如果应用程序开发者所编写的代码实现的是一个“周知协议”的服务器端，那么开发者就必须为其分配一个相应的周知端口号。通常，应用程序的客户端让运输层自动地（并且是透明地）分配端口号，而服务器端则分配一个特定的端口号。

通过为 UDP 套接字分配端口号，我们现在能够精确地描述 UDP 的复用与分解了。假定在主机 A 中的一个进程具有 UDP 端口 19157，它要发送一个应用程序数据块给位于主机 B 中的另一进程，该进程具有 UDP 端口 46428。主机 A 中的运输层创建一个运输层报文段，其中包括应用程序数据、源端口号（19157）、目的端口号（46428）和两个其他值（将在后面讨论，它对当前的讨论并不重要）。然后，运输层将得到的报文段传递到网络层。网络层将该报文段封装到一个 IP 数据报中，并尽力而为地将报文段交付给接收主机。如果该报文段到达接收主机 B，接收主机运输层就检查该报文段中的目的端口号（46428）并将该报文段交付给端口号 46428 所标识的套接字。值得注意的是，主机 B 能够运行多个进程，每个进程有自己的 UDP 套接字及相应的端口号。值得注意的是，主机 B 可能运行多个进程，每个进程都具有其自己的 UDP 套接字和相联系的端口号。当从网络到达 UDP 报文段时，主机 B 通过检查该报文段中的目的端口号，将每个报文段定向（分解）到相应的套接字。

注意到下述事实是重要的：一个 UDP 套接字是由一个二元组来全面标识的，该二元组包含一个目的 IP 地址和一个目的端口号。因此，如果两个 UDP 报文段有不同的源 IP 地址和/或源端口号，但具有相同的目的 IP 地址和目的端口号，那么这两个报文段将通过相同的目的套接字被定向到相同的目的进程。

你也许现在想知道，源端口号的用途是什么呢？如图 3-4 所示，在 A 到 B 的报文段中，源端口号用作“返回地址”的一部分，即当 B 需要回发一个报文段给 A 时，B 到 A 的报文段中的目的端口号便从 A 到 B 的报文段中的源端口号中取值。（完整的返回地址是 A 的 IP 地址和源端口号。）举一个例子，回想 2.7 节学习过的那个 UDP 服务器程序。在 UDPServer. py 中，服务器使用 recvform() 方法从其自客户接收到的报文段中提取出客户端（源）端口号，然后，它将所提取的源端口号作为目的端口号，向客户发送一个新的报文段。

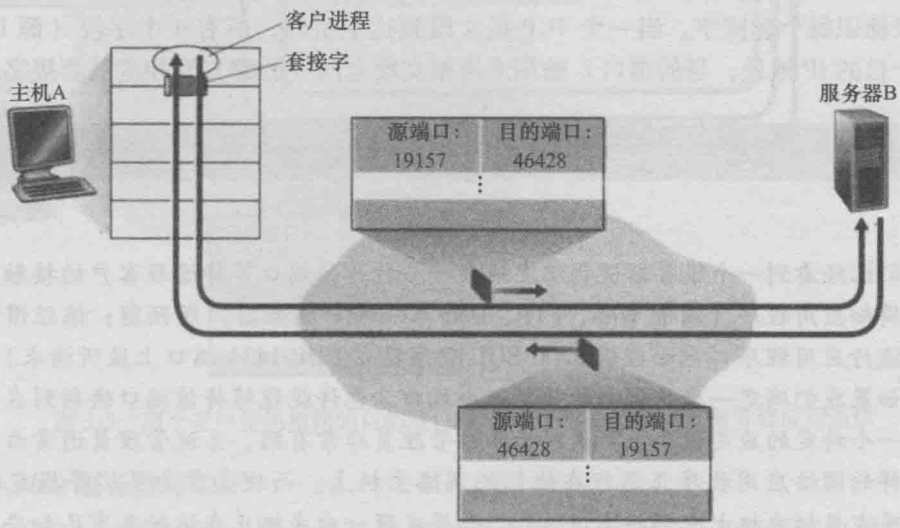

图 3-4 源端口号与目的端口号的反转