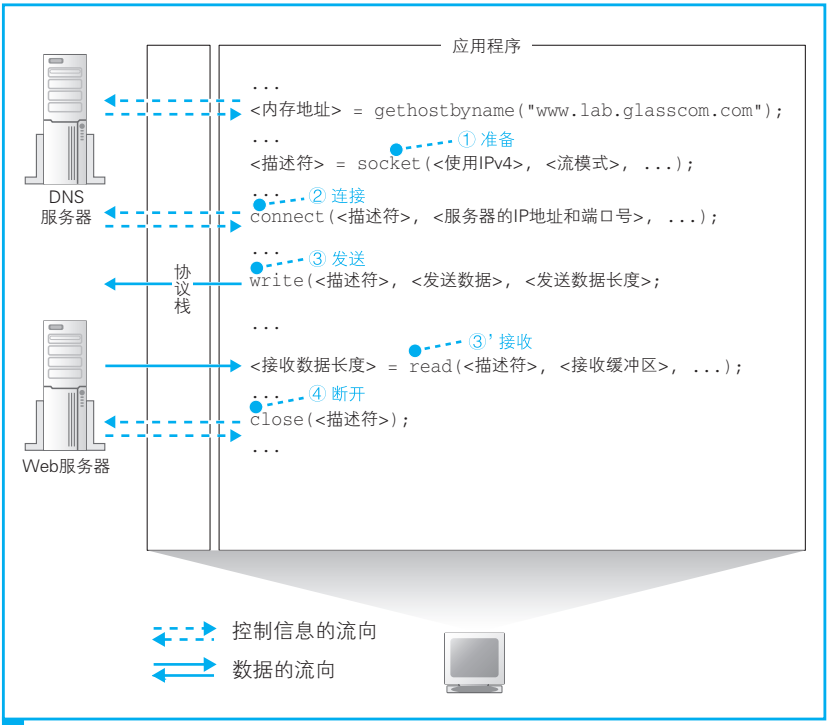


图 1.18　客户端和服务器之间收发数据操作的情形
内部分为创建套接字、连接 Web 服务器、发送数据、接收数据、断开连接几个阶段。

多个套接字，在这样的情况下，我们就需要一种方法来识别出某个特定的套接字，这种方法就是描述符。我们可以将描述符理解成给某个套接字分配的编号。也许光说编号还不够形象，大家可以想象一下在酒店寄存行李时的场景，酒店服务人员会给你一个号码牌，向服务人员出示号码牌，就可以取回自己寄存的行李，描述符的原理和这个差不多。当创建套接字后，我们就可以使用这个套接字来执行收发数据的操作了。这时，只要我们出示描述符，协议栈就能够判断出我们希望用哪一个套接字来连接或者收发数据了。

应用程序是通过“描述符”这一类似号码牌的东西来识别套接字的。

1.4.3　连接阶段：把管道接上去

接下来，我们需要委托协议栈将客户端创建的套接字与服务器那边的套接字连接起来。应用程序通过调用 Socket 库中的名为 connect 的程序组件来完成这一操作。这里的要点是当调用 connect 时，需要指定描述符、服务器 IP 地址和端口号这 3 个参数（图 1.18 ②）。

第 1 个参数，即描述符，就是在创建套接字的时候由协议栈返回的那个描述符。connect 会将应用程序指定的描述符告知协议栈，然后协议栈根据这个描述符来判断到底使用哪一个套接字去和服务器端的套接字进行连接，并执行连接的操作[①]。

第 2 个参数，即服务器 IP 地址，就是通过 DNS 服务器查询得到的我们要访问的服务器的 IP 地址。在 DNS 服务器的部分已经讲过，在进行数据收发操作时，双方必须知道对方的 IP 地址并告知协议栈。这个参数就是那个 IP 地址了。

第 3 个参数，即端口号，这个需要稍微解释一下。可能大家会觉得，IP 地址就像电话号码，只要知道了电话号码不就可以联系到对方了吗？其实，网络通信和电话还是有区别的，我们先来看一看 IP 地址到底能用来干什么。IP 地址是为了区分网络中的各个计算机而分配的数值[②]。因此，只要知道了 IP 地址，我们就可以识别出网络上的某台计算机。但是，连接操作的对象是某个具体的套接字，因此必须要识别到具体的套接字才行，而仅凭 IP 地址是无法做到这一点的。我们打电话的时候，也需要通过“请帮我找一下某某某”这样的方式来找到具体的某个联系人，而端口号就是这样

① 当调用 Socket 库中的程序组件时，应用程序所指定的参数会通过 Socket 库的程序组件传递给协议栈，并由协议栈来实际执行相应的操作。在后面的内容中，这一过程都是相同的，因此不再赘述。

② 准确地说，IP 地址不是分配给每一台设备的，而是分配给设备中安装的网络硬件的。因此，如果一台设备中安装了多个网络硬件，那么就会有多个 IP 地址。

一种方式。当同时指定 IP 地址和端口号时，就可以明确识别出某台具体的计算机上的某个具体的套接字。

也许有人会说：“能不能用前面创建套接字时提到的那个描述符来识别套接字呢?”这种方法其实是行不通的，因为描述符是和委托创建套接字的应用程序进行交互时使用的，并不是用来告诉网络连接的另一方的，因此另一方并不知道这个描述符。同样地，客户端也无法知道服务器上的描述符。因此，客户端也无法通过服务器端的描述符去确定位于服务器上的某一个套接字。所以，我们需要另外一个对客户端也同样适用的机制，而这个机制就是端口号。如果说描述符是用来在一台计算机内部识别套接字的机制，那么端口号就是用来让通信的另一方能够识别出套接字的机制[①]。

既然需要通过端口号来确定连接对象的套接字，那么到底应该使用几号端口呢？网址中好像并没有端口号[②]，也不能像 IP 地址一样去问 DNS 服务器[③]。找了半天也没有任何线索，这可怎么办？其实，这件事情也并没有那么神奇，服务器上所使用的端口号是根据应用的种类事先规定好的，仅此而已。比如 Web 是 80 号端口，电子邮件是 25 号端口[④]。关于端口号，我们将在第 6 章探索服务器内部工作的时候进行介绍，这里大家只要这样记住就行了：只要指定了事先规定好的端口号，就可以连接到相应的服务器程序的套接字。也就是说，浏览器访问 Web 服务器时使用 80 号端口，这是已经规定好的。

可能大家还有一个疑问，既然确定连接对象的套接字需要使用端口号，

---

① 也许会有人说，既然可以用端口号来识别套接字，那为什么还需要描述符呢？要回答这个问题，我们需要对端口号进行更深入的了解，详细请参见 6.1.3 节的讲解。

② 看图 1.1 的前两张图，实际上根据网址的规则，是有用来写端口号的地方的，但实际的网址中很少出现端口号，大部分情况下都省略了。

③ 实际上存在通过 DNS 服务器查询端口号的机制，只能说并没有广泛普及。

④ 端口号的规则是全球统一的，为了避免重复和冲突，端口号和 IP 地址一样都是由 IANA（Internet Assigned Number Authority，互联网编号管理局）这一组织来统一管理的。

那么服务器也得知道客户端的套接字号码才行吧，这个问题是怎么解决的呢？事情是这样的，首先，客户端在创建套接字时，协议栈会为这个套接字随便分配一个端口号[①]。接下来，当协议栈执行连接操作时，会将这个随便分配的端口号通知给服务器。这部分内容我们会在第 2 章探索协议栈内部工作时进行介绍。

说了这么多，总而言之，就是当调用 connect 时，协议栈就会执行连接操作。当连接成功后，协议栈会将对方的 IP 地址和端口号等信息保存在套接字中，这样我们就可以开始收发数据了。

描述符：应用程序用来识别套接字的机制
IP 地址和端口号：客户端和服务器之间用来识别对方套接字的机制

1.4.4 通信阶段：传递消息

当套接字连接起来之后，剩下的事情就简单了。只要将数据送入套接字，数据就会被发送到对方的套接字中。当然，应用程序无法直接控制套接字，因此还是要通过 Socket 库委托协议栈来完成这个操作。这个操作需要使用 write 这个程序组件，具体过程如下。

首先，应用程序需要在内存中准备好要发送的数据。根据用户输入的网址生成的 HTTP 请求消息就是我们要发送的数据。接下来，当调用 write 时，需要指定描述符和发送数据（图 1.18 ③），然后协议栈就会将数据发送到服务器。由于套接字中已经保存了已连接的通信对象的相关信息，所以只要通过描述符指定套接字，就可以识别出通信对象，并向其发送数据。接着，发送数据会通过网络到达我们要访问的服务器。

接下来，服务器执行接收操作，解析收到的数据内容并执行相应的操作，向客户端返回响应消息[②]。

① 在创建套接字时，服务器也可以自行指定端口号，但一般并不常用。
② 详细内容在第 6 章讲解。

当消息返回后，需要执行的是接收消息的操作。接收消息的操作是通过 Socket 库中的 read 程序组件委托协议栈来完成的（图 1.18 ③'）。调用 read 时需要指定用于存放接收到的响应消息的内存地址，这一内存地址称为接收缓冲区。于是，当服务器返回响应消息时，read 就会负责将接收到的响应消息存放到接收缓冲区中。由于接收缓冲区是一块位于应用程序内部的内存空间，因此当消息被存放到接收缓冲区中时，就相当于已经转交给了应用程序。

1.4.5 断开阶段：收发数据结束

当浏览器收到数据之后，收发数据的过程就结束了。接下来，我们需要调用 Socket 库的 close 程序组件进入断开阶段（图 1.18 ④）。最终，连接在套接字之间的管道会被断开，套接字本身也会被删除。

断开的过程如下。Web 使用的 HTTP 协议规定，当 Web 服务器发送完响应消息之后，应该主动执行断开操作[①]，因此 Web 服务器会首先调用 close 来断开连接。断开操作传达到客户端之后，客户端的套接字也会进入断开阶段。接下来，当浏览器调用 read 执行接收数据操作时，read 会告知浏览器收发数据操作已结束，连接已经断开。浏览器得知后，也会调用 close 进入断开阶段。

这就是 HTTP 的工作过程。HTTP 协议将 HTML 文档和图片都作为单独的对象来处理，每获取一次数据，就要执行一次连接、发送请求消息、接收响应消息、断开的过程。因此，如果一个网页中包含很多张图片，就必须重复进行很多次连接、收发数据、断开的操作。对于同一台服务器来说，重复连接和断开显然是效率很低的，因此后来人们又设计出了能够在一次连接中收发多个请求和响应的方法。在 HTTP 版本 1.1 中就可以使用这种方法，在这种情况下，当所有数据都请求完成后，浏览器会主动触发断开连接的操作。

① 根据应用种类不同，客户端和服务器哪一方先执行 close 都有可能。有些应用中是客户端先执行 close，而另外一些应用中则是服务器先执行 close。

本章我们探索了浏览器与 Web 服务器之间收发消息的过程，但实际负责收发消息的是协议栈、网卡驱动和网卡，只有这 3 者相互配合，数据才能够在网络中流动起来。下一章我们将对这一部分进行探索。

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，请确认一下自己的成果吧。

问题

1. http://www.nikkeibp.co.jp/ 中的 http 代表什么意思？
2. 下面两个网址有什么不同？
   a. http://www.nikkeibp.co.jp/sample
   b. http://www.nikkeibp.co.jp/sample/
3. 用来识别连接在互联网上的计算机和服务器的地址叫什么？
4. 根据 Web 服务器的域名来查询 IP 地址时所使用的服务器叫什么？
5. 向 DNS 服务器发送请求消息的程序叫什么？

Column
网络术语其实很简单

怪杰 Resolver[①]

词汇是人类创造的，如果能理解词汇创造者的思路，也就能理解这个词的真正含义。而理解网络中每个词汇的真正含义之后，对网络的理解也会更加深入，反过来也会更加理解设计和创造网络的那些人。各位有不懂的词吗？问问我们的探索队长吧！

探索队员：DNS 客户端的名字叫解析器，这个名字有点怪呢，为什么要叫解析器呢？

探索队长：你知道英语里面 resolve 这个词是什么意思吗？

队员：哎？不知道……等等，我查查字典。嗯……好像是分析、求解、转换之类的意思呢。

队长：那么字典上有没有给出这个词的名词形式？

队员：哦，是 resolver。

队长：那就对了，解析器的英文就是 resolver，明白了不？

队员：可是我还是不明白啊，别卖关子了赶快告诉我好不好？

队长：没卖关子啊，分析、求解、转换，这不就是解析器的工作吗？

队员：是吗？我还是不明白它到底是分析了什么啊……

队长：你回想一下解析器的工作方式，解析器到底是用来干什么的呢？

队员：当需要根据域名查询 IP 地址的时候向 DNS 服务器发送查询消息之类的吧……

队长：发送查询之后呢？

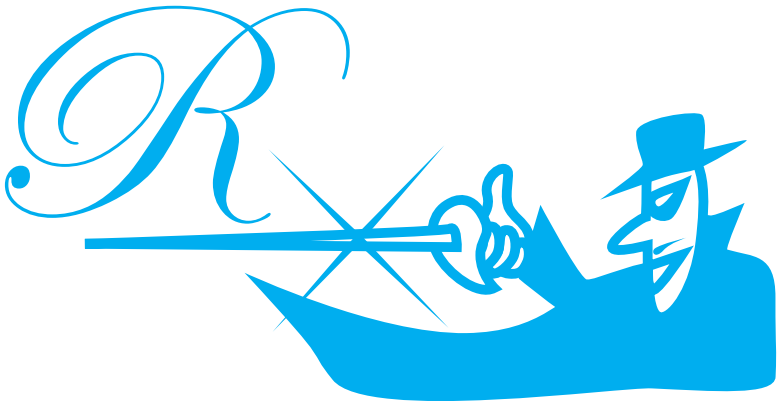
队员：DNS 服务器会返回响应。

队长：接收这个响应然后把答案告诉应用程序，这也是解析器的工作对吧？

队员：是呢。

队长：在调用解析器的应用程序看来，

① “怪杰”一词出自“怪杰佐罗”，日语中“怪杰”与“解决”（resolve）的读音相同，这里是一个谐音的梗。——译者注



我只要给解析器一个域名，解析器就能分析它并给我求出 IP 地址，是不是？

队员：原来如此。

队长：或者我们也可以认为是将域名转换成了 IP 地址。

队员：是啊，这两种情形都符合 resolver 这个词的意思呢。

队长：所以才管它叫 resolver 嘛。

队员：原来是这样啊。那个，我还有一个问题。

队长：什么问题？

队员：地址解析协议（Address Resolution Protocol，ARP）中的 resolution 也是一样的意思吧？

队长：没错。ARP 是根据已知 IP 地址求出 MAC 地址这个答案，从这个角度来看是一个意思。

队员：那么负责执行 ARP 的程序叫什么呢？叫 ARP 解析器吗？

队长：嗯……从意思来看叫 ARP 解析器也没错，不过好像没听说过谁这么叫的。

队员：那到底叫什么呢？

队长：唔……到底叫什么来着？？

队员：其实队长你也不知道吧！

队长：呃……

小测验答案

1. HTTP 协议（参见【1.1.1】）
2. a 中的 sample 代表文件名，b 中的 sample 代表目录名（参见【1.1.3】）
3. IP 地址（参见【1.2.1】）
4. DNS 服务器（参见【1.2.3 和 1.3】）
5. 解析器（参见【1.2.3】）

第2章

用电信号传输 TCP/IP 数据

——探索协议栈和网卡

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中存在不符合国际标准（IEEE802.3/802.2）的部分。
2. TCP/IP 是由 TCP 和 IP 两个协议的名字组合而成的，最开始这两个协议是合在一起的。
3. 网络包通信技术是 20 世纪 60 年代为用计算机进行数据通信而设计出来的。

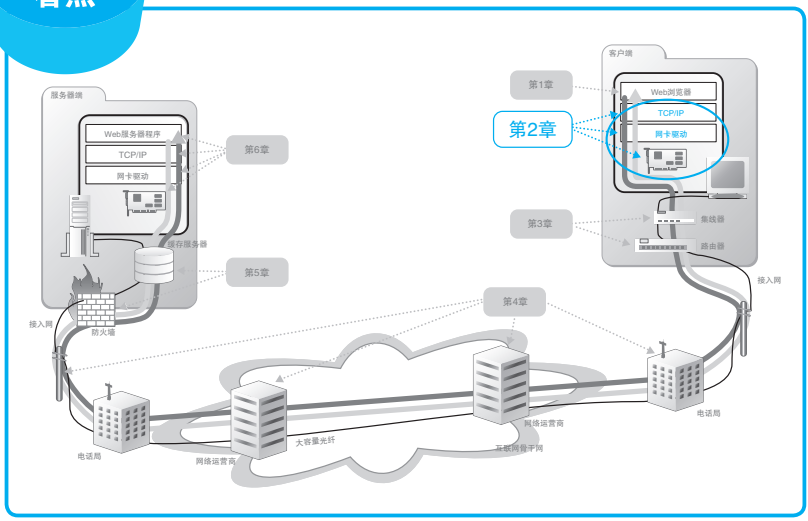
答案

1. √。一般情况下，以太网的头部（网络包开头的控制信息）格式并非遵循国际标准（IEEE802.3/802.2），而是遵循一个更古老的规格（以太网第 2 版，又称 DIX 规格），相对地，国际标准（IEEE802.3/802.2）的头部格式由于长度太长、效率降低而没有普及。
2. √。最早的 TCP/IP 协议原型设计相当于现在的 TCP 和 IP 合在一起的样子，后来才拆分成为 TCP 和 IP 两个协议。
3. √。在网络包出现之前，通信都是像电话一样把线路连接起来进行的。但是，连接线路的通信方式只能和固定的对象进行通信，无法发挥计算机可以处理多种工作的特点。为了解决这个问题，人们设计出了使用网络包来进行通信的方式。

前情提要

第 1 章，我们从解析浏览器中输入的网址开始，探索了生成 HTTP 请求消息、委托操作系统发送消息等步骤。本章，我们将讲解操作系统中的协议栈是如何处理数据发送请求的。第 1 章介绍了发送消息的场景，接下来我们将视角切换到协议栈的内部来继续探索吧。

探索之旅的看点



(1) 创建套接字

从应用程序收到委托后，协议栈通过 TCP 协议收发数据的操作可以分为 4 个阶段。首先是创建套接字，在这个阶段，我们将介绍协议栈的内部结构、套接字的实体，以及创建套接字的操作过程。到这里，大家应该可以对套接字到底是什么样的一个东西有一个比较具体的理解。

(2) 连接服务器

接下来是客户端套接字向服务器套接字进行连接的阶段。我们将介绍“连接”具体是进行怎样的操作，在这个过程中协议栈到底是如何工作的，以及客户端和服务器是如何进行交互的。

(3) 收发数据

两端的套接字完成连接之后，就进入收发消息的阶段了。在这个阶段，协议栈会将从应用程序收到的数据切成小块并发送给服务器，考虑到通信过程中可能会出错导致网络包丢失，协议栈还需要确认切分出的每个包是否已经送达服务器，对于没有送达的包要重新发送一次。这里我们将对收发数据的情形加以说明。

(4) 从服务器断开连接并删除套接字

收发消息的操作全部结束之后，接下来要断开服务器的连接并删除套接字。断开操作的本质是当消息收发完成后客户端和服务器相互进行确认的过程，但这个过程并不只是相互确认并删除套接字那么简单，其中有些地方是很有意思的。

(5) IP 与以太网的包收发操作

在介绍 TCP 协议收发消息的操作之后，我们再来看看实际的网络包是如何进行收发的。协议栈会与网卡进行配合，将数据切分成小块并封装成网络包，再将网络包转换成电信号或者光信号发送出去。介绍完这个过程之后，大家应该就可以对计算机网络功能有一个完整的概念了。

(6) 用 UDP 协议收发数据的操作

TCP 协议有很多方便的功能，比如网络包出错丢失时可以重发，因此很多应用程序都是使用 TCP 协议来收发数据的，但这些方便的功能也有帮倒忙的时候，在这种情况下我们还有另外一种叫 UDP 的协议。这里我们将介绍 UDP 的必要性以及它与 TCP 的差异。

2.1 创建套接字

2.1.1 协议栈的内部结构

本章我们将探索操作系统中的网络控制软件（协议栈）和网络硬件（网卡）是如何将浏览器的消息发送给服务器的。和浏览器不同的是，协议栈的工作我们从表面上是看不见的，可能比较难以想象。因此，在实际探索之前，我们先来对协议栈做个解剖，看看里面到底有些什么。

协议栈的内部如图 2.1 所示，分为几个部分，分别承担不同的功能。这张图中的上下关系是有一定规则的，上面的部分会向下面的部分委派工

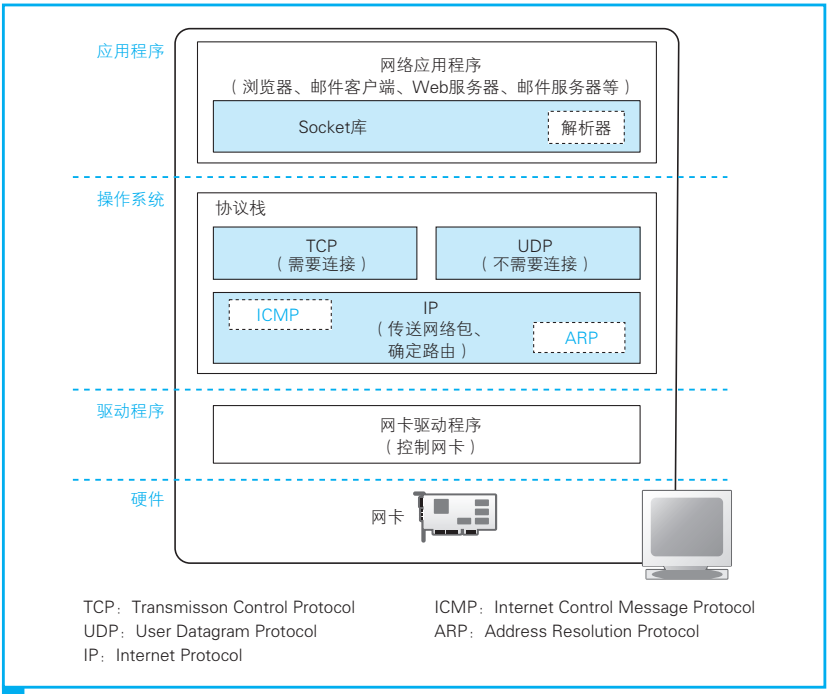


图 2.1 TCP/IP 软件采用分层结构
上层会向下层逐层委派工作。

作，下面的部分接受委派的工作并实际执行，这一点大家在看图时可以参考一下。当然，这一上下关系只是一个总体的规则，其中也有一部分上下关系不明确，或者上下关系相反的情况，所以也不必过于纠结。此外，对于图中的每个部分以及它们的工作方式，本章将按顺序进行介绍，因此对于里面的细节现在看不明白也没关系，只要大体上看出有哪些组成要素就可以了。

下面我们从上到下来看一遍。图中最上面的部分是网络应用程序，也就是浏览器、电子邮件客户端、Web 服务器、电子邮件服务器等程序，它们会将收发数据等工作委派给下层的部分来完成。当然，除了浏览器之外，其他应用程序在网络上收发数据的操作也都是类似上面这样的，也就是说，尽管不同的应用程序收发的数据内容不同，但收发数据的操作是共通的。因此，下面介绍的内容不仅适用于浏览器，也适用于各种应用程序。

应用程序的下面是 Socket 库，其中包括解析器，解析器用来向 DNS 服务器发出查询，它的工作过程我们在第 1 章已经介绍过了。

再下面就是操作系统内部了，其中包括协议栈。协议栈的上半部分有两块，分别是负责用 TCP 协议收发数据的部分和负责用 UDP 协议收发数据的部分，它们会接受应用程序的委托执行收发数据的操作。关于 TCP 和 UDP 我们将在后面讲解，现在大家只要先记住下面这句话就可以了：像浏览器、邮件等一般的应用程序都是使用 TCP 收发数据的，而像 DNS 查询等收发较短的控制数据的时候则使用 UDP。

浏览器、邮件等一般应用程序收发数据时用 TCP；
DNS 查询等收发较短的控制数据时用 UDP。

下面一半是用 IP 协议控制网络包收发操作的部分。在互联网上传送数据时，数据会被切分成一个一个的网络包[①]，而将网络包发送给通信对象的

① 网络包：网络中的数据会被切分成几十字节到几千字节的小块，每一个小数据块被称为一个包。我们会在 2.5.1 节进行讲解。

操作就是由 IP 来负责的。此外，IP 中还包括 ICMP[①] 协议和 ARP[②] 协议。ICMP 用于告知网络包传送过程中产生的错误以及各种控制消息，ARP 用于根据 IP 地址查询相应的以太网 MAC 地址[③]。

IP 下面的网卡驱动程序负责控制网卡硬件，而最下面的网卡则负责完成实际的收发操作，也就是对网线中的信号执行发送和接收的操作。

2.1.2 套接字的实体就是通信控制信息

我们已经了解了协议栈的内部结构，而对于在数据收发中扮演关键角色的套接字，让我们来看一看它具体是个怎样的东西。

在协议栈内部有一块用于存放控制信息的内存空间，这里记录了用于控制通信操作的控制信息，例如通信对象的 IP 地址、端口号、通信操作的进行状态等。本来套接字就只是一个概念而已，并不存在实体，如果一定要赋予它一个实体，我们可以说这些控制信息就是套接字的实体，或者说存放控制信息的内存空间就是套接字的实体。

协议栈在执行操作时需要参阅这些控制信息[④]。例如，在发送数据时，需要看一看套接字中的通信对象 IP 地址和端口号，以便向指定的 IP 地址和端口发送数据。在发送数据之后，协议栈需要等待对方返回收到数据的响应信息，但数据也可能在中途丢失，永远也等不到对方的响应。在这样的情况下，我们不能一直等下去，需要在等待一定时间之后重新发送丢失的数据，这就需要协议栈能够知道执行发送数据操作后过了多长时间。为此，套接字中必须要记录是否已经收到响应，以及发送数据后经过了多长

① ICMP：在 2.5.11 节讲解。

② ARP：在 2.5.5 节讲解。

③ MAC 地址：符合 IEEE 规格的局域网设备都使用同一格式的地址，这种地址被称为 MAC 地址。我们会在 2.5.6 节进行讲解。

④ 这里的控制信息类似于我们在笔记本上记录的日程表和备忘录。我们可以根据笔记本上的日程表和备忘录来决定下一步应该做些什么，同样地，协议栈也是根据这些控制信息来决定下一步操作内容的。