道对等方1存在于该DHT之中。对等方13将先要向对等方1发送一条报文，说“13的前任和后继是什么?”该报文将通过DHT到达对等方12，而它认识到自己将是13的前任，并且它的当前后继即对等方15将成为13的后继。接下来，对等方12向对等方13发送它的前任和后继信息。对等方13此时能够加入DHT，标识它的后继为对等方15，并通知对等方12它应当将其直接后继改为13。

DHT已经在实践中得到了广泛使用。例如，BitTorrent使用Kademlia DHT来产生一个分布式跟踪器。在BitTorrent中，其键是洪流标识符而其值是当前参与洪流的所有对等方的IP地址［Falkner 2007，Neglia 2007］。以这种方式，通过用某洪流标识符来查询DHT，一个新到达的BitTorrent对等方能够确定负责该标识符（即在洪流中跟踪对等方）的对等方。在找到该对等方后，到达的对等方能够向它查询在洪流中的其他对等方列表。

2.7 TCP套接字编程

我们已经看到了一些重要的网络应用，下面就探讨一下网络应用程序是如何实际编写的。在2.1节讲过，典型的网络应用是由一对程序（即客户程序和服务器程序）组成的，它们位于两个不同的端系统中。当运行这两个程序时，创建了一个客户进程和一个服务器进程，同时它们通过从套接字读出和写入数据彼此之间进行通信。开发者创建一个网络应用时，其主要任务就是编写客户程序和服务器程序的代码。

网络应用程序有两类。一类是实现在协议标准（如一个RFC或某种其他标准文档）中所定义的操作；这样的应用程序又称为“开放”的，因为定义其操作的这些规则人所共知。对于这样的实现，客户程序和服务器程序必须遵守由该RFC所规定的规则。例如，某客户程序可能是FTP协议客户端的一种实现，如在2.3节所描述，该协议由RFC 959明确定义；类似地，其服务器程序能够是FTP服务器协议的一种实现，也明确由RFC 959定义。如果一个开发者编写客户程序的代码，另一个开发者编写服务器程序的代码，并且两者都完全遵从该RFC的各种规则，那么这两个程序将能够交互操作。实际上，今天大多数网络应用程序涉及客户和服务器程序间的通信，这些程序都是由不同的程序员单独开发的。例如，与Apache Web服务器通信的Firefox浏览器，或与BitTorrent跟踪器通信的BitTorrent客户。

另一类网络应用程序是专用的网络应用程序。在这种情况下，由客户和服务器程序应用的应用层协议没有公开发布在某RFC中或其他地方。某单独的开发者（或开发团队）创建了客户和服务器程序，并且该开发者用他的代码完全控制程序的功能。但是因为这些代码并没有实现一个开放的协议，其他独立的开发者将不能开发出和该应用程序交互的代码。

在本节中，我们将考察研发一个客户-服务器应用程序中的关键问题，我们将“亲历亲为”来实现一个非常简单的客户-服务器应用程序代码。在研发阶段，开发者必须最先做的一个决定是，应用程序是运行在TCP上还是运行在UDP上。前面讲过TCP是面向连接的，并且为两个端系统之间的数据流动提供可靠的字节流通道。UDP是无连接的，从一个端系统向另一个端系统发送独立的数据分组，不对交付提供任何保证。前面也讲过当客户或服务器程序实现了一个由某RFC定义的协议，它应当使用与该协议关联的周知端口号；与之相反，当研发一个专用应用程序，研发者必须注意避免使用这样的周知端口号。