7.6.1 VCR控制功能

大多数视频服务器也实现了标准的VCR控制功能，包括暂停、快进和倒带。暂停是相当简单的。用户发送一个消息给视频服务器，告诉它停止。视频服务器此时要做的全部事情是记住下一次要送出的是哪一帧。当用户要求服务器恢复播放时，服务器只要从它停止的地方继续就可以了。

然而，这里存在着一个复杂因素。为了获得可接受的性能，服务器应该为每个流出的数据流保留诸如磁盘带宽和内存缓冲区等资源。当电影暂停时继续占用这些资源将造成浪费，特别是如果用户打算到厨房中找到一块冷冻的比萨饼（或许是特大号的）、用微波炉烹调并且美餐一顿的时候。当然，在暂停的时候可以很容易地将资源释放，但是这引入了风险：当用户试图恢复播放的时候，有可能无法重新获得这些资源。

真正的倒带实际上非常简单，没有任何复杂性。服务器要做的全部事情是注意到下一次要送出的帧是第0帧。还有比这更容易的吗？然而，快进和快倒（也就是在倒带的同时播放）就难处理多了。如果没有压缩，那么以10倍的速度前进的一种方法是每10帧只显示一帧，以20倍的速度前进则要求每20帧显示一帧。实际上，在不存在压缩的情况下，以任意速度前进和后退都是十分容易的。要以正常速度的k倍运行，只要每k帧显示一帧就可以了。要以正常速度的k倍后退，只要沿另一个方向做相同的事情就可以了。这一方法在推送型服务器和拉取型服务器上工作得同样好。

压缩则使快进和快倒复杂起来。对于便携式摄像机的DV磁带，由于其每一帧都是独立于其他帧而压缩的，所以只要能够快速地找到所需要的帧，使用这一策略还是有可能的。由于视其内容不同每一帧的压缩量也有所不同，所以每一帧具有不同的大小，因而在文件中向前跳过k帧并不能通过数字计算来完成。此外，音频压缩是独立于视频压缩的，所以对于在高速模式中显示的每一视频帧，还必须找到正确的音频帧（除非在高于正常速度播放时将声音关闭）。因此，对一个DV文件进行快进操作需要有一个索引，该索引可以使帧的查找快速地实现，但是至少在理论上这样做是可行的。

对于MPEG，由于使用I帧、P帧和B帧，这一方案即使在理论上也是不能工作的。向前跳过k帧（就算假设能这样做）可能落在一个P帧上，而这个P帧则基于刚刚跳过的一个I帧。没有基本帧，只有从基本帧发生的增量变化（这正是P帧所包含的）是无用的。MPEG要求按顺序播放文件。

攻克这一难题的另一个方法是实际尝试以10倍的速度顺序地播放文件。然而，这样做就要求以10倍的速度将数据拉出磁盘。此时，服务器可能试图将帧解压缩（这是正常情况下服务器不需要做的事情），判定需要哪一帧，然后每隔10帧重新压缩成一个I帧。然而，这样做给服务器增加了沉重的负担。这一方法还要求服务器了解压缩格式，正常情况下服务器不必了解这些东西。

作为替代，可以通过网络实际发送所有的数据给用户，并在用户端选出正确的帧，这样做就要求网络以10倍的速度运行，这或许是可行的，但是在这么高的速度下正常操作肯定不是一件容易的事情。

总而言之，不存在容易的方法。惟一可行的策略要求预先规划。可以做的事情是建立一个特殊的文件，包含每隔10帧中的一帧，并且将该文件以通常的MPEG算法进行压缩。这个文件正是在图7-3中注为“快进”的那个文件。要切换到快进模式，服务器必须判定在快进文件中用户当前所在的位置。例如，如果当前帧是48 210并且快进文件以10倍的速度运行，那么服务器在快进文件中必须定位到4821帧并且在此处以正常速度开始播放。当然，这一帧可能是P帧或B帧，但是客户端的解码进程可以简单地跳过若干帧直到看见一个I帧。利用特别准备的快倒文件，可以用类似的方法实现快倒。

当用户切换回到正常速度时，必须使用相反的技巧。如果在快进文件中当前帧是5734，服务器只要切换回到常规文件并且从57 340帧处继续播放。同样，如果这一帧不是一个I帧，客户端的解码进程必须忽略所有的帧直到看见一个I帧。

尽管有了这两个额外的文件可以做这些工作，这一方案还是有某些缺点。首先，需要某些额外的磁盘空间来存放额外的文件。其次，快进和倒带只能以对应于特别文件的速度进行。第三，在常规文件、快进文件和快倒文件之间来回切换需要额外的复杂算法。

7.6.2 近似视频点播

有k个用户取得相同的电影和这些用户取得k部不同的电影在本质上给服务器施加了相同的工作量。然而，通过对模型做一个小小的修改，就可能获得巨大的性能改进。视频点播面临的问题是用户可能在