的2.5%的空闲时间，5ms的间隙必须每200ms出现一次，这就是间隙为什么没有在图7-15中出现的原因。

一个有趣的问题是RMS为什么会失败。根本上，使用静态优先级只有在CPU的利用率不太高的时候才能工作。Liu和Layland（1973）证明了对于任何周期性进程系统，如果

$$\sum_{i=1}^{m}\frac{C_i}{P_i}\leqslant m(2^{1/m}-1)$$

那么就可以保证RMS工作。对于m = 3、4、5、10、20和100，最大允许利用率为0.780、0.757、0.743、0.718、0.705和0.696。随着$m\to\infty$，最大利用率逼近ln 2。换句话说，Liu和Layland证明了，对于三个进程，如果CPU利用率等于或小于0.780，那么RMS总是可以工作的。在第一个例子中，CPU利用率为0.808而RMS工作正常，但那只不过是幸运罢了。对于不同的周期和运行时间，利用率为0.808很可能会失败。在第二个例子中，CPU利用率如此之高（0.975），根本不存在RMS能够工作的希望。

与此相对照，EDF对于任意一组可调度的进程总是可以工作的，它可以达到100%的CPU利用率，付出的代价是更为复杂的算法。因而，在一个实际的视频服务器中，如果CPU利用率低于RMS限度，可以使用RMS，否则，应该选择EDF。

7.6 多媒体文件系统范型

至此我们已经讨论了多媒体系统中的进程调度，下面继续我们的研究，看一看多媒体文件系统。这样的文件系统使用了与传统文件系统不同的范型。我们首先回顾传统的文件I/O，然后将注意力转向多媒体文件服务器是如何组织的。进程要访问一个文件时，首先要发出open系统调用。如果该调用成功，则调用者被给予某种令牌以便在未来的调用中使用，该令牌在UNIX中被称为文件描述符，在Windows中被称为句柄。这时，进程可以发出read系统调用，提供令牌、缓冲区地址和字节计数作为参数。操作系统则在缓冲区中返回请求的数据。以后还可以发出另外的read调用，直到进程结束，在进程结束时它将调用close以关闭文件并返回其资源。

由于实时行为的需要，这一模型对于多媒体并不能很好地工作。在显示来自远程视频服务器的多媒体文件时，该模型的工作尤为拙劣。第一个问题是用户必须以相当精确的时间间隔进行read调用。第二个问题是视频服务器必须能够没有延迟地提供数据块，当请求没有计划地到来并且预先没有保留资源时，做到这一点是十分困难的。

为解决这些问题，多媒体文件服务器使用了一个完全不同的范型：像录像机（Video Cassette Recorder，VCR）一样工作。为了读取一个多媒体文件，用户进程发出start系统调用，指定要读的文件和各种其他参数，例如，要使用哪些音频和字幕轨迹。接着，视频服务器开始以必要的速率送出帧。然后用户进程以帧进来的速率对它们进行处理。如果用户对所看的电影感到厌烦，那么发出stop系统调用可以将数据流终止。具有这种数据流模型的文件服务器通常被称为推送型服务器（push server），因为它将数据推送给用户；与此相对照的是传统的拉取型服务器（pull server），用户不得不通过重复地调用read一块接一块地取得数据，每调用一次可以拉取出一块数据。这两个模型之间的区别如图7-16所示。

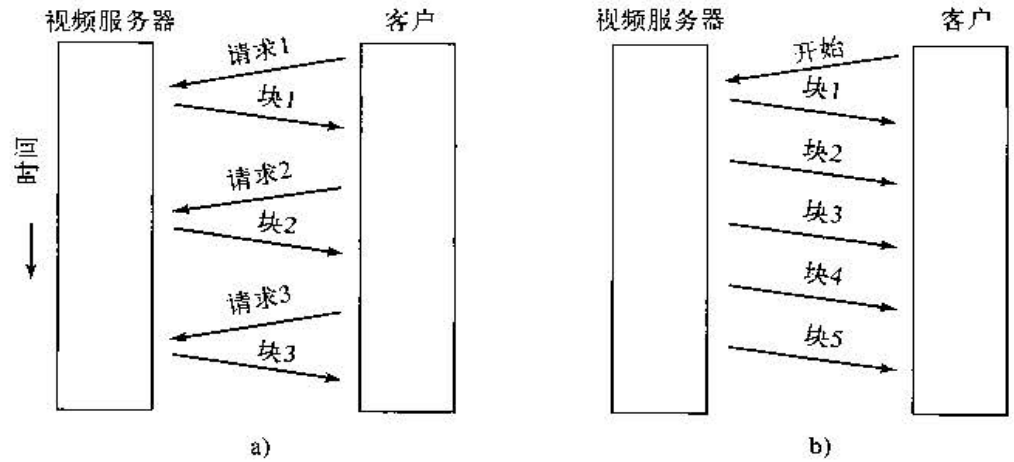

图7-16 a) 拉取型服务器；b) 推送型服务器