a. 对于一个给定的 TCP 连接，假定 4 个确认报文相继到达，带有 4 个对应的 RTT 值：$SampleRT_{T4}$、$SampleRT_{T3}$、$SampleRT_{T2}$和 $SampleRT_{T1}$。根据这 4 个样本 RTT 表示 EstimatedRTT。

b. 将你得到的公式一般化到 n 个 RTT 样本的情况。

c. 对于在（b）中得到的公式，令 n 趋于无穷。试说明为什么这个平均过程被称为指数移动平均。

P33. 在 3.5.3 节中，我们讨论了 TCP 的往返时间的估计。TCP 避免测量重传报文段的 SampleRTT，对此你有何看法？

P34. 3.5.4 节中的变量 SendBase 和 3.5.5 节中的变量 LastByteRcvd 之间有什么关系？

P35. 3.5.5 节中的变量 LastByteRcvd 和 3.5.4 节中的变量 y 之间有什么关系？

P36. 在 3.5.4 节中，我们看到 TCP 直到收到 3 个冗余 ACK 才执行快速重传。你对 TCP 设计者没有选择在收到对报文段的第一个冗余 ACK 后就快速重传有何看法？

P37. 比较 GBN、SR 和 TCP（无延时的 ACK）。假设对所有 3 个协议的超时值足够长，使得 5 个连续的数据报文段及其对应的 ACK 能够分别由接收主机（主机 B）和发送主机（主机 A）收到（如果在信道中无丢失）。假设主机 A 向主机 B 发送 5 个数据报文段，并且第二个报文段（从 A 发送）丢失。最后，所有 5 个数据报文段已经被主机 B 正确接收。

a. 主机 A 总共发送了多少报文段和主机 B 总共发送了多少 ACK？它们的序号是什么？对所有 3 个协议回答这个问题。

b. 如果对所有 3 个协议超时值比 5RTT 长得多，则哪个协议在最短的时间间隔中成功地交付所有 5 个数据报文段？

P38. 在图 3-52 中的 TCP 描述中，阈值 ssthresh 的值在几个地方被设置为 ssthresh = cwnd/2，并且当出现一个丢包事件时，ssthresh 的值被设置为窗口长度的一半。当出现丢包事件时，发送方发送的速率，每个 RTT 必须大约等于 cwnd 报文段吗？解释你的答案。如果你的回答是没有，你能建议一种不同的方式，进行 ssthresh 设置吗？

P39. 考虑图 3-46b。如果 λ'_{in}增加超过了 $R/2$，λ_{out}能够增加超过 $R/3$ 吗？试解释之。现在考虑图 3-46c。假定一个分组从路由器到接收方平均转发两次的话，如果 λ'_{in}增加超过 $R/2$，λ_{out}能够增加超过 $R/4$ 吗？试解释之。

P40. 考虑图 3-58。假设 TCP Reno 是一个经历如上所示行为的协议，回答下列问题。在各种情况中，简要地论证你的回答。

a. 指出 TCP 慢启动运行时的时间间隔。

b. 指出 TCP 拥塞避免运行时的时间间隔。

c. 在第 16 个传输轮回之后，报文段的丢失是根据 3 个冗余 ACK 还是根据超时检测出来的？

d. 在第 22 个传输轮回之后，报文段的丢失是根据 3 个冗余 ACK 还是根据超时检测出来的？

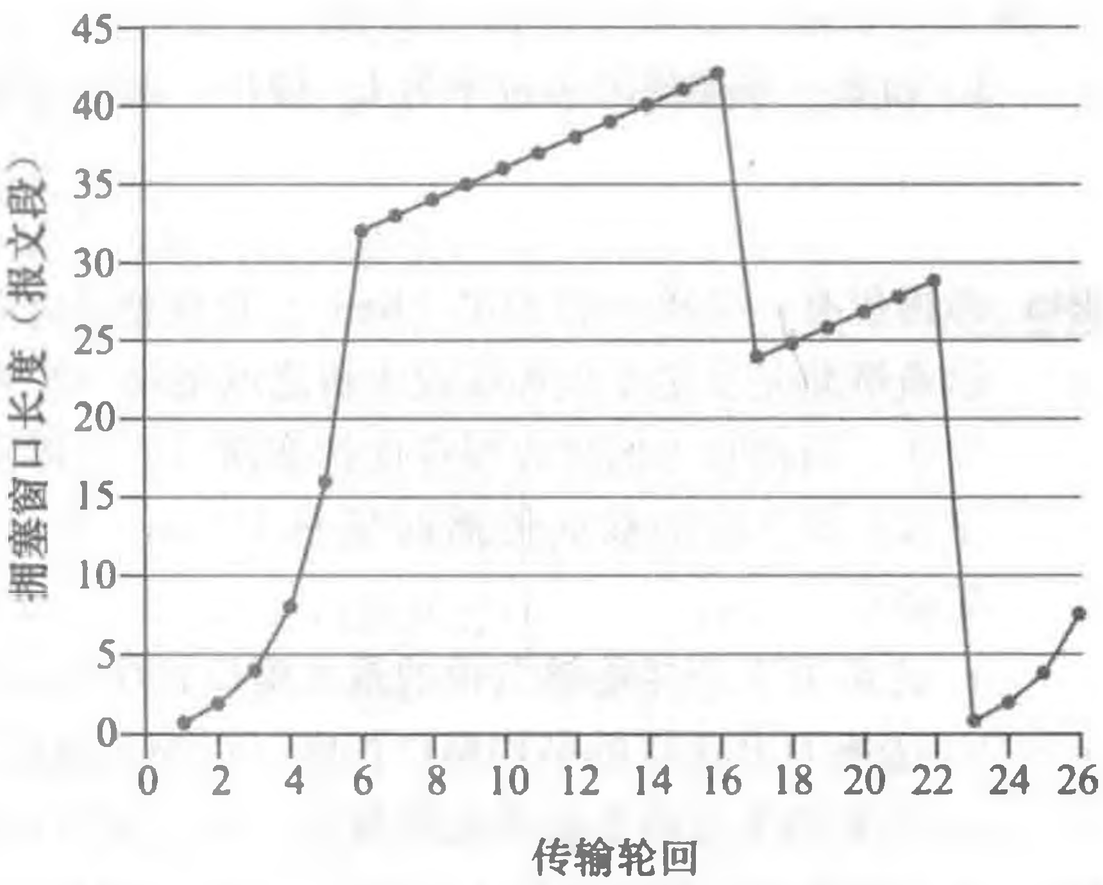

图 3-58 TCP 窗口长度作为时间的函数

e. 在第 1 个传输轮回里，ssthresh 的初始值设置为多少？

f. 在第 18 个传输轮回里，ssthresh 的值设置为多少？

g. 在第 24 个传输轮回里，ssthresh 的值设置为多少？

h. 在哪个传输轮回内发送第 70 个报文段？

i. 假定在第 26 个传输轮回后，通过收到 3 个冗余 ACK 检测出有分组丢失，拥塞的窗口长度和 ssthresh 的值应当是多少？

j. 假定使用 TCP Tahoe（而不是 TCP Reno），并假定在第 16 个传输轮回收到 3 个冗余 ACK。在第 19 个传输轮回，ssthresh 和拥塞窗口长度是什么？

k. 再次假设使用 TCP Tahoe，在第 22 个传输轮回有一个超时事件。从第 17 个传输轮回到第 22 个传输轮回（包括这两个传输轮回），一共发送了多少分组？

P41. 参考图 3-55，该图描述了 TCP 的 AIMD 算法的收敛特性。假设 TCP 不采用乘性减，而是采用按某一常量减小窗口。所得的 AIAD 算法将收敛于一种平等共享算法吗？使用类似于图 3-55 中的图来证实你的结论。

P42. 在 3.5.4 节中，我们讨论了在发生超时事件后将超时间隔加倍。为什么除了这种加倍超时间隔机制外，TCP 还需要基于窗口的拥塞控制机制（如在 3.7 节中学习的那种机制）呢？

P43. 主机 A 通过一条 TCP 连接向主机 B 发送一个很大的文件。在这条连接上，不会出现任何分组丢失和定时器超时。主机 A 与因特网连接链路的传输速率表示为 R bps。假设主机 A 上的进程能够以 S bps 的速率向 TCP 套接字发送数据，其中 $S = 10 \times R$。进一步假设 TCP 的接收缓存足够大，能够容纳整个文件，并且发送缓存只能容纳这个文件的百分之一。如何防止主机 A 上的进程连续地向 TCP 套接字以速率 S bps 传送数据呢？还是用 TCP 流量控制呢？还是用 TCP 拥塞控制？或者用其他措施？阐述其理由。

P44. 考虑从一台主机经一条没有丢包的 TCP 连接向另一台主机发送一个大文件。

a. 假定 TCP 使用不具有慢启动的 AIMD 进行拥塞控制。假设每当收到一批 ACK 时，cwnd 增加 1 个 MSS，并且假设往返时间大约恒定，cwnd 从 6MSS 增加到 12MSS 要花费多长时间（假设没有丢包事件）？

b. 对于该连接，到时间 =6RTT，其平均吞吐量是多少（根据 MSS 和 RTT）？

P45. 回想 TCP 吞吐量的宏观描述。在连接速率从 $W/(2 \times \text{RTT})$ 变化到 W/RTT 的周期内，只丢失了一个分组（在该周期的结束）。

a. 证明其丢包率（分组丢失的比率）等于：

$$L = \text{丢包率} = \frac{1}{\frac{3}{8}W^2 + \frac{3}{4}W}$$

b. 如果一条连接的丢包率为 L，使用上面的结果，则它的平均速率近似由下式给出：

$$\text{平均速率} \approx \frac{1.22 * \text{MSS}}{\text{RTT}\sqrt{L}}$$

P46. 考虑仅有一条单一的 TCP（Reno）连接使用一条 10Mbps 链路，且该链路没有缓存任何数据。假设这条链路是发送主机和接收主机之间的唯一拥塞链路。假定某 TCP 发送方向接收方有一个大文件要发送，而接收方的接收缓存比拥塞窗口要大得多。我们也做下列假设：每个 TCP 报文段长度为 1500 字节；该连接的双向传播时延是 150ms；并且该 TCP 连接总是处于拥塞避免阶段，即忽略了慢启动。

a. 这条 TCP 连接能够取得的最大窗口长度（以报文段计）是多少？

b. 这条 TCP 连接的平均窗口长度（以报文段计）和平均吞吐量（以 bps 计）是多少？

c. 这条 TCP 连接在从丢包恢复后，再次到达其最大窗口要经历多长时间？

P47. 考虑在前面习题中所描述的场景。假设 10Mbps 链路能够缓存有限个报文段。试论证为了使该链路总是忙于发送数据，我们将要选择缓存长度，使得其至少为发送方和接收方之间链路速率 C 与双向传播时延之积。

P48. 重复习题 46，但用一条 10Gbps 链路代替 10Mbps 链路。注意到在对 c 部分的答案中，应当认识到在从丢包恢复后，拥塞窗口长度到达最大窗口长度将需要很长时间。给出解决该问题的基本思路。

P49. 令 T（用 RTT 度量）表示一条 TCP 连接将拥塞窗口从 $W/2$ 增加到 W 所需的时间间隔，其中 W 是最大的拥塞窗口长度。论证 T 是 TCP 平均吞吐量的函数。

P50. 考虑一种简化的 TCP 的 AIMD 算法，其中拥塞窗口长度用报文段的数量来度量，而不是用字节度量。在加性增中，每个 RTT 拥塞窗口长度增加一个报文段。在乘性减中，拥塞窗口长度减小一半（如果结果不是一个整数，向下取整到最近的整数）。假设两条 TCP 连接 C1 和 C2，它们共享一条速

率为每秒 30 个报文段的单一拥塞链路。假设 C1 和 C2 均处于拥塞避免阶段。连接 C1 的 RTT 是 50ms，连接 C2 的 RTT 是 100ms。假设当链路中的数据速率超过了链路的速率时，所有 TCP 连接经受数据报文段丢失。

a. 如果在时刻 t_0，C1 和 C2 具有 10 个报文段的拥塞窗口，在 1000ms 后它们的拥塞窗口为多长？

b. 经长时间运行，这两条连接将取得共享该拥塞链路的相同的带宽吗？

P51. 考虑在前面习题中描述的网络。现在假设两条 TCP 连接 C1 和 C2，它们具有相同的 100ms RTT。假设在时刻 t_0，C1 的拥塞窗口长度为 15 个报文段，而 C2 的拥塞窗口长度是 10 个报文段。

a. 在 2200ms 后，它们的拥塞窗口长度为多长？

b. 经长时间运行，这两条连接将取得共享该拥塞链路的相同的带宽吗？

c. 如果这两条连接在相同时间达到它们的最大窗口长度，并在相同时间达到它们的最小窗口长度，我们说这两条连接是同步的。经长时间运行，这两条连接将最终变得同步吗？如果是，它们的最大窗口长度是多少？

d. 这种同步将有助于改善共享链路的利用率吗？为什么？给出打破这种同步的某种思路。

P52. 考虑修改 TCP 的拥塞控制算法。不使用加性增，使用乘性增。无论何时某 TCP 收到一个合法的 ACK，就将其窗口长度增加一个小正数 a（$0 < a < 1$）。求出丢包率 L 和最大拥塞窗口 W 之间的函数关系。论证：对于这种修正的 TCP，无论 TCP 的平均吞吐量如何，一条 TCP 连接将其拥塞窗口长度从 $W/2$ 增加到 W，总是需要相同的时间。

P53. 在 3.7 节对 TCP 未来的讨论中，我们注意到为了取得 10Gbps 的吞吐量，TCP 仅能容忍 2×10^{-10} 的报文段丢失率（或等价为每 5 000 000 000 个报文段有一个丢包事件）。给出针对 3.7 节中给定的 RTT 和 MSS 值的对 2×10^{-10} 值的推导。如果 TCP 需要支持一条 100Gbps 的连接，所能容忍的丢包率是多少？

P54. 在 3.7 节中对 TCP 拥塞控制的讨论中，我们隐含地假定 TCP 发送方总是有数据要发送。现在考虑下列情况，某 TCP 发送方发送大量数据，然后在 t_1 时刻变得空闲（因为它没有更多的数据要发送）。TCP 在相对长的时间内保持空闲，然后在 t_2 时刻要发送更多的数据。当 TCP 在 t_2 开始发送数据时，让它使用在 t_1 时刻的 cwnd 和 ssthresh 值，将有什么样的优点和缺点？你建议使用什么样的方法？为什么？

P55. 在这个习题中我们研究是否 UDP 或 TCP 提供了某种程度的端点鉴别。

a. 考虑一台服务器接收到在一个 UDP 分组中的请求并对该请求进行响应（例如，如由 DNS 服务器所做的那样）。如果一个具有 IP 地址 X 的客户用地址 Y 进行哄骗的话，服务器将向何处发送它的响应？

b. 假定一台服务器接收到具有 IP 源地址 Y 的一个 SYN，在用 SYNACK 响应之后，接收一个具有 IP 源地址 Y 和正确确认号的 ACK。假设该服务器选择了一个随机初始序号并且没有“中间人”，该服务器能够确定该客户的确位于 Y 吗？（并且不在某个其他哄骗为 Y 的地址 X。）

P56. 在这个习题中，我们考虑由 TCP 慢启动阶段引入的时延。考虑一个客户和一个 Web 服务器直接连接到速率 R 的一条链路。假定该客户要取回一个对象，其长度正好等于 $15S$，其中 S 是最大段长度（MSS）。客户和服务器之间的往返时间表示为 RTT（假设为常数）。忽略协议首部，确定在下列情况下取回该对象的时间（包括 TCP 连接创建）：

a. $4S/R > S/R + \text{RTT} > 2S/R$

b. $S/R + \text{RTT} > 4S/R$

c. $S/R > \text{RTT}$

编程作业

实现一个可靠运输协议

在这个编程作业实验中，你将要编写发送和接收运输层的代码，以实现一个简单的可靠数据运输协

议。这个实验有两个版本，即比特交替协议版本和 GBN 版本。这个实验应当是有趣的，因为你的实现将与实际情况下所要求的差异很小。

因为可能没有你能够修改其操作系统的独立机器，你的代码将不得不在模拟的硬件/软件环境中执行。然而，为你提供例程的编程接口（即从上层和下层调用你的实体的代码），非常类似于在实际 UNIX 环境中做那些事情的接口。（实际上，在本编程作业中描述的软件接口比起许多教科书中描述的无限循环的发送方和接收方要真实得多。）停止和启动定时器也是模拟的，定时器中断将激活你的定时器处理例程。

这个完整的实验作业以及你所需要的代码，可从本书的 Web 网站上获得：http://www.pearsonhighered.com/cs-resources。

Wireshark 实验：探究 TCP

在这个实验中，你将使用 Web 浏览器访问来自某 Web 服务器的一个文件。如同在前面的 Wireshark 实验中一样，你将使用 Wireshark 来俘获到达你计算机的分组。与前面实验不同的是，你也能够从该 Web 服务器下载一个 Wireshark 可读的分组踪迹，记载你从服务器下载文件的过程。在这个服务器踪迹文件里，你将发现自己访问该 Web 服务器所产生的分组。你将分析客户端和服务器端踪迹文件，以探究 TCP 的方方面面。特别是你将评估在你的计算机与该 Web 服务器之间 TCP 连接的性能。你将跟踪 TCP 窗口行为、推断分组丢失、重传、流控和拥塞控制行为并估计往返时间。

与所有的 Wireshark 实验一样，该实验的全面描述能够在本书 Web 站点 http://www.pearsonhighered.com/cs-resources 上找到。

Wireshark 实验：探究 UDP

在这个简短实验中，你将进行分组俘获并分析那些使用 UDP 的你喜爱的应用程序（例如，DNS 或如 Skype 这样的多媒体应用）。如我们在 3.3 节中所学的那样，UDP 是一种简单的、不提供不必要服务的运输协议。在这个实验中，你将研究在 UDP 报文段中的各首部字段以及检验和计算。

与所有的 Wireshark 实验一样，该实验的全面描述能够在本书 Web 站点 http://www.pearsonhighered.com/cs-resources 上找到。

人物专访

Van Jacobson 就职于谷歌公司，以前是 PARC 的高级研究员。在此之前，他是分组设计（Packet Design）组织的共同创建者和首席科学家。再往前，他是思科公司的首席科学家。在加入思科之前，他是劳伦兹伯克利国家实验室的网络研究组的负责人，并在加州大学伯克利分校和斯坦福大学任教。Van 于 2001 年因其在通信网络领域的贡献而获得 ACM SIGCOMM 终身成就奖，于 2002 年因其“对网络拥塞的理解和成功研制用于因特网的拥塞控制机制”而获得 IEEE Kobayashi 奖。他于 2004 年当选为美国国家工程院院士。

Van Jacobson

- 请描述您职业生涯中做过的一两个最令人激动的项目。最大的挑战是什么？

学校教会我们许多寻找答案的方式。在每个我致力于的感兴趣的问题中，艰巨的任务是找到正确的问题。当 Mike Karels 和我开始关注 TCP 拥塞时，我们花费数月凝视协议和分组踪迹，询问“为什么它会失效？”。有一天在 Mike 的办公室，我们中的一个说：“我无法弄明白它失效的原因是因为我不理解它究竟如何开始运转的。”这导致提出了一个正确问题，它迫使我们弄明白使 TCP 运转的“ack 计时”。从那以后，其他东西就容易了。

- 从更为一般的意义上讲，您认为网络和因特网未来将往何处发展？

对于大多数人来说，Web 是因特网。对此，网络奇才们将会善意地窃笑，因为我们知道 Web 是一个

运行在因特网上的应用程序，但要是以上说法正确又该如何呢？因特网使得主机对之间能够进行交谈。Web 用于分布信息的生产和消耗。“信息传播”是一种非常一般意义上的通信，而“成对交谈”只是其中一个极小的子集。我们需要向更大的范围进发。今天的网络以点到点连线的方式处理广播媒体（无线电、PON 等）。那是极为低效的。经过指头敲击或智能手机，遍及全世界的每秒兆兆（10^{12}）比特的数据正在交换，但我们不知道如何将其作为“网络”处理。ISP 正在忙于建立缓存和 CDN，以可扩展地分发视频和音频。缓存是该解决方案的必要部分，但今天的网络缺乏这个部分。从信息论、排队论或流量理论直到因特网协议规范，都告诉我们如何建造和部署它。我认为并希望在未来几年中，网络将演化为包含多得多的通信愿景，支撑 Web 的运行。

- 是谁激发了您的职业灵感？

当我还在研究生院时，Richard Feynman 访问了学校并做了学术报告。他讲到了一些量子理论知识，使我整学期都在努力理解该理论，他的解释非常简单和明白易懂，使得那些对我而言难以理解的东西变得显而易见和不可避免。领会和表达复杂世界背后的简单性的能力是给我的罕见和绝妙的礼物。

- 您对进入计算机科学和网络领域的学生有什么忠告吗？

网络是奇妙的领域，计算机和网络对社会的影响，也许比自有文字记载以来的任何发明都大。网络本质上是有关连接的东西，研究它有助于你进行智能连接：蚁群搜索和蜜蜂舞蹈显示了协议设计好于 RFC，流量拥挤或人们离开挤满人的体育馆是拥塞的要素，在感恩节暴风雪后寻找航班返回学校的学生们是动态路由选择的核心。如果你对许多东西感兴趣，并且要对此干点事，很难想象还有什么比网络更好的领域了。

第4章

Computer Networking: A Top-Down Approach, Seventh Edition

网络层：数据平面

在前一章中我们学习了运输层依赖于网络层的主机到主机的通信服务，提供各种形式的进程到进程的通信。我们也学习了运输层工作时不具备任何有关网络层实际实现这种服务的知识。因此也许你现在想知道，这种主机到主机通信服务的真实情况是什么？是什么使得它工作起来的呢？

在本章和下一章中，我们将学习网络层实际是怎样实现主机到主机的通信服务的。我们将看到与运输层和应用层不同的是，在网络中的每一台主机和路由器中都有一个网络层部分。正因如此，网络层协议是协议栈中最具挑战性（因而也是最有趣）的部分。

网络层在协议栈中毋庸置疑是最复杂的层次，因此我们将在这里用大量篇幅来讨论。的确因为涉及的内容太多，我们要用两章的篇幅来讨论网络层。我们将看到网络层能够被分解为两个相互作用的部分，即**数据平面**和**控制平面**。在第4章，我们将首先学习网络层的数据平面功能，即网络层中每台路由器的功能，该数据平面功能决定到达路由器输入链路之一的数据报（即网络层的分组）如何转发到该路由器的输出链路之一。我们将涉及传统的IP转发（其中转发基于数据报的目的地址）和通用的转发（其中可以使用数据报首部中的几个不同域的值执行转发和其他功能）。我们将详细地学习IPv4和IPv6协议及其寻址。在第5章，我们将涉及网络层的控制平面功能，即网络范围的逻辑，该控制平面功能控制数据报沿着从源主机到目的主机的端到端路径中路由器之间的路由方式。我们将学习路由选择算法，以及广泛用于今天因特网中的诸如OSPF和BGP等路由选择协议。传统上，这些控制平面路由选择协议和数据平面转发功能已被实现成一个整体，位于一台路由器中。软件定义网络（Software-Defined Networking，SDN）通过将这些控制平面功能作为一种单独服务，明确地分离数据平面和控制平面，控制平面功能通常置于一台远程“控制器”中。我们将在第5章涉及SDN控制器。

网络层中数据平面和控制平面之间的功能区别很重要，当你学习网络层时，心中要记住这个区别。它将有助于你构思网络层，并且反映计算机网络中网络层角色的现代观点。

4.1 网络层概述

图4-1显示了一个简单网络，其中有H1和H2两台主机，在H1与H2之间的路径上有几台路由器。假设H1正在向H2发送信息，考虑这些主机与中间路由器的网络层所起的作用。H1中的网络层取得来自于H1运输层的报文段，将每个报文段封装成一个数据报，然后向相邻路由器R1发送该数据报。在接收方主机H2，网络层接收来自相邻路由器R2的数据报，提取出运输层报文段，并将其向上交付给H2的运输层。每台路由器的数据平面的主要作用是从其输入链路向其输出链路转发数据报；控制平面的主要作用是协调这些本地的每路由器转发动作，使得数据报沿着源和目的地主机之间的路由器路径最终进行端到端传送。注意到图4-1中所示路由器具有截断的协议栈，即没有网络层以上的部分，

因为路由器不运行我们已在第 2、3 章学习过的应用层和运输层协议。

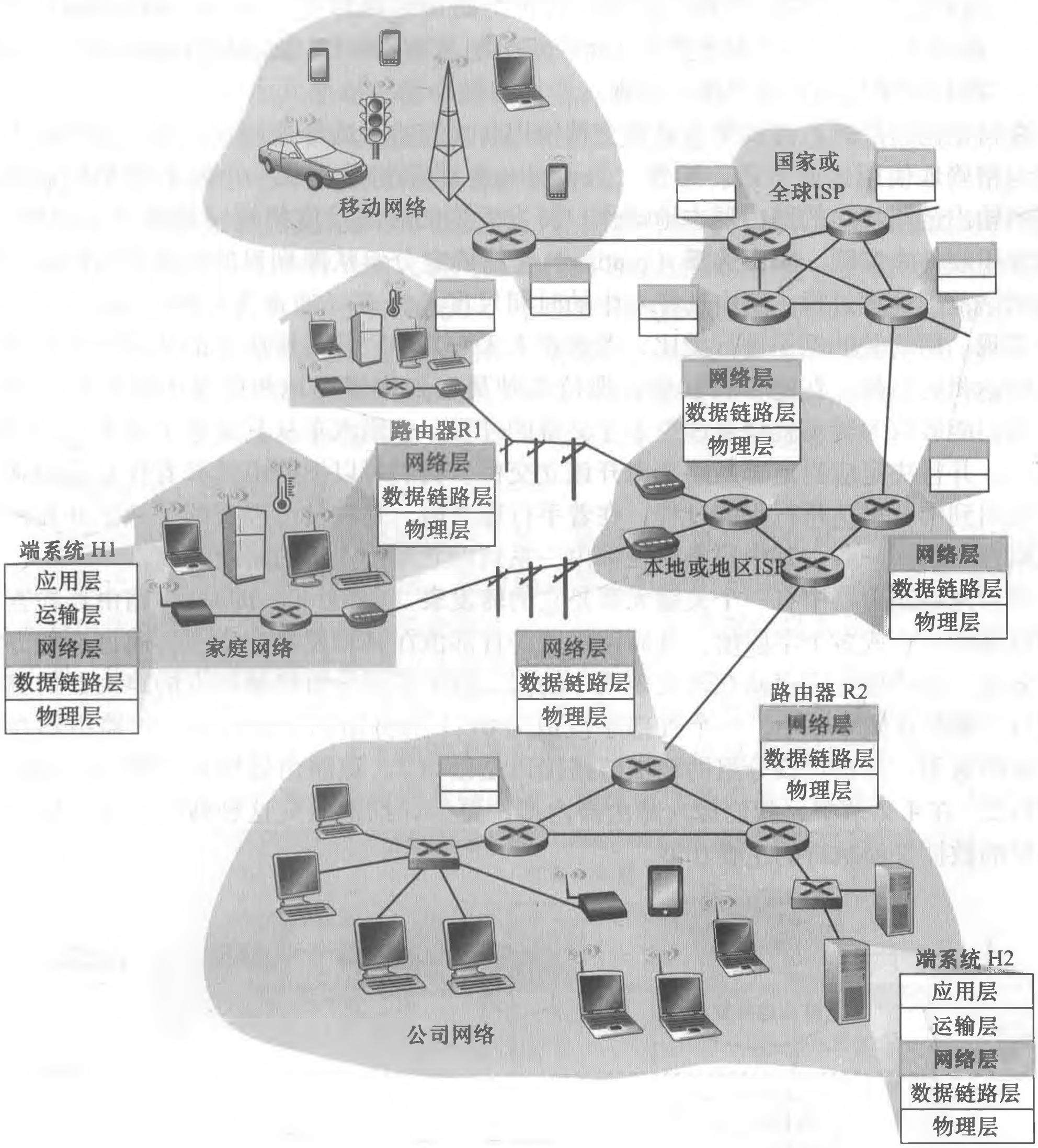

图 4-1　网络层

4.1.1　转发和路由选择：数据平面和控制平面

网络层的作用从表面上看极为简单，即将分组从一台发送主机移动到一台接收主机。为此，需要使用两种重要的网络层功能：

- 转发。当一个分组到达某路由器的一条输入链路时，该路由器必须将该分组移动到适当的输出链路。例如，在图 4-1 中来自主机 H1 到路由器 R1 的一个分组，必须向到达 H2 的路径上的下一台路由器转发。如我们将看到的那样，转发是在数据平面中实现的唯一功能（尽管是最为常见和重要的功能）。在最为常见的场合（我们将在 4.4 节中讨论），分组也可能被现有的路由器阻挡（例如，该分组来源于一个已知的恶意主机，或者该分组发向一个被禁止的目的主机），或者可能是冗余的并经过多条出链路发送。

- 路由选择。当分组从发送方流向接收方时，网络层必须决定这些分组所采用的路由或路径。计算这些路径的算法被称为**路由选择算法**（routing algorithm）。例如，在图4-1中一个路由选择算法将决定分组从H1到H2流动所遵循的路径。路由选择在网络层的控制平面中实现。

在讨论网络层时，许多作者经常交替使用转发和路由选择这两个术语。我们在本书中将更为精确地使用这些术语。**转发**（forwarding）是指将分组从一个输入链路接口转移到适当的输出链路接口的路由器本地动作。转发发生的时间尺度很短（通常为几纳秒），因此通常用硬件来实现。**路由选择**（routing）是指确定分组从源到目的地所采取的端到端路径的网络范围处理过程。路由选择发生的时间尺度长得多（通常为几秒），因此通常用软件来实现。用驾驶的例子进行类比，考虑在1.3.1节中旅行者所历经的从宾夕法尼亚州到佛罗里达州的行程。在这个行程中，那位驾驶员在到佛罗里达州的途中经过了许多立交桥。我们能够认为转发就像通过单个立交桥的过程：一辆汽车从其道路上进入立交桥的一个入口，并且决定应当走哪条路来离开该立交桥。我们可以把路由选择看作是规划从宾夕法尼亚州到佛罗里达州行程的过程：在着手行程之前，驾驶员已经查阅了地图并在许多可能的路径中选择一条，其中每条路径都由一系列经立交桥连接的路段组成。

每台网络路由器中有一个关键元素是它的**转发表**（forwarding table）。路由器检查到达分组首部的一个或多个字段值，进而使用这些首部值在其转发表中索引，通过这种方法来转发分组。这些值对应存储在转发表项中的值，指出了该分组将被转发的路由器的输出链路接口。例如在图4-2中，一个首部字段值为0111的分组到达路由器。该路由器在它的转发表中索引，并确定该分组的输出链路接口是接口2。该路由器则在内部将该分组转发到接口2。在4.2节中，我们深入路由器内部，更为详细地研究这种转发功能。转发是由网络层的数据平面执行的主要功能。

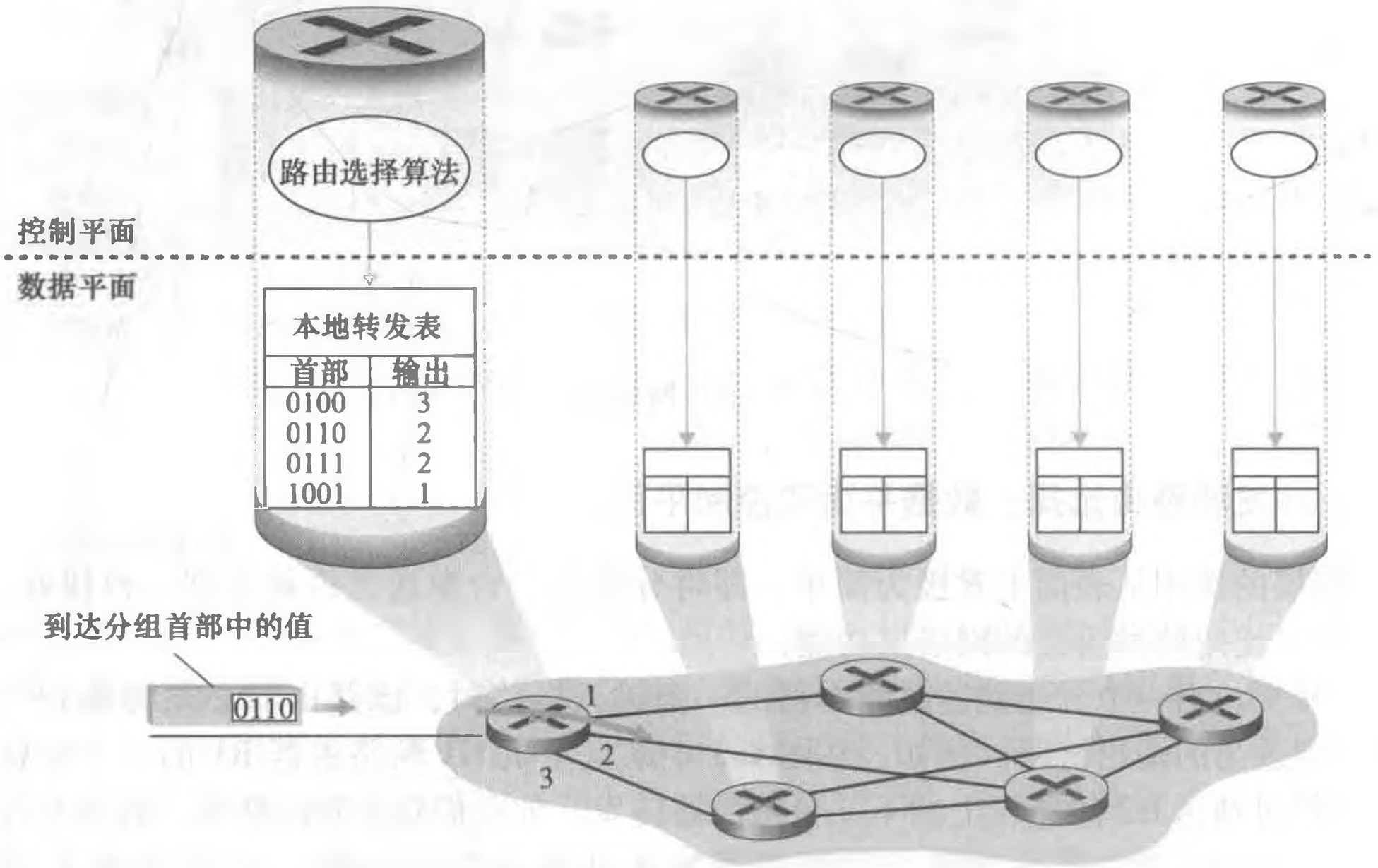

图4-2　路由选择算法决定转发表中的值

1. 控制平面：传统的方法

你也许现在想知道路由器中的转发表一开始是如何配置的。这是一个关键问题，它揭

示了路由选择和转发间的重要相互作用关系。如图 4-2 所示，路由选择算法决定了插入该路由器转发表的内容。在这个例子中，路由选择算法运行在每台路由器中，并且在每台路由器中都包含转发和路由选择两种功能。如我们将在 5.3 节和 5.4 节中所见，在一台路由器中的路由选择算法与在其他路由器中的路由选择算法通信，以计算出它的转发表的值。这种通信是如何执行的呢？通过根据路由选择协议交换包含路由选择信息的路由选择报文！我们将在 5.2～5.4 节讨论路由选择算法和协议。

通过考虑网络中的假想情况（不真实的，但技术上是可行的），也就是说路由器中物理上存在的所有转发表的内容是由人类网络操作员直接配置的，进一步说明转发和路由选择功能的区别和不同目的。在这种情况下，不需要任何路由选择协议！当然，这些人类操作员将需要彼此交互，以确保该转发表的配置能使分组到达它们想要到达的目的地。出现下列现象也很可能：人工配置更容易出错，并且对于网络拓扑变化的响应比起路由选择协议来更慢。我们要为所有网络具有转发和路由选择功能而感到幸运！

2. 控制平面：SDN 方法

图 4-2 中显示的实现路由选择功能的方法，是路由选择厂商在其产品中采用的传统方法，至少最近还是如此。使用该方法，每台路由器都有一个与其他路由器的路由选择组件通信的路由选择组件。然而，对人类能够手动配置转发表的观察启发我们，对于控制平面功能来说，也许存在其他方式来确定数据平面转发表的内容。

图 4-3 显示了从路由器物理上分离的另一种方法，远程控制器计算和分发转发表以供每台路由器所使用。注意到图 4-2 和图 4-3 的数据平面组件是相同的。而在图 4-3 中，控制平面路由选择功能与物理的路由器是分离的，即路由选择设备仅执行转发，而远程控制

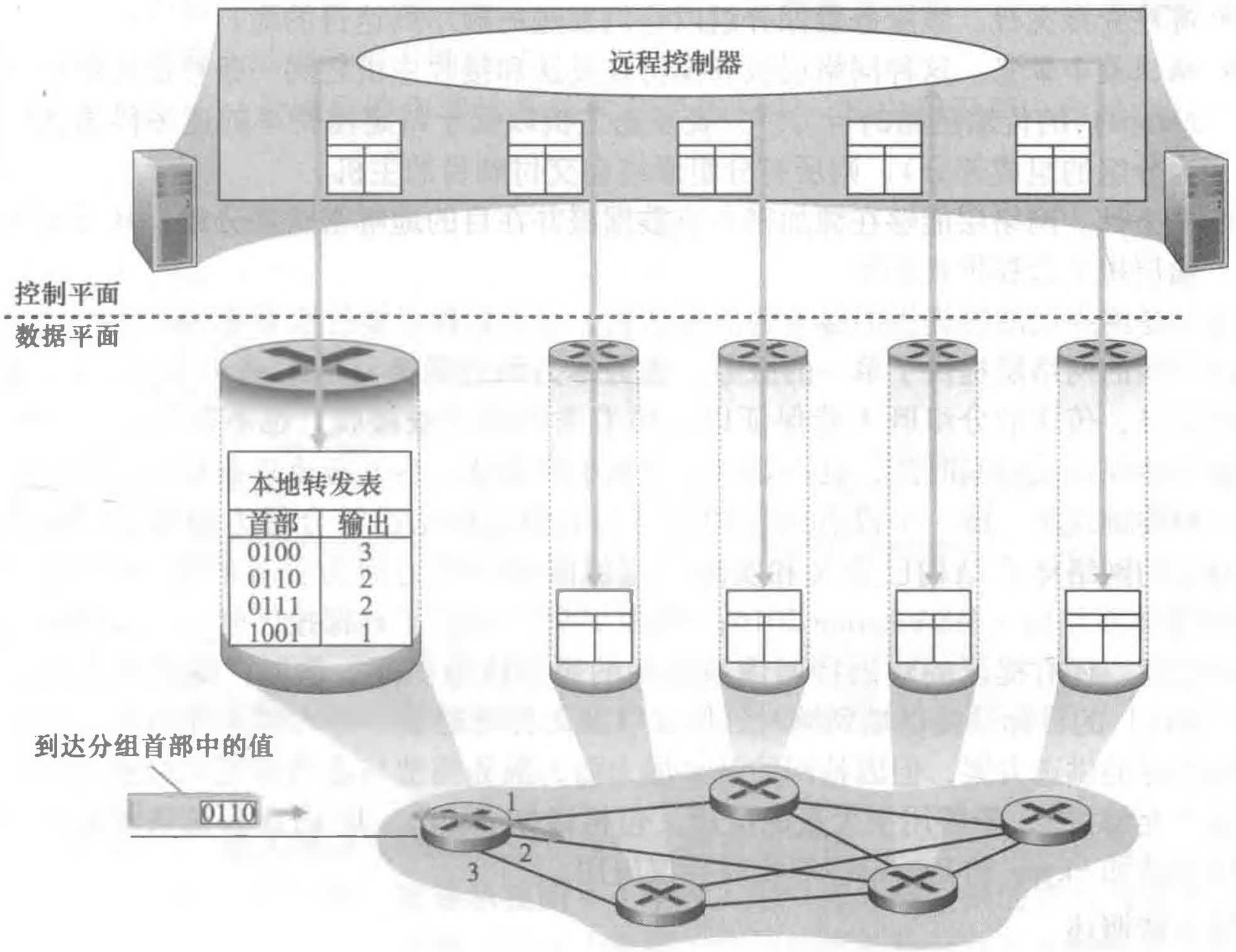

图 4-3　远程控制器确定并分发转发表中的值

器计算并分发转发表。远程控制器可能实现在具有高可靠性和冗余的远程数据中心中，并可能由 ISP 或某些第三方管理。路由器和远程控制器是如何通信的呢？通过交换包含转发表和其他路由选择信息的报文。显示在图 4-3 中的控制平面方法是**软件定义网络**（Software-Defined Networking，SDN）的本质，因为计算转发表并与路由器交互的控制器是用软件实现的，故网络是“软件定义”的。这些软件实现也越来越开放，换言之类似于 Linux 操作系统代码，这些代码可为公众所用，允许 ISP（以及网络研究者和学生）去创新并对控制网络层功能的软件提出更改建议。我们将在 5.5 节中讨论 SDN 控制平面。

4.1.2　网络服务模型

在钻研网络层的数据平面之前，我们将以开阔的视野来专注于我们引入的新东西并考虑网络层可能提供的不同类型的服务。当位于发送主机的运输层向网络传输分组时（即在发送主机中将分组向下交给网络层），运输层能够指望网络层将该分组交付给目的地吗？当发送多个分组时，它们会按发送顺序按序交付给接收主机的运输层吗？发送两个连续分组的时间间隔与接收到这两个分组的时间间隔相同吗？网络层会提供关于网络中拥塞的反馈信息吗？在发送主机与接收主机中连接运输层通道的抽象视图（特性）是什么？对这些问题和其他问题的答案由网络层提供的服务模型所决定。**网络服务模型**（network service model）定义了分组在发送与接收端系统之间的端到端运输特性。

我们现在考虑网络层能提供的某些可能的服务。这些服务可能包括：

- 确保交付。该服务确保分组将最终到达目的地。
- 具有时延上界的确保交付。该服务不仅确保分组的交付，而且在特定的主机到主机时延上界内（例如在 100ms 内）交付。
- 有序分组交付。该服务确保分组以它们发送的顺序到达目的地。
- 确保最小带宽。这种网络层服务模仿在发送和接收主机之间一条特定比特率（例如 1Mbps）的传输链路的行为。只要发送主机以低于特定比特率的速率传输比特（作为分组的组成部分），则所有分组最终会交付到目的主机。
- 安全性。网络层能够在源加密所有数据报并在目的地解密这些分组，从而对所有运输层报文段提供机密性。

这只是网络层能够提供的服务的部分列表，有无数种可能的服务变种。

因特网的网络层提供了单一的服务，称为**尽力而为服务**（best-effort service）。使用尽力而为服务，传送的分组既不能保证以它们发送的顺序被接收，也不能保证它们最终交付；既不能保证端到端时延，也不能保证有最小的带宽。尽力而为服务看起来是根本无服务的一种委婉说法，即一个没有向目的地交付分组的网络也符合尽力而为交付服务的定义！其他的网络体系结构已定义和实现了超过因特网尽力而为服务的服务模型。例如，ATM 网络体系结构［MFA Forum 2016；Black 1995］提供了确保按序时延、有界时延和确保最小带宽。还有提议的对因特网体系结构的服务模型扩展，例如，集成服务体系结构［RFC 1633］的目标是提供端到端时延保证以及无拥塞通信。令人感兴趣的是，尽管有这些研发良好的供选方案，但因特网的基本尽力而为服务模型与适当带宽供给相结合已被证明超过“足够好”，能够用于大量的应用，包括诸如 Netflix、IP 语音和视频等流式视频服务，以及诸如 Skype 和 Facetime 等实时会议应用。

第 4 章概述

在提供了网络层的概述后，我们将在本章后续几节中讨论网络层的数据平面组件。在

4.2 节中，我们将深入探讨路由器的内部硬件操作，包括输入和输出分组处理、路由器的内部交换机制以及分组排队和调度。在 4.3 节中，我们将学习传统的 IP 转发，其中分组基于它们的目的 IP 地址转发到输出端口。我们将学习到 IP 寻址、令人称道的 IPv4 和 IPv6 协议等。在 4.4 节中，我们将涉及更为一般的转发，此时分组可以基于大量首部值（即不仅基于目的 IP 地址）转发到输出端口。分组可能在路由器中受阻或冗余，或者可能让某些首部字段重写，即所有都在软件控制之下完成。这种分组转发的更为一般的形式是现代网络数据平面的关键组件，包括软件定义网络（SDN）中的数据平面。

我们在这里顺便提到，许多计算机网络研究者和从业人员经常互换地使用转发和交换这两个术语。我们在这本教科书中也将互换使用这些术语。在我们开始讨论术语的主题时，还需要指出经常互换使用的两个其他术语，但我们将更为小心地使用它们。我们将约定术语分组交换机是指一台通用分组交换设备，它根据分组首部字段中的值，从输入链路接口到输出链路接口转移分组。某些分组交换机称为**链路层交换机**（link-layer switch）（在第 6 章仔细学习），基于链路层帧中的字段值做出转发决定，这些交换机因此被称为链路层（第 2 层）设备。其他分组交换机称为**路由器**（router），基于网络层数据报中的首部字段值做出转发决定。路由器因此是网络层（第 3 层）设备。（为了全面理解这种重要区别，你可能要回顾 1.5.2 节，在那里我们讨论了网络层数据报和链路层帧及其关系。）因为在本章中我们关注的是网络层，所以我们将主要使用术语路由器来代替交换机。

4.2　路由器工作原理

既然我们已经概述了网络层中的数据平面和控制平面、转发与路由选择之间的重要区别以及网络层的服务与功能，我们将注意力转向网络层的**转发功能**，即实际将分组从一台路由器的入链路传送到适当的出链路。

图 4-4 显示了一个通用路由器体系结构的总体视图，其中标识了一台路由器的 4 个组件。

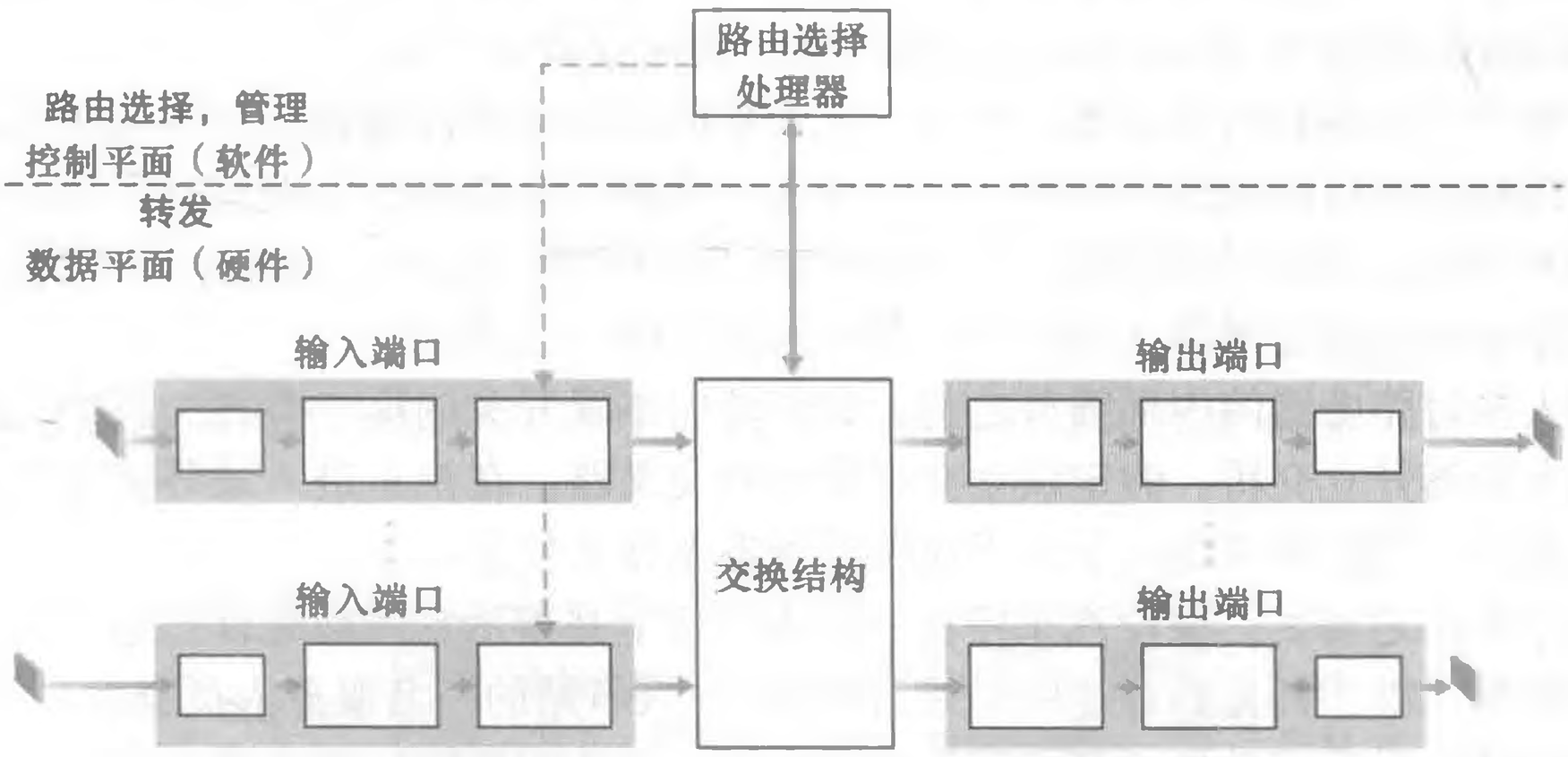

图 4-4　路由器体系结构

- 输入端口。**输入端口**（input port）执行几项重要功能。它在路由器中执行终结入物理链路的物理层功能，这显示在图 4-4 中输入端口部分最左侧的方框与输出端口部分最右侧的方框中。它还要与位于入链路远端的数据链路层交互来执行数据链路层功能，这显示在输入与输出端口部分中间的方框中。也许更为重要的是，在输入端

口还要执行查找功能，这显示在输入端口最右侧的方框中。正是在这里，通过查询转发表决定路由器的输出端口，到达的分组通过路由器的交换结构转发到输出端口。控制分组（如携带路由选择协议信息的分组）从输入端口转发到路由选择处理器。注意这里的“端口”一词，指的是路由器的物理输入和输出接口，这完全不同于第2、3章中所讨论的与网络应用程序和套接字相关联的软件端口。在实践中，一台路由器所支持的端口数量范围较大，从企业路由器具有数量相对少的端口，到位于某ISP边缘的路由器具有数以百计10Gbps端口（其中入线路的数量趋于最大）。例如，边缘路由器Juniper MX2020具有80Tbps的总体路由器系统容量，支持多达960个10Gbps以太网端口［Juniper MX 2020 2016］。

- *交换结构*。交换结构将路由器的输入端口连接到它的输出端口。这种交换结构完全包含在路由器之中，即它是一个网络路由器中的网络！
- *输出端口*。输出端口存储从交换结构接收的分组，并通过执行必要的链路层和物理层功能在输出链路上传输这些分组。当一条链路是双向的时（即承载两个方向的流量），输出端口通常与该链路的输入端口成对出现在同一线路卡上。
- *路由选择处理器*。路由选择处理器执行控制平面功能。在传统的路由器中，它执行路由选择协议（我们将在5.3节和5.4节学习），维护路由选择表与关联链路状态信息，并为该路由器计算转发表。在SDN路由器中，路由选择处理器（在其他活动中）负责与远程控制器通信，目的是接收由远程控制器计算的转发表项，并在该路由器的输入端口安装这些表项。路由选择处理器还执行网络管理功能，我们将在5.7节学习相关内容。

路由器的输入端口、输出端口和交换结构几乎总是用硬件实现，如图4-4所示。为了理解为何需要用硬件实现，考虑具有10Gbps输入链路和64字节的IP数据报，其输入端口在另一个数据报到达前仅有51.2ns来处理数据报。如果N个端口结合在一块线路卡上（因为实践中常常这样做），数据报处理流水线必须以N倍速率运行，这远快过软件实现的速率。转发硬件既能够使用路由器厂商自己的硬件设计来实现，也能够使用购买的商用硅片（例如由英特尔和Broadcom公司所出售）的硬件设计来实现。

当数据平面以纳秒时间尺度运行时，路由器的控制功能以毫秒或秒时间尺度运行，这些控制功能包括执行路由选择协议、对上线或下线的连接链路进行响应、与远程控制器通信（在SDN场合）和执行管理功能。因而这些**控制平面**（control plane）的功能通常用软件实现并在路由选择处理器（通常是一种传统的CPU）上执行。

在深入探讨路由器的内部细节之前，我们转向本章开头的那个类比，其中分组转发好比汽车进入和离开立交桥。假定该立交桥是环状交叉路，在汽车进入该环状交叉路前，需要做一点处理。我们来考虑一下对于这种处理需要什么信息。

- *基于目的地转发*。假设汽车停在一个入口站上并指示它的最终目的地（并非在本地环状交叉路，而是其旅途的最终目的地）。入口站的一名服务人员查找最终目的地，决定通向最后目的地的环状交叉路的出口，并告诉驾驶员要走哪个出口。
- *通用转发*。除了目的地之外，服务人员也能够基于许多其他因素确定汽车的出口匝道。例如，所选择的出口匝道可能与该汽车的起点如发行该车牌照的州有关。来自某些州的汽车可能被引导使用某个出口匝道（经过一条慢速道路通向目的地），而来自其他州的汽车可能被引导使用一个不同的出口匝道（经过一条高速路通向目的地）。基于汽车的模型、品牌和寿命，可能做出相同的决定。或者认为不适合上路

的汽车可能被阻止并且不允许通过环状交叉路。就通用转发来说，许多因素都会对服务人员为给定汽车选择出口匝道产生影响。

一旦汽车进入环状交叉路（该环状交叉路可能挤满了从其他输入道路进入的其他汽车，朝着其他环状交叉路出口前进），并且最终离开预定的环状交叉路出口匝道，在这里可能遇到了从该出口离开环状交叉路的其他汽车。

在这个类比中，我们能够在图 4-4 中识别最重要的路由器组件：入口道路和入口站对应于输入端口（具有查找功能以决定本地输出端口）；环状交叉路对应于交换结构；环状交叉路出口匝道对应于输出端口。借助于这个类比，我们可以考虑瓶颈可能出现的地方。如果汽车以极快的速率到达（例如，该环状交叉路位于德国或意大利!）而车站服务人员很慢，将发生什么情况？这些服务人员必须工作得多快，以确保在入口路上没有车辆拥堵？甚至对于极快的服务人员，如果汽车在环状交叉路上开得很慢，将发生什么情况，拥堵仍会出现吗？如果大多数进入的汽车都要在相同的出口匝道离开环状交叉路，将发生什么情况，在出口匝道或别的什么地方会出现拥堵吗？如果我们要为不同的汽车分配优先权，或先行阻挡某些汽车进入环状交叉路，环状交叉路将如何运行？这些全都与路由器和交换机设计者面对的问题形成类比。

在下面的各小节中，我们将更为详细地考察路由器功能。［Iyer 2008；Chao 2001；Chuang 2005；Turner 1988；McKeown 1997a；Partridge 1998；Sopranos 2011］提供了对一些特定路由器体系结构的讨论。为了具体和简单起见，我们在本节中初始假设转发决定仅基于分组的目的地址，而非基于通用的分组首部字段。我们将在 4.4 节中学习更为通用的分组转发情况。

4.2.1　输入端口处理和基于目的地转发

图 4-5 中显示了一个更详细的输入处理的视图。如前面讨论的那样，输入端口的线路端接功能与链路层处理实现了用于各个输入链路的物理层和链路层。在输入端口中执行的查找对于路由器运行是至关重要的。正是在这个地方，路由器使用转发表来查找输出端口，使得到达的分组能经过交换结构转发到该输出端口。转发表是由路由选择处理器计算和更新的（使用路由选择协议与其他网络路由器中的路由选择处理器进行交互），或者转发表接收来自远程 SDN 控制器的内容。转发表从路由选择处理器经过独立总线（例如一个 PCI 总线）复制到线路卡，在图 4-4 中该总线由从路由选择处理器到输入线路卡的虚线所指示。使用在每个输入端口的影子副本，转发决策能在每个输入端口本地做出，无须基于每个分组调用集中式路由选择处理器，因此避免了集中式处理的瓶颈。

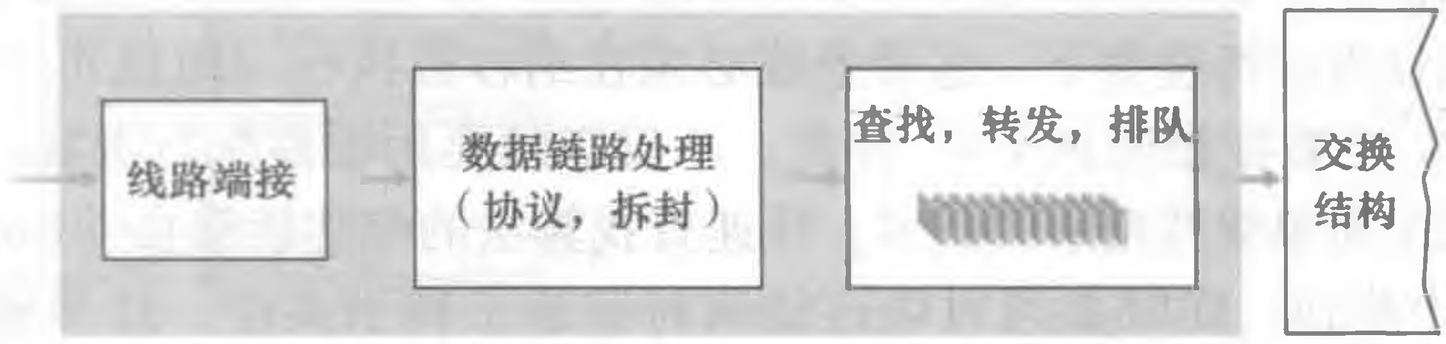

图 4-5　输入端口处理

现在我们来考虑“最简单”的情况，一个入分组基于该分组的目的地址交换到输出端口。在 32 比特 IP 地址的情况下，转发表的蛮力实现将针对每个目的地址有一个表项。因

为有超过40亿个可能的地址，选择这种方法总体上是不可行的。

作为一个说明怎样处理规模问题的例子，假设我们的路由器具有4条链路，编号0到3，分组以如下方式转发到链路接口：

目的地址范围	链路接口
11001000 00010111 00010000 00000000 到 11001000 00010111 00010111 11111111	0
11001000 00010111 00011000 00000000 到 11001000 00010111 00011000 11111111	1
11001000 00010111 00011001 00000000 到 11001000 00010111 00011111 11111111	2
其他	3

显然，对于这个例子，在路由器的转发表中没有必要有40亿个表项。例如，我们能够有一个如下仅包括4个表项的转发表：

前缀匹配	链路接口
11001000 00010111 00010	0
11001000 00010111 00011000	1
11001000 00010111 00011	2
其他	3

使用这种风格的转发表，路由器用分组目的地址的**前缀**（prefix）与该表中的表项进行匹配；如果存在一个匹配项，则路由器向与该匹配项相关联的链路转发分组。例如，假设分组的目的地址是11001000 00010111 00010110 10100001，因为该地址的21比特前缀匹配该表的第一项，所以路由器向链路接口0转发该分组。如果一个前缀不匹配前3项中的任何一项，则路由器向链路接口3转发该分组。尽管听起来足够简单，但这里还是有重要的微妙之处。你可能已经注意到一个目的地址可能与不止一个表项相匹配。例如，地址11001000 00010111 00011000 10101010的前24比特与表中的第二项匹配，而该地址的前21比特与表中的第三项匹配。当有多个匹配时，该路由器使用**最长前缀匹配规则**（longest prefix matching rule）；即在该表中寻找最长的匹配项，并向与最长前缀匹配相关联的链路接口转发分组。当在4.3节中详细学习因特网编址时，我们将完全明白使用这种最长前缀匹配规则的理由。

假定转发表已经存在，从概念上讲表查找是简单的，硬件逻辑只是搜索转发表查找最长前缀匹配。但在吉比特速率下，这种查找必须在纳秒级执行（回想我们前面10Gbps链路和一个64字节IP数据报的例子）。因此，不仅必须要用硬件执行查找，而且需要对大型转发表使用超出简单线性搜索的技术；快速查找算法的综述能够在［Gupta 2001，Ruiz-Sanchez 2011］中找到。同时必须对内存访问时间给予特别关注，这导致用嵌入式片上DRAM和更快的SRAM（用作一种DRAM缓存）内存来设计。实践中也经常使用**三态内容可寻址存储器**（Tenary Content Address Memory，TCAM）来查找［Yu 2004］。使用TCAM，一个32比特IP地址被放入内存，TCAM在基本常数时间内返回对该地址的转发表项的内容。Cisco Catalyst 6500和7500系列路由器及交换机能够保存100多万TCAM转发表项

[Cisco TCAM 2014]。

一旦通过查找确定了某分组的输出端口，则该分组就能够发送进入交换结构。在某些设计中，如果来自其他输入端口的分组当前正在使用该交换结构，一个分组可能会在进入交换结构时被暂时阻塞。因此，一个被阻塞的分组必须要在输入端口处排队，并等待稍后被及时调度以通过交换结构。我们稍后将仔细观察分组（位于输入端口与输出端口中）的阻塞、排队与调度。尽管“查找”在输入端口处理中可认为是最为重要的动作，但必须采取许多其他动作：①必须出现物理层和链路层处理，如前面所讨论的那样；②必须检查分组的版本号、检验和以及寿命字段（这些我们将在4.3节中学习），并且重写后两个字段；③必须更新用于网络管理的计数器（如接收到的IP数据报的数目）。

在结束输入端口处理的讨论之前，注意到输入端口查找目的IP地址（“匹配”），然后发送该分组进入交换结构（“动作”）的步骤是一种更为一般的“匹配加动作”抽象的特定情况，这种抽象在许多网络设备中执行，而不仅在路由器中。在链路层交换机（在第6章讨论）中，除了发送帧进入交换结构去往输出端口外，还要查找链路层目的地址，并采取几个动作。在防火墙（在第8章讨论）中，首部匹配给定准则（例如源/目的IP地址和运输层端口号的某种组合）的入分组可能被阻止转发，而防火墙是一种过滤所选择的入分组的设备。在网络地址转换器（NAT，在4.3节讨论）中，一个运输层端口号匹配某给定值的入分组，在转发（动作）前其端口号将被重写。的确，“匹配加动作”抽象不仅作用大，而且在网络设备中无所不在，并且对于我们将在4.4节中学习的通用转发是至关重要的。

4.2.2　交换

交换结构位于一台路由器的核心部位，因为正是通过这种交换结构，分组才能实际地从一个输入端口交换（即转发）到一个输出端口中。交换可以用许多方式完成，如图4-6所示。

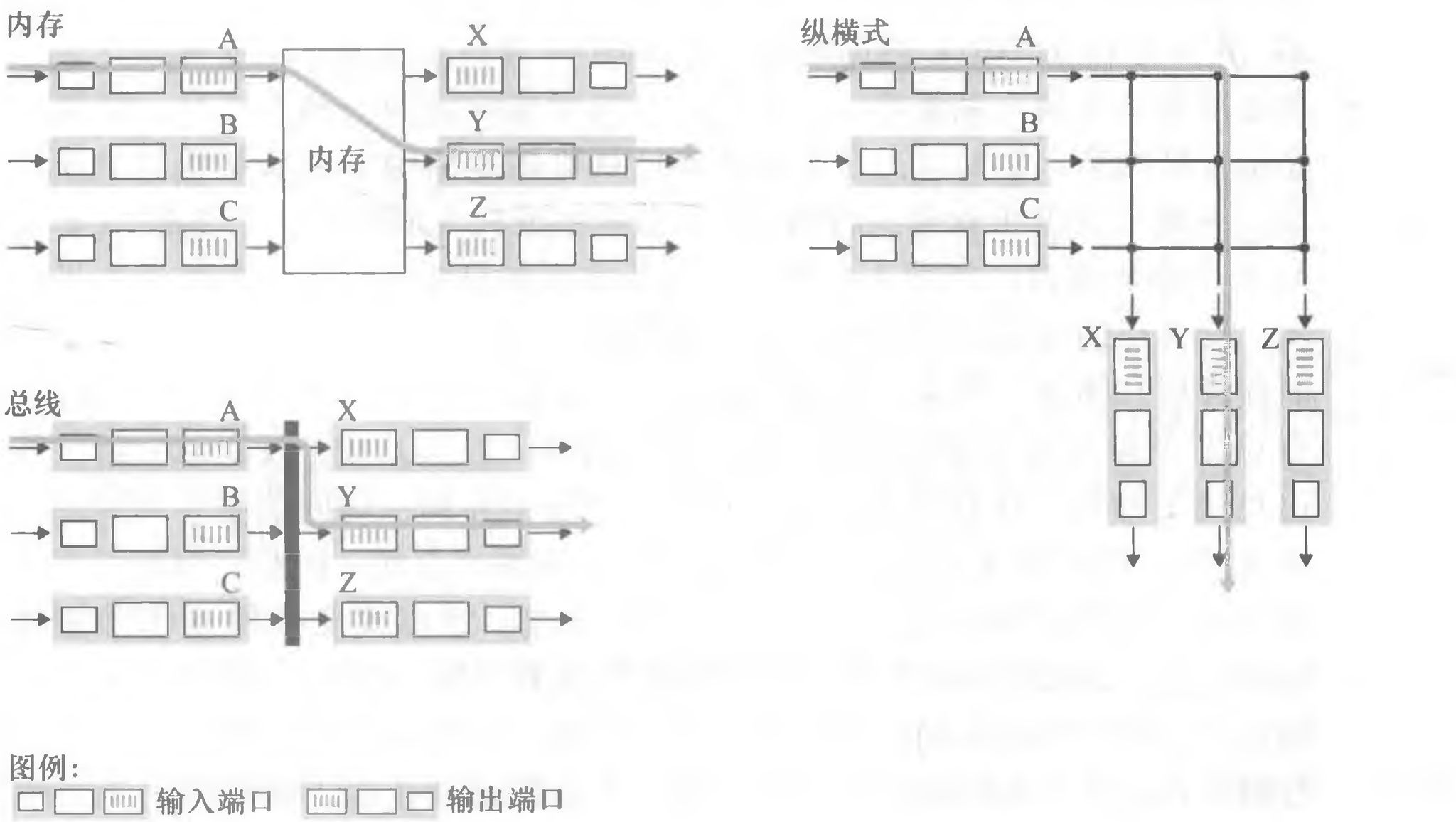

图4-6　三种交换技术