8.2.3 用户层通信软件

在多计算机中，不同CPU上的进程通过互相发送消息实现通信。在最简单的情况下，这种消息传送是暴露给用户进程的。换句话说，操作系统提供了一种发送和接收消息的途径，而库过程使得这些低层的调用对用户进程可用。在较复杂的情形下，通过使得远程通信看起来像过程调用的办法，将实际的消息传递对用户隐藏起来。下面将讨论这两种方法。

1. 发送和接收

在最简化的的情形下，所提供的通信服务可以减少到两个（库）调用，一个用于发送消息，另一个用于接收消息。发送一条消息的调用可能是

```
send(dest, &mptr);
```

而接收消息的调用可能是

```
receive(addr, &mptr);
```

前者把由mptr参数所指向的消息发送给由dest参数所标识的进程，并且引起对调用者的阻塞，直到该消息被发出。后者引起对调用者的阻塞，直到消息到达。该消息到达后，被复制到由mptr参数所指向的缓冲区，并且撤销对调用者的阻塞。addr参数指定了接收者要监听的地址。这两个过程及其参数有许多可能的变种。

一个问题是如何编址。由于多计算机是静态的，CPU数目是固定的，所以处理编址问题的最便利的办法是使addr由两部分的地址组成，其中一部分是CPU编号，另一部分是在这个已编址的CPU上的一个进程或端口的编号。在这种方式中，每个CPU可以管理自己的地址而不会有潜在的冲突。

2. 阻塞调用和非阻塞调用

上面所叙述的调用是阻塞调用（有时称为同步调用）。当一个进程调用send时，它指定一个目标以及用以发送消息到该目标的一个缓冲区。当消息发送时，发送进程被阻塞（挂起）。在消息已经完全发送出去之前，不会执行跟随在调用send后面的指令，如图8-19a所示。类似地，在消息真正接收并且放入由参数指定的消息缓冲区之前，对receive的调用也不会把控制返回。在receive中进程保持挂起状态，直到消息到达为止，这甚至有可能等待若干小时。在有些系统中，接收者可以指定希望从谁处接收消息，在这种情况下接收者就保持阻塞状态，直到来自那个发送者的消息到达为止。

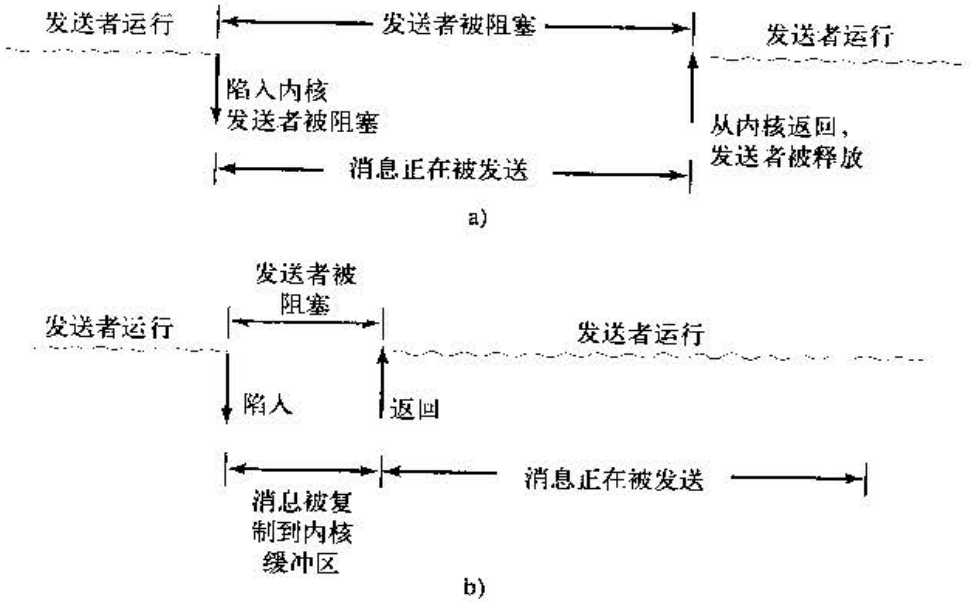

图8-19 a) 一个阻塞的send调用；b) 一个非阻塞的send调用

相对于阻塞调用的另一种方式是非阻塞调用（有时称为异步调用）。如果send是非阻塞的，在消息发出之前，它立即将控制返回给调用者。这种机制的优点是发送进程可以继续运算，与消息传送并行，而不是让CPU空闲（假设没有其他可运行的进程）。通常是由系统设计者做出在阻塞原语和非阻塞原语