可靠的传送（如果底层的硬件会丢包）、多播（将包发送到多于一个的目的地）、压缩/解压缩、加密/解密以及在多进程系统中处理安全事务等。但是，有两个CPU则意味着它们必须同步，以避免竞争条件的发生，这将增加额外的开销，并且对于操作系统来说意味着要承担更多的工作。

8.2.2 低层通信软件

在多计算机系统中高性能通信的敌人是对包的过度复制。在最好的情形下，在源节点会有从RAM到接口板的一次复制，从源接口板到目的接口板的一次复制（如果在路径上没有存储和转发发生）以及从目的接口板再到目的地RAM的一次复制，这样一共有三次复制。但是，在许多系统中情况要糟糕得多。特别是，如果接口板被映射到内核虚拟地址空间中而不是用户虚拟地址空间的话，用户进程只能通过发出一个陷入到内核的系统调用的方式来发送包。内核会同时在输入和输出时把包复制到自己的存储空间去，从而在传送到网络上时避免出现缺页异常（page fault）。同样，接收包的内核在有机会检查包之前，可能也不知道应该把进来的包放置到哪里。上述五个复制步骤如图8-18所示。

如果说进出RAM的复制是性能瓶颈，那么进出内核的额外复制会将端到端的延迟加倍，并把吞吐量（throughput）降低一半。为了避免这种对性能的影响，不少多计算机把接口板映射到用户空间，并允许用户进程直接把包送到卡上，而不需要内核的参与。尽管这种处理确实改善了性能，但却带来了两个问题。

首先，如果在节点上有若干个进程运行而且需要访问网络以发送包，该怎么办？哪一个进程应该在其地址空间中获得接口板呢？映射拥有一个系统调用将接口板映射进出一个虚拟地址空间，其代价是很高的，但是，如果只有一个进程获得了卡，那么其他进程该如何发送包呢？如果网卡被映射进了进程A的虚拟地址空间，而所到达的包却是进程B的，又该怎么办？尤其是，如果A和B属于不同的所有者，其中任何一方都不打算协助另一方，又怎么办？

一个解决方案是，把接口板映射到所有需要它的进程中去，但是这样做就需要有一个机制用以避免竞争。例如，如果A申明接口板上的一个缓冲区，而由于时间片，B开始运行并且申明同一个缓冲区，那么就会发生灾难。需要有某种同步机制，但是那些诸如互斥信号量（mutex）一类的机制需要在进程会彼此协作的前提下才能工作。在有多个用户的分时环境下，所有的用户都希望其工作尽快完成，某个用户也许会锁住与接口板有关的互斥信号量而不肯释放。从这里得到的结论是，对于将接口板映射到用户空间的方案，只有在每个节点上只有一个用户进程运行时才能够发挥作用，否则必须设置专门的预防机制（例如，对不同的进程可以把接口板上RAM的不同部分映射到各自的地址空间）。

第二个问题是，内核本身会经常需要访问互连网络，例如，访问远程节点上的文件系统。如果考虑让内核与任何用户共享同一块接口板，即便是基于分时方式，也不是一个好主意。假设当板被映射到用户空间，收到了一个内核的包，那么怎么办？或者若某个用户进程向一个伪装成内核的远程机器发送了一个包，又该怎么办？结论是，最简单的设计是使用两块网络接口板，一块映射到用户空间供应用程序使用，另一块映射到内核空间供操作系统使用。许多多计算机就正是这样做的。

节点至网络接口通信

下一个问题是如何将包送到接口板上。最快的方法是使用板上的DMA芯片直接将它们从RAM复制到板上。这种方式的问题是，DMA使用物理地址而不是虚拟地址，并且独立于CPU运行。首先，尽管一个用户进程肯定知道它打算发送的任何包所在的虚拟地址，但它通常不知道有关的物理地址。设计一个系统调用进行虚拟地址到物理地址的映射是不可取的，因为把接口板放到用户空间的首要原因就是为了避免不得不为每个要发送的包进行一次系统调用。

另外，如果操作系统决定替换一个页面，而DMA芯片正在从该页面复制一个包，就会传送错误的数据。然而更加糟糕的是，如果操作系统在替换某一个页面的同时DMA芯片正在把一个包复制进该页面，结果不仅进来的包会丢失，无辜的存储器页面也会被毁坏。

为了以避免上述问题，可采用一类将页面钉住和释放的系统调用，把有关页面标记成暂时不可交换的。但是不仅需要有一个系统调用钉住含有每个输出包的页面，还要有另一个系统调用进行释放工作，这样做的代价太大。如果包很小，比如64字节或更小，就不能忍受钉住和释放每个缓冲区的开销。对于大的包，比如说1KB或更大，也许会容忍相关开销。对于大小在这两者之间的包，就要取决于硬件的具体情况了。除了会对性能带来影响，钉住和释放页面将会增加软件的复杂性。