客户应用缓存中。一旦在缓存中字节数量超过了预先设定的阈值，该客户应用程序开始播放，具体而言，它周期性地从客户应用缓存中抓取视频帧，对帧解压缩并在用户屏幕上显示它们。

我们在第3章学习过，当通过TCP传输一个文件时，由于TCP的拥塞控制机制，服务器到客户的传输速率可能变化很大。特别是，传输速率以与TCP拥塞控制相关联的“锯齿”形（例如，图3-53）变化并非罕见。此外，分组也能由于重传机制而被大大延迟。因为TCP的这些特点，在20世纪90年代大多数人关于会话式的看法是流式视频将不可能在TCP上很好地工作。然而，随着时间的推移，流式视频系统的设计者知道了当使用了客户缓存和预取（在下面讨论）技术时，TCP的拥塞控制和可靠数据传输机制并不一定会妨碍连续播放。

在TCP上使用HTTP也使得视频穿越防火墙和NAT（它们常常被配置为阻挡UDP流量但允许大部分HTTP流量通过）更为容易。HTTP流消除了因需要媒体控制服务器（如RTSP服务器）带来的不便，减少了在因特网上大规模部署的成本。由于所有这些优点，今天的大多数流式视频应用（包括YouTube和Netflix）都使用HTTP流（在TCP上）作为它的底层流式协议。

1. 预取视频

我们刚才学习了客户端缓存可用于缓解变化的端到端时延和变化的可用带宽的影响。在前面图7-1的例子中，服务器以视频播放的速率传输。然而，对于流式存储视频，客户能够尝试以高于消耗速率的速率下载视频，因此**预取**（prefetching）将来会被消耗的视频帧。该预取的视频当然存储在客户应用缓存中。这样的预取自然伴随TCP流出现，因为TCP拥塞避免机制将试图使用服务器和客户之间的所有可用带宽。

为了深入洞察预取技术，我们来举个简单的例子。假设视频消耗速率是1Mbps，而网络从服务器到客户能够以恒定的1.5Mbps速率交付视频。客户则不仅能够以非常小的播放时延播放该视频，而且还能够以每秒500Kb的量增加缓存的视频数据。以这种方式，如果后来该客户在一段短暂时间内以小于1Mbps的速率接收数据，该客户由于在其缓存中的储备将能够继续提供连续的播放。[Wang 2008]显示了当平均TCP吞吐量大致为媒体比特率的两倍时，TCP流导致最小的饥饿和低缓存时延。

2. 客户应用缓存和TCP缓存

图7-2说明了客户和服务器之间HTTP流的交互。在服务器侧，视频文件中的白色部分已经通过服务器的套接字进行发送，而黑色部分是留下待发送的部分。在“通过套接字的门传送”之后，放置在TCP发送缓存中的字节在被传输进因特网之前如第3章所描述。在图7-2中，因为TCP发送缓存显示为满，服务器瞬间防止从视频文件发送更多的字节到套接字。在客户侧，客户应用程序（媒体播放器）从TCP接收缓存（通过其客户套接字）读出字节并将字节放入客户应用缓存中。与此同时，客户应用程序周期性地从客户应用缓存中抓取视频帧，解压缩并显示在用户屏幕上。注意到如果客户应用缓存大于该视频文件，则从服务器存储器到客户应用缓存移动字节的整个过程等价于普通文件经HTTP的下载过程，即客户直接将视频用TCP允许的尽可能快的速率从服务器中拉出来。

现在考虑在流播放期间当用户暂停视频时将发生的现象。在暂停期间，比特未从客户应用缓存中删除，甚至比特继续从服务器进入缓存。如果客户应用缓存是有限的，它可能