将包放入缓冲区中[①]。到这里，MAC 模块的工作就完成了，接下来网卡会通知计算机收到了一个包。

通知计算机的操作会使用一个叫作中断的机制。在网卡执行接收包的操作的过程中，计算机并不是一直监控着网卡的活动，而是去继续执行其他的任务。因此，如果网卡不通知计算机，计算机是不知道包已经收到了这件事的。网卡驱动也是在计算机中运行的一个程序，因此它也不知道包到达的状态。在这种情况下，我们需要一种机制能够打断计算机正在执行的任务，让计算机注意到网卡中发生的事情，这种机制就是中断。

具体来说，中断的工作过程是这样的。首先，网卡向扩展总线中的中断信号线发送信号，该信号线通过计算机中的中断控制器连接到 CPU。当产生中断信号时，CPU 会暂时挂起正在处理的任务，切换到操作系统中的中断处理程序[②]。然后，中断处理程序会调用网卡驱动，控制网卡执行相应的接收操作。

中断是有编号的，网卡在安装的时候就在硬件中设置了中断号，在中断处理程序中则将硬件的中断号和相应的驱动程序绑定。例如，假设网卡的中断号为 11，则在中断处理程序中将中断号 11 和相应的网卡驱动绑定起来，当网卡发起中断时，就会自动调用网卡驱动了。现在的硬件设备都遵循即插即用[③]规范自动设置中断号，我们没必要去关心中断号了，在以前需要手动设置中断号的年代，经常发生因为设置了错误的中断号而导致网卡无法正常工作的问题。

网卡驱动被中断处理程序调用后，会从网卡的缓冲区中取出收到的包，并通过 MAC 头部中的以太类型字段判断协议的类型。现在我们在大多数情况下都是使用 TCP/IP 协议，但除了 TCP/IP 之外还有很多其他类型的协

① 有一个特殊的例子，其实我们也可以让网卡不检查包的接收方地址，不管是不是自己的包都统统接收下来，这种模式叫作“混杂模式”（Promiscuous Mode）。

② 中断处理程序执行完毕之后，CPU 会继续处理原来的任务。

③ 英文缩写为 PnP（Plug and Play），是一种自动对扩展卡和周边设备进行配置的功能。

议，例如 NetWare 中使用的 IPX/SPX，以及 Mac 电脑中使用的 AppleTalk 等协议。这些协议都被分配了不同的以太类型，如 0080（十六进制）代表 IP 协议，网卡驱动就会把这样的包交给 TCP/IP 协议栈；如果是 809B 则表示 AppleTalk 协议，就把包交给 AppleTalk 协议栈，以此类推[①]。

按照探索之旅的思路，大家可能会认为向 Web 服务器发送包之后，后面收到的一定是 Web 服务器返回的包，其实并非如此。计算机中同时运行了很多程序，也会同时进行很多通信操作，因此收到的包也有可能是其他应用程序的。不过，即便如此也没问题，网卡不会关心包里的内容，只要按照以太类型将包交给对应的协议栈就可以了。接下来，协议栈会判断这个包应该交给哪个应用程序，并进行相应的处理。

2.5.11 将服务器的响应包从 IP 传递给 TCP

下面我们假设 Web 服务器返回了一个网络包，那么协议栈会进行哪些处理呢[②]？服务器返回的包的以太类型应该是 0800，因此网卡驱动会将其交给 TCP/IP 协议栈来进行处理。接下来就轮到 IP 模块先开始工作了，第一步是检查 IP 头部，确认格式是否正确。如果格式没有问题，下一步就是查看接收方 IP 地址。如果接收网络包的设备是一台 Windows 客户端计算机，那么服务器返回的包的接收方 IP 地址应该与客户端网卡的地址一致，检查确认之后我们就可以接收这个包了。

如果接收方 IP 地址不是自己的地址，那一定是发生了什么错误。客户端计算机不负责对包进行转发，因此不应该收到不是发给自己的包[③]。当发

① 前提是操作系统内部存在以太类型所对应的协议栈。如果不存在相应的协议栈，则会视作错误，直接丢弃这个包。

② 正如介绍发送操作时提到过的一样，IP 模块的工作方式对于 TCP 模块所委派的任何操作都是共通的。

③ 如果是服务器就不一定了。服务器的操作系统具备和路由器相同的包转发功能，当打开这一功能时，它就可以像路由器一样对包进行转发。在这种情况下，当收到不是发给自己的包的时候，就会像路由器一样执行包转发操作。由于这一过程和路由器是相同的，因此我们将在第 3 章探索路由器时进行介绍。

生这样的错误时，IP 模块会通过 ICMP 消息将错误告知发送方（图 2.1）。ICMP 规定了各种类型的消息，如表 2.4 所示。当我们遇到这个错误时，IP 模块会通过表 2.4 中的 Destination unreachable 消息通知对方。从这张表的内容中我们可以看到在包的接收和转发过程中能够遇到的各种错误，因此希望大家看一看这张表。

表 2.4　主要的 ICMP 消息

消息	类型	含　义
Echo reply	0	响应 Echo 消息
Destination unreachable	3	出于某些原因包没有到达目的地而是被丢弃，则通过此消息通知发送方。可能的原因包括目标 IP 地址在路由表中不存在；目标端口号不存在对应的套接字；需要分片，但分片被禁用
Source quench	4	当发送的包数量超过路由器的转发能力时，超过的部分会被丢弃，这时会通过这一消息通知发送方。但是，并不是说遇到这种情况一定会发送这一消息。当路由器的性能不足时，可能连这条消息都不发送，就直接把多余的包丢弃了。当发送方收到这条消息时，必须降低发送速率
Redirect	5	当查询路由表后判断该包的入口和出口为同一个网络接口时，则表示这个包不需要该路由器转发，可以由发送方直接发送给下一个路由器。遇到这种情况时，路由器会发送这条消息，给出下一个路由器的 IP 地址，指示发送方直接发送过去
Echo	8	ping 命令发送的消息。收到这条消息的设备需返回一个 Echo reply 消息，以便确认通信对象是否存在
Time exceeded	11	由于超过了 IP 头部中的 TTL 字段表示的存活时间而被路由器丢弃，此时路由器会向发送方发送这条消息
Parameter problem	12	由于 IP 头部字段存在错误而被丢弃，此时会向发送方发送这条消息

如果接收方 IP 地址正确，则这个包会被接收下来，这时还需要完成另一项工作。IP 协议有一个叫作分片的功能，具体的内容我们将在第 3 章探

索路由器时进行介绍。简单来说，网线和局域网中只能传输小包，因此需要将大的包切分成多个小包。如果接收到的包是经过分片的，那么 IP 模块会将它们还原成原始的包。分片的包会在 IP 头部的标志字段中进行标记，当收到分片的包时，IP 模块会将其暂存在内部的内存空间中，然后等待 IP 头部中具有相同 ID 的包全部到达，这是因为同一个包的所有分片都具有相同的 ID。此外，IP 头部还有一个分片偏移量（fragment offset）字段，它表示当前分片在整个包中所处的位置。根据这些信息，在所有分片全部收到之后，就可以将它们还原成原始的包，这个操作叫作分片重组。

到这里，IP 模块的工作就结束了，接下来包会被交给 TCP 模块。TCP 模块会根据 IP 头部中的接收方和发送方 IP 地址，以及 TCP 头部中的接收方和发送方端口号来查找对应的套接字[①]。找到对应的套接字之后，就可以根据套接字中记录的通信状态，执行相应的操作了。例如，如果包的内容是应用程序数据，则返回确认接收的包，并将数据放入缓冲区，等待应用程序来读取；如果是建立或断开连接的控制包，则返回相应的响应控制包，并告知应用程序建立和断开连接的操作状态。

---

① 严格来说，TCP 模块和 IP 模块有各自的责任范围，TCP 头部属于 TCP 的责任范围，而 IP 头部属于 IP 模块的责任范围。根据这样的逻辑，当包交给 TCP 模块之后，TCP 模块需要查询 IP 头部中的接收方和发送方 IP 地址来查找相应的套接字，这个过程就显得有点奇怪。因为 IP 头部是 IP 模块负责的，TCP 模块去查询它等于是越权了。如果要避免越权，应该对两者进行明确的划分，IP 模块只向 TCP 模块传递 TCP 头部以及它后面的数据，而对于 IP 头部中的重要信息，即接收方和发送方的 IP 地址，则由 IP 模块以附加参数的形式告知 TCP 模块。然而，如果根据这种严格的划分来开发程序的话，IP 模块和 TCP 模块之间的交互过程必然会产生成本，而且 IP 模块和 TCP 模块进行类似交互的场景其实非常多，总体的交互成本就会很高，程序的运行效率就会下降。因此，就像之前提过的一样，不妨将责任范围划分得宽松一些，将 TCP 和 IP 作为一个整体来看待，这样可以带来更大的灵活性。

此外，关于为什么查找套接字同时需要接收方和发送方的 IP 地址和端口号，我们会在第 6 章介绍端口号机制时一起讲解。

2.6 UDP 协议的收发操作

2.6.1 不需要重发的数据用 UDP 发送更高效

跟着第 1 章的脚步，本章我们探索了通过套接字收发数据的整个过程，这个过程到这里已经告一段落了。接下来，网络包会从计算机出来跑向集线器，这个过程我们将在下一章来介绍，现在先来说点题外话。

大多数的应用程序都像之前介绍的一样使用 TCP 协议来收发数据，但当然也有例外。有些应用程序不使用 TCP 协议，而是使用 UDP 协议来收发数据。向 DNS 服务器查询 IP 地址的时候我们用的也是 UDP 协议。下面就简单介绍一下 UDP 协议。

其实 TCP 中就包含了 UDP 的一些要点。TCP 的工作方式十分复杂，如果我们能够理解 TCP 为什么要设计得如此复杂，也就能够理解 UDP 了。那么，为什么要设计得如此复杂呢？因为我们需要将数据高效且可靠地发送给对方。为了实现可靠性，我们就需要确认对方是否收到了我们发送的数据，如果没有还需要再发一遍。

要实现上面的要求，最简单的方法是数据全部发送完毕之后让接收方返回一个接收确认。这样一来，如果没收到直接全部重新发送一遍就好了，根本不用像 TCP 一样要管理发送和确认的进度。但是，如果漏掉了一个包就要全部重发一遍，怎么看都很低效。为了实现高效的传输，我们要避免重发已经送达的包，而是只重发那些出错的或者未送达的包。TCP 之所以复杂，就是因为要实现这一点。

不过，在某种情况下，即便没有 TCP 这样复杂的机制，我们也能够高效地重发数据，这种情况就是数据很短，用一个包就能装得下。如果只有一个包，就不用考虑哪个包未送达了，因为全部重发也只不过是重发一个包而已，这种情况下我们就不需要 TCP 这样复杂的机制了。而且，如果不使用 TCP，也不需要发送那些用来建立和断开连接的控制包了。此外，我们发送了数据，对方一般都会给出回复，只要将回复的数据当作接收确认

就行了，也不需要专门的接收确认包了。

2.6.2 控制用的短数据

这种情况就适合使用 UDP。像 DNS 查询等交换控制信息的操作基本上都可以在一个包的大小范围内解决，这种场景中就可以用 UDP 来代替 TCP[①]。UDP 没有 TCP 的接收确认、窗口等机制，因此在收发数据之前也不需要交换控制信息，也就是说不需要建立和断开连接的步骤，只要在从应用程序获取的数据前面加上 UDP 头部，然后交给 IP 进行发送就可以了（表 2.5）。接收也很简单，只要根据 IP 头部中的接收方和发送方 IP 地址，以及 UDP 头部中的接收方和发送方端口号，找到相应的套接字并将数据交给相应的应用程序就可以了。除此之外，UDP 协议没有其他功能了，遇到错误或者丢包也一概不管。因为 UDP 只负责单纯地发送包而已，并不像 TCP 一样会对包的送达状态进行监控，所以协议栈也不知道有没有发生错误。但这样并不会引发什么问题，因此出错时就收不到来自对方的回复，应用程序会注意到这个问题，并重新发送一遍数据。这样的操作本身并不复杂，也并不会增加应用程序的负担。

① UDP 可发送的数据最大长度为 IP 包的最大长度减去 IP 头部和 UDP 头部的长度。不过，这个长度与 MTU、MSS 不是一个层面上的概念。MTU 和 MSS 是基于以太网和通信线路上网络包的最大长度来计算的，而 IP 包的最大长度是由 IP 头部中的“全长”字段决定的。“全长”字段的长度为 16 比特，因此从 IP 协议规范来看，IP 包的最大长度为 65 535 字节，再减去 IP 头部和 UDP 头部的长度，就是 UDP 协议所能发送的数据最大长度。如果不考虑可选字段的话，一般来说 IP 头部为 20 字节，UDP 头部为 8 字节，因此 UDP 的最大数据长度为 65 507 字节。当然，这么长的数据已经超过了以太网和通信线路的最大传输长度，因此需要让 IP 模块使用分片功能拆分之后再传输。

表 2.5　UDP 头部中的控制信息

字段名称		长度（比特）	含　义
UDP 头部（8 字节）	发送方端口号	16	网络包发送方的端口号
	接收方端口号	16	网络包接收方的端口号
	数据长度	16	UDP 头部后面数据的长度
	校验和	16	用于校验错误

2.6.3　音频和视频数据

还有另一个场景会使用 UDP，就是发送音频和视频数据的时候。音频和视频数据必须在规定的时间内送达，一旦送达晚了，就会错过播放时机，导致声音和图像卡顿。如果像 TCP 一样通过接收确认响应来检查错误并重发，重发的过程需要消耗一定的时间，因此重发的数据很可能已经错过了播放的时机。一旦错过播放时机，重发数据也是没有用的，因为声音和图像已经卡顿了，这是无法挽回的。当然，我们可以用高速线路让重发的数据能够在规定的时间内送达，但这样一来可能要增加几倍的带宽才行[①]。

此外，音频和视频数据中缺少了某些包并不会产生严重的问题，只是会产生一些失真或者卡顿而已，一般都是可以接受的[②]。

在这些无需重发数据，或者是重发了也没什么意义的情况下，使用 UDP 发送数据的效率会更高。

本章我们探索了在收发数据时，操作系统中的协议栈是如何工作的，以及网卡是如何将包转换成电信号通过网线发送出去的。到这里，我们的网络包已经沿着网线流出了客户端计算机，下一章，我们将探索网络包如何经过集线器、交换机、路由器等设备，最终到达互联网。

① UDP 经常会被防火墙阻止，因此当需要穿越防火墙传输音频和视频数据时，尽管需要消耗额外的带宽，但有时候也只能使用 TCP。

② 如果错误率太高，超过了可接受的限度，那么另当别论。此外，也有一些情况下连一丁点卡顿都不允许，当然这种情况相当特殊。

小测验

本章的旅程告一段落，我们为大家准备了一些小测验题目，确认一下自己的成果吧。

问题

1. 表示网络包收件人的接收方 IP 地址是位于 IP 头部还是 TCP 头部中呢？
2. 端口号用来指定服务器程序的种类，那么它位于 TCP 头部还是 IP 头部中呢？
3. 会对包是否正确送达进行确认的是 TCP 还是 IP 呢？
4. 根据 IP 地址查询 MAC 地址的机制叫什么？
5. 在收到 ACK 号之前继续发送下一个包的方式叫什么？

Column
网络术语其实很简单

插进 Socket 里的是灯泡还是程序

探索队员：Socket 库也好，套接字（socket）也好，这个名字到底是怎么来的呢？

探索队长：你知道灯泡的插座吗？就是灯具里面把灯泡拧进去的那个孔。

队员：知道呀。

队长：其实那个就是 socket。

队员：啥？你说 Socket 库就是灯泡的插座？

队长：没错，看起来我们还是查查字典比较好。

队员：稍等一下。字典上说，socket 就是凹进去的可以往里面插东西的圆孔。

队长：所以凡是能插东西的孔都可以叫作 socket。

队员：这样啊。

队长：把灯泡插进去，灯就亮了，对吧？

队员：这不是废话嘛……

队长：其实网络通信也是差不多的意思。

队员：怎么讲？

队长：你想象一下，假设我们有一段程序，把它“咔”一下插到一个套接字里，于是我们就可以开始通信了，就跟灯泡插进去就亮一样。

队员：似乎有点牵强吧？

队长：哪有？套接字的背后就是传输数据的通道，这个通道和我们的通信对象是相连接的，就像流过电线的电流一样，数据就在这个通道中流动，所以我们插进去一个程序，就可以和对方通信了，能理解不？

队员：这个通道是什么呢？

队长：探索之旅的时候你是不是睡着了？我们不是说过 TCP 建立连接之后会形成一个像管子一样的东西吗？

队员：哦，好像隐隐约约听到过这个说法……

队长：真的睡着了啊！算了，反正通道就是那根管子一样的东西啦。

队员：明白了，那我就这么理解吧。

队长：也就你一个人不明白吧。

队员：你这种牵强附会的比喻谁能明白啊？

队长：你就扯吧，用 socket 的这个词的又不只有 TCP/IP。

队员：是吗？

队长：是啊，当初施乐（Xerox）公司在开发以太网技术的时候就设计过一个叫 XNS 的协议，那里面就用 socket 这个词了。

队员：施乐不光设计了以太网，还搞出了这个词？

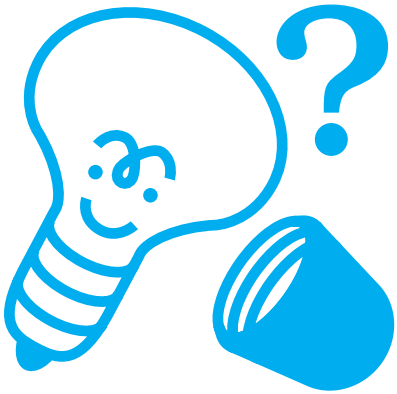
队长：不仅如此，我们现在用的计算机的原型也是施乐设计的。

队员：哦？

队长：当时他们是在研究未来的计算机架构，计算机和以太网只是其中一个环节的产物，话说好像有点跑题了？

队员：是啊，我们说的是 socket 的事吧。话说，这个 XNS 协议的 socket 跟 TCP/IP 的 socket 是一码事吗？

队长：跟 TCP/IP 有点区别，XNS 中



的 socket 差不多相当于 TCP/IP 中的端口号。

队员：那还是不一样的呢。

队长：也不能说不一样，TCP/IP 在创建套接字的时候也是要分配一个端口号的，所以说，套接字和端口号背后的思路其实是有关联的。

队员：哦。

队长：看来你还不懂网络的“心”呀。

队员：“心”又是什么鬼啦？

小测验答案

1. IP 头部（参见【2.5.3】）
2. TCP 头部（参见【2.2.3】）
3. TCP（参见【2.3.3】）
4. ARP（参见【2.5.5】）
5. 滑动窗口方式（参见【2.3.5】）

第3章

从网线到网络设备

——探索集线器、交换机和路由器

热身问答

在开始探索之旅之前，我们准备了一些和本章内容有关的小题目，请大家先试试看。

这些题目是否答得出来并不影响接下来的探索之旅，因此请大家放轻松。

下列说法是正确的（√）还是错误的（×）？

1. 我们现在使用的以太网线（双绞线）是由美国的室内电话线发展而来的。
2. 路由器比交换机问世时间更早。
3. 对于路由器和交换机，如果包在传输过程中发生错误，会直接丢弃错误的包而不会尝试修复。

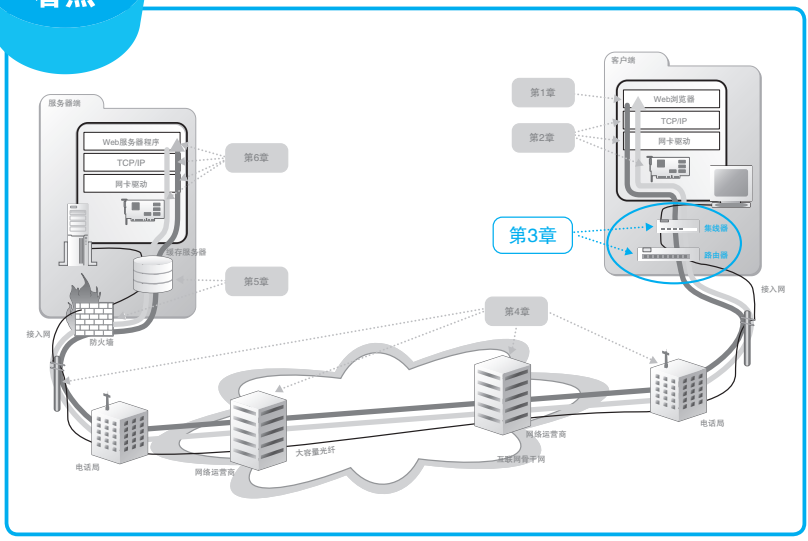
答案

1. √。最早的以太网使用专用同轴网线，后来变成由美国室内电话线改良的版本，原因是它可以兼容电话线的布线工具和材料，比较方便。
2. √。交换机比路由器更加简单，因此可能有人以为交换机应该比路由器出现得更早，其实是路由器先问世的。
3. √。不过操作系统中的网络控制软件（协议栈）会对丢弃的包进行重发，数据不会因此丢失。

前情提要

上一章，我们探索了客户端中的协议栈和网卡，介绍了发送网络包，也就是将网络包转换成电信号通过网线传输出去的过程。本章我们将继续跟着上一章的脚步，看一看通过网线传输出去的包是如何经过集线器、交换机和路由器等网络设备，最终进入互联网的。

探索之旅的看点



(1) 信号在网线和集线器中传输

信号从计算机中流出之后，会在网线中经过集线器等设备前进。此时，信号是如何在网线和集线器传输的，就是我们的第一个看点。信号在传输过程中会衰减，还会受到噪声干扰而失真，如何抑制这些影响是我们的另一个看点。

（2）交换机的包转发操作

交换机的工作方式也是本章看点之一。交换机并不只是简单地让信号流过，而是先接收信号并将其还原为数字信息，然后再重新转换成信号并发送出去的过程。这里我们将详细探索这一过程。

（3）路由器的包转发操作

路由器和交换机一样也负责对包进行转发，但它们的工作方式有一些差异。交换机是基于以太网规格工作的设备，而路由器是基于 IP 工作的，它们之间的差异也是本章看点之一。

（4）路由器的附加功能

位于互联网接入端的路由器通常还会提供一些附加功能，例如将私有地址转换为公有地址的地址转换功能，以及阻止危险网络包的包过滤功能等。本章最后将介绍一下这些功能，这样我们就会对路由器有较全面的认识。