP 地址（在 NAT 后面的接口除外，在 4.3.4 节中讨论）。然而，这些地址不能随意地自由选择。一个接口的 IP 地址的一部分需要由其连接的子网来决定。

图 4-18 提供了一个 IP 编址与接口的例子。在该图中，一台路由器（具有 3 个接口）用于互联 7 台主机。仔细观察分配给主机和路由器接口的 IP 地址，有几点需要注意。图 4-18中左上侧的 3 台主机以及它们连接的路由器接口，都有一个形如 223.1.1.xxx 的 IP 地址。这就是说，在它们的 IP 地址中，最左侧的 24 比特是相同的。这 4 个接口也通过一个并不包含路由器的网络互联起来。该网络可能由一个以太网 LAN 互联，在此情况下，这些接口将通过一台以太网交换机互联（如第 6 章中讨论的那样），或者通过一个无线接入点互联（如第 7 章中讨论的那样）。我们此时将这种无路由器连接这些主机的网络表示为一朵云，在第 6、7 章中再深入这些网络的内部。

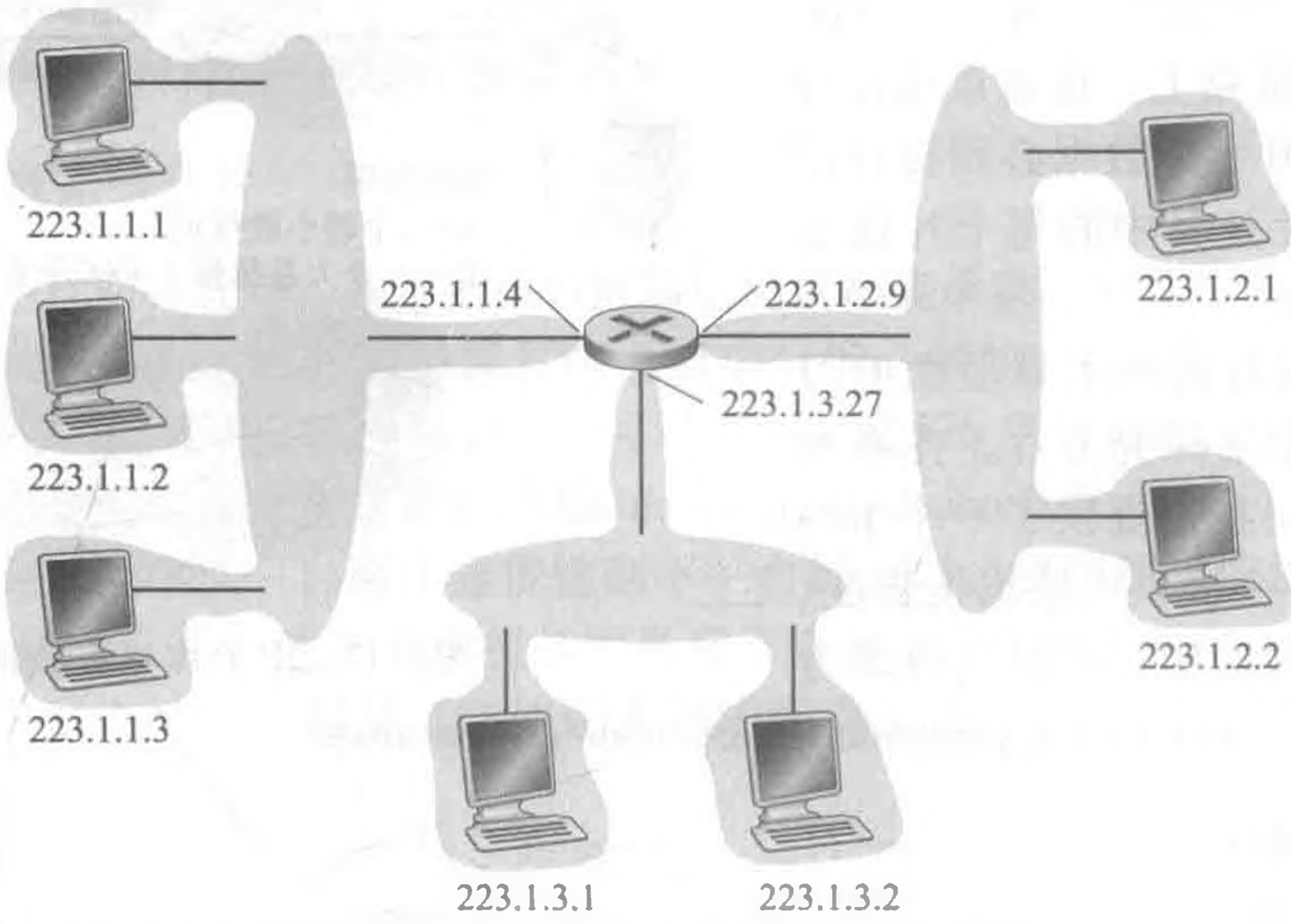

图 4-18　接口地址和子网

用 IP 的术语来说，互联这 3 个主机接口与 1 个路由器接口的网络形成一个**子网**（subnet）［RFC 950］。（在因特网文献中，子网也称为 IP 网络或直接称为网络。）IP 编址为这个子网分配一个地址 223.1.1.0/24，其中的 /24 记法，有时称为**子网掩码**（network mask），指示 32 比特中的最左侧 24 比特定义了子网地址。因此子网 223.1.1.0/24 由 3 个主机接口（223.1.1.1、223.1.1.2 和 223.1.1.3）和 1 个路由器接口（223.1.1.4）组成。任何其他要连到 223.1.1.0/24 网络的主机都要求其地址具有 223.1.1.xxx 的形式。图 4-18中显示了另外两个网络：223.1.2.0/24 网络与 223.1.3.0/24 子网。图 4-19 图示了在

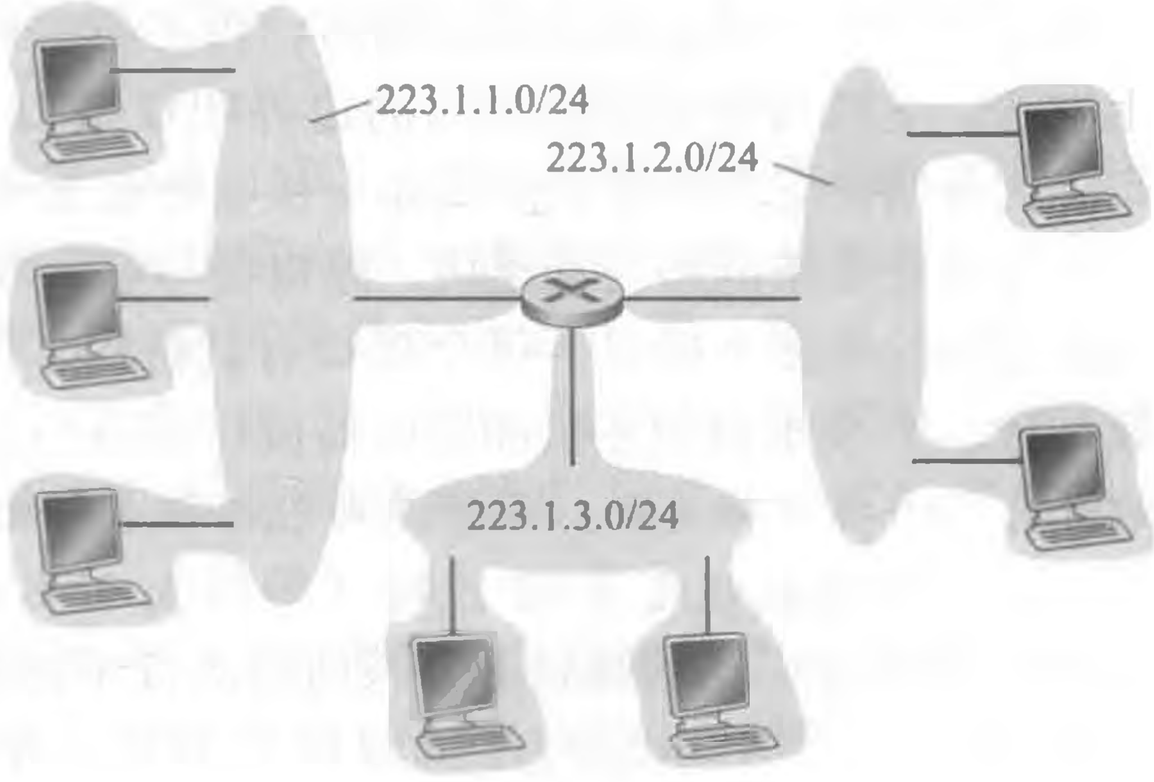

图 4-19　子网地址

图 4-18 中存在的 3 个 IP 子网。

一个子网的 IP 定义并不局限于连接多台主机到一个路由器接口的以太网段。为了搞清其中的道理，可考虑图 4-20，图中显示了 3 台通过点对点链路彼此互联的路由器。每台路由器有 3 个接口，每条点对点链路使用一个，一个用于直接将路由器连接到一对主机的广播链路。这里出现了几个子网呢？3 个子网 223.1.1.0/24、223.1.2.0/24 和 223.1.3.0/24 类似于我们在图 4-18 中遇到的子网。但注意到在本例中还有其他 3 个子网：一个子网是 223.1.9.0/24，用于连接路由器 R1 与 R2 的接口；另外一个子网是 223.1.8.0/24，用于连接路由器 R2 与 R3 的接口；第三个子网是 223.1.7.0/24，用于连接路由器 R3 与 R1 的接口。对于一个路由器和主机的通用互联系统，我们能够使用下列有效方法定义系统中的子网：

> 为了确定子网，分开主机和路由器的每个接口，产生几个隔离的网络岛，使用接口端接这些隔离的网络的端点。这些隔离的网络中的每一个都叫作一个**子网**（subnet）。

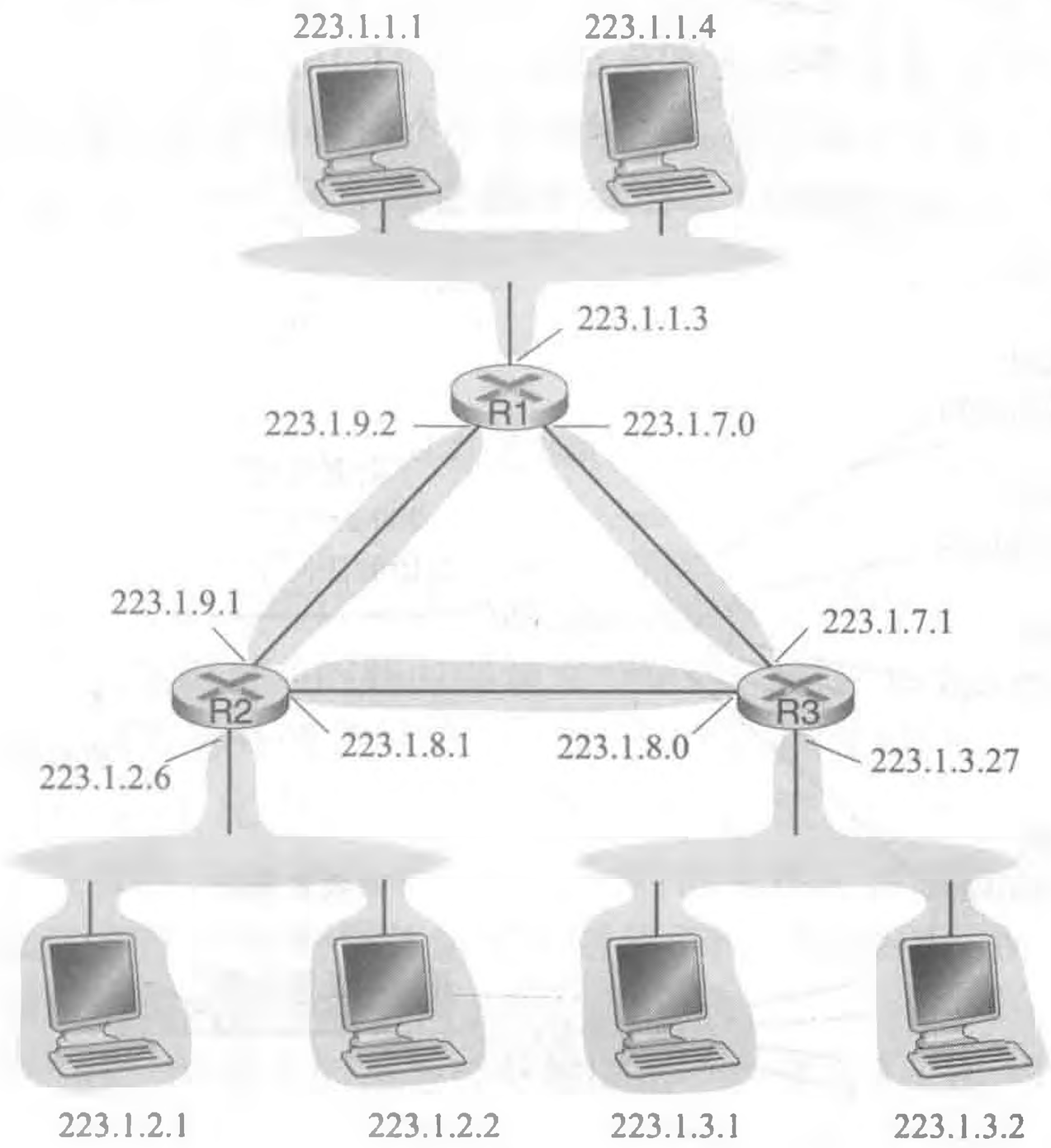

图 4-20　3 台路由器互联 6 个子网

如果我们将该过程用于图 4-20 中的互联系统上，会得到 6 个岛或子网。

从上述讨论显然可以看出，一个具有多个以太网段和点对点链路的组织（如一个公司或学术机构）将具有多个子网，在给定子网上的所有设备都具有相同的子网地址。原则上，不同的子网能够具有完全不同的子网地址。然而，在实践中，它们的子网地址经常有许多共同之处。为了理解其中的道理，我们来关注在全球因特网中是如何处理编址的。

因特网的地址分配策略被称为**无类别域间路由选择**（Classless Interdomain Routing，CIDR）［RFC 4632］。CIDR 将子网寻址的概念一般化了。当使用子网寻址时，32 比特的 IP 地址被划分为两部分，并且也具有点分十进制数形式 *a.b.c.d/x*，其中 *x* 指示了地址的第一部分中的比特数。

形式为 *a. b. c. d/x* 的地址的 *x* 最高比特构成了 IP 地址的网络部分，并且经常被称为该地址的**前缀**（prefix）（或网络前缀）。一个组织通常被分配一块连续的地址，即具有相同前缀的一段地址（参见“实践原则”）。在这种情况下，该组织内部的设备的 IP 地址将共享共同的前缀。当我们在 5.4 节中论及因特网的 BGP 路由选择协议时，将看到该组织网络外部的路由器仅考虑前面的前缀比特 *x*。这就是说，当该组织外部的一台路由器转发一个数据报，且该数据报的目的地址位于该组织的内部时，仅需要考虑该地址的前面 *x* 比特。这相当大地减少了在这些路由器中转发表的长度，因为形式为 *a. b. c. d/x* 的单一表项足以将数据报转发到该组织内的任何目的地。

实践原则

这是一个 ISP 将 8 个组织连接到因特网的例子，它也很好地说明了仔细分配 CIDR 化的地址有利于路由选择的道理。如图 4-21 所示，假设该 ISP（我们称之为 Fly-By-Night-ISP）向外界通告，它应该发送所有地址的前 20 比特与 200.23.16.0/20 相符的数据报。外界的其他部分不需要知道在地址块 200.23.16.0/20 内实际上还存在 8 个其他组织，其中每个组织有自己的子网。这种使用单个网络前缀通告多个网络的能力通常称为**地址聚合**（address aggregation），也称为**路由聚合**（route aggregation）或**路由摘要**（route summarization）。

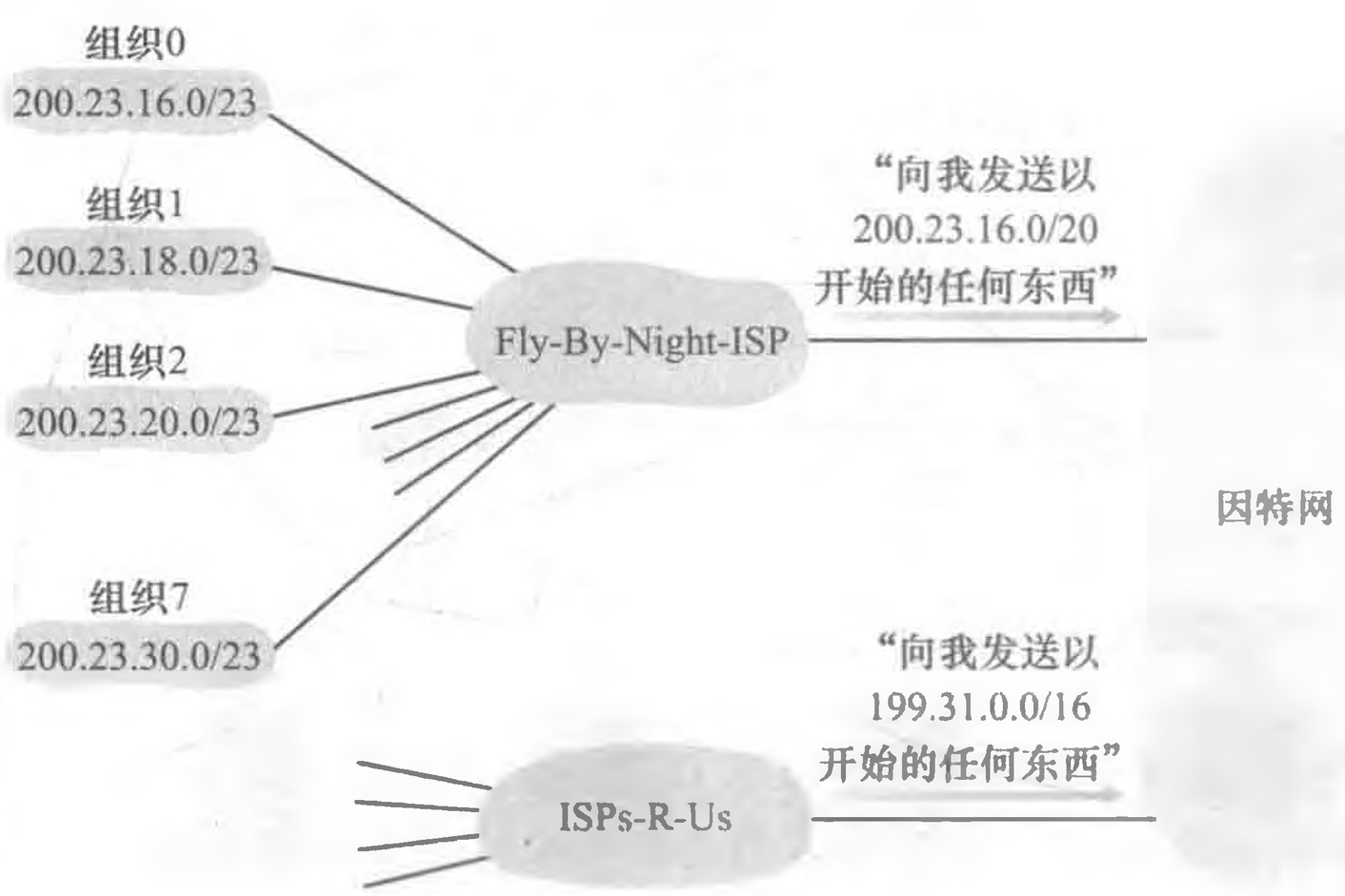

图 4-21　层次编址与路由聚合

当地址按块分给 ISP，然后又由 ISP 分给客户组织时，地址聚合工作极为有效。但是当地址不是按这样的层次方式分配时，会出现什么情况呢？例如，如果 Fly-By-Night-ISP 获取了 ISPs-R-Us，然后让组织 1 通过它辅助的 ISPs-R-Us 与因特网相连，将会发生什么情况呢？如图 4-21 中所示，该辅助的 ISPs-R-Us 拥有地址块 199.31.0.0/16，但很遗憾的是组织 1 的 IP 地址在该地址块之外。这里可以采取什么措施呢？组织 1 无疑可以将其所有的路由器和主机重新编号，使得地址在 ISPs-R-Us 的地址块内。但这是一种代价很高的方案，而且组织 1 将来也许还会从 ISPs-R-Us 更换到另一个 ISP。采用的典型方案是，组织 1 保持其 IP 地址在 200.23.18.0/23 内。在这种情况下，如图 4-22 所示，

Fly-By-Night-ISP 继续通告地址块 200.23.16.0/20，并且 ISPs-R-Us 也继续通告地址块 199.31.0.0/16。然而，ISPs-R-Us 现在还要通告组织 1 的地址块 200.23.18.0/23。当在更大的因特网上的其他路由器看见地址块 200.23.16.0/20（来自 Fly-By-Night-ISP）和 200.23.18.0/23（来自 ISPs-R-Us），并且想路由选择到地址块 200.23.18.0/23 内的一个地址时，它们将使用**最长前缀匹配**（参见 4.2.1 节），并朝着 ISPs-R-Us 路由，因为它通告了与目的地址相匹配的最长（最具体）的地址前缀。

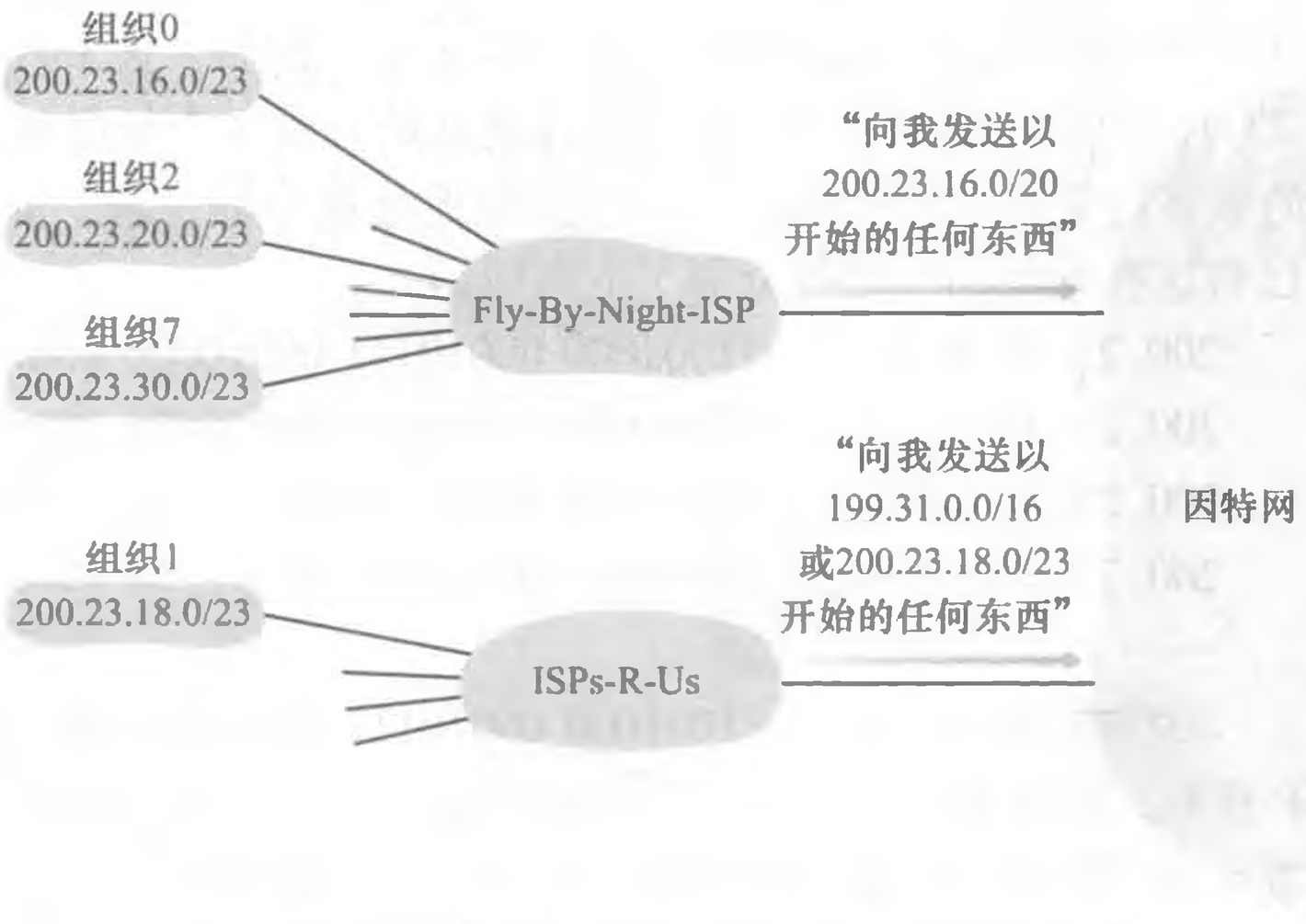

图 4-22　ISPs-R-Us 具有到组织 1 的一条更具体的路由

一个地址的剩余 32 − x 比特可认为是用于区分该组织内部设备的，其中的所有设备具有相同的网络前缀。当该组织内部的路由器转发分组时，才会考虑这些比特。这些较低阶比特可能（或可能不）具有另外的子网结构，如前面所讨论的那样。例如，假设某 CIDR 化的地址 *a. b. c. d*/21 的前 21 比特定义了该组织的网络前缀，它对该组织中所有主机的 IP 地址来说是共同的。其余的 11 比特标识了该组织内的主机。该组织的内部结构可以采用这样的方式，使用最右边的 11 比特在该组织中划分子网，就像前面所讨论的那样。例如，*a. b. c. d*/24 可能表示该组织内的特定子网。

在 CIDR 被采用之前，IP 地址的网络部分被限制为长度为 8、16 或 24 比特，这是一种称为**分类编址**（classful addressing）的编址方案，这是因为具有 8、16 和 24 比特子网地址的子网分别被称为 A、B 和 C 类网络。一个 IP 地址的网络部分正好为 1、2 或 3 字节的要求，已经在支持数量迅速增加的具有小规模或中等规模子网的组织方面出现了问题。一个 C 类（/24）子网仅能容纳多达 $2^8 - 2 = 254$（$2^8 = 256$，其中的两个地址预留用于特殊用途）台主机，这对于许多组织来说太小了。然而一个 B 类（/16）子网可支持多达 65 534 台主机，又太大了。在分类编址方法下，比方说一个有 2000 台主机的组织通常被分给一个 B 类（/16）地址。这就导致了 B 类地址空间的迅速损耗以及所分配的地址空间的利用率低下。例如，为具有 2000 台主机的组织分配一个 B 类地址，就具有足以支持多达 65 534 个接口的地址空间，剩下的超过 63 000 个地址却不能被其他组织使用。

如果还不提及另一种类型的 IP 地址，即 IP 广播地址 255. 255. 255. 255，那将是我们的疏漏。当一台主机发出一个目的地址为 255. 255. 255. 255 的数据报时，该报文会交付给

同一个网络中的所有主机。路由器也会有选择地向邻近的子网转发该报文（虽然它们通常不这样做）。

现在我们已经详细地学习了 IP 编址，需要知道主机或子网最初是如何得到它们的地址的。我们先看一个组织是如何为其设备得到一个地址块的，然后再看一个设备（如一台主机）是如何从某组织的地址块中分配到一个地址的。

1. 获取一块地址

为了获取一块 IP 地址用于一个组织的子网内，某网络管理员也许首先会与他的 ISP 联系，该 ISP 可能会从已分给它的更大地址块中提供一些地址。例如，该 ISP 也许自己已被分配了地址块 200.23.16.0/20。该 ISP 可以依次将该地址块分成 8 个长度相等的连续地址块，为本 ISP 支持的最多达 8 个组织中的一个分配这些地址块中的一块，如下所示。（为了便于查看，我们已将这些地址的网络部分加了下划线。）

ISP 的地址块	200.23.16.0/20	11001000 00010111 00010000 00000000
组织 0	200.23.16.0/23	11001000 00010111 00010000 00000000
组织 1	200.23.18.0/23	11001000 00010111 00010010 00000000
组织 2	200.23.20.0/23	11001000 00010111 00010100 00000000
……	……	……
组织 7	200.23.30.0/23	11001000 00010111 00011110 00000000

尽管从一个 ISP 获取一组地址是一种得到一块地址的方法，但这不是唯一的方法。显然，必须还有一种方法供 ISP 本身得到一块地址。是否有一个全球性的权威机构，它具有管理 IP 地址空间并向各 ISP 和其他组织分配地址块的最终责任呢？的确有一个！IP 地址由因特网名字和编号分配机构（Internet Corporation for Assigned Names and Numbers，ICANN）［ICANN 2016］管理，管理规则基于［RFC 7020］。非营利的 ICANN 组织［NTIA 1998］的作用不仅是分配 IP 地址，还管理 DNS 根服务器。它还有一项容易引起争论的工作，即分配域名与解决域名纷争。ICANN 向区域性因特网注册机构（如 ARIN、RIPE、APNIC 和 LACNIC）分配地址，这些机构一起形成了 ICANN 的地址支持组织［ASO-ICANN 2016］，处理本区域内的地址分配/管理。

2. 获取主机地址：动态主机配置协议

某组织一旦获得了一块地址，它就可为本组织内的主机与路由器接口逐个分配 IP 地址。系统管理员通常手工配置路由器中的 IP 地址（常常在远程通过网络管理工具进行配置）。主机地址也能手动配置，但是这项任务目前更多的是使用**动态主机配置协议**（Dynamic Host Configuration，DHCP）［RFC 2131］来完成。DHCP 允许主机自动获取（被分配）一个 IP 地址。网络管理员能够配置 DHCP，以使某给定主机每次与网络连接时能得到一个相同的 IP 地址，或者某主机将被分配一个**临时的 IP 地址**（temporary IP address），每次与网络连接时该地址也许是不同的。除了主机 IP 地址分配外，DHCP 还允许一台主机得知其他信息，例如它的子网掩码、它的第一跳路由器地址（常称为默认网关）与它的本地 DNS 服务器的地址。

由于 DHCP 具有将主机连接进一个网络的网络相关方面的自动能力，故它又常被称为**即插即用协议**（plug-and-play protocol）或**零配置**（zeroconf）**协议**。这种能力对于网络管理员来说非常有吸引力，否则他将不得不手工执行这些任务！DHCP 还广泛地用于住宅因特网接入网、企业网与无线局域网中，其中的主机频繁地加入和离开网络。例如，考虑一个学