报文段。当将该机制与选择重传机制结合起来使用时（即跳过重传那些已被接收方选择性地确认过的报文段），TCP 看起来就很像我们通常的 SR 协议。因此，TCP 的差错恢复机制也许最好被分类为 GBN 协议与 SR 协议的混合体。

3.5.5 流量控制

前面讲过，一条 TCP 连接每一侧主机都为该连接设置了接收缓存。当该 TCP 连接收到正确、按序的字节后，它就将数据放入接收缓存。相关联的应用进程会从该缓存中读取数据，但不必是数据刚一到达就立即读取。事实上，接收方应用也许正忙于其他任务，甚至要过很长时间后才去读取该数据。如果某应用程序读取数据时相对缓慢，而发送方发送得太多、太快，发送的数据就会很容易地使该连接的接收缓存溢出。

TCP 为它的应用程序提供了**流量控制服务**（flow-control service）以消除发送方使接收方缓存溢出的可能性。流量控制因此是一个速度匹配服务，即发送方的发送速率与接收方应用程序的读取速率相匹配。前面提到过，TCP 发送方也可能因为 IP 网络的拥塞而被遏制；这种形式的发送方的控制被称为**拥塞控制**（congestion control），我们将在 3.6 节和 3.7 节详细地讨论这个主题。即使流量控制和拥塞控制采取的动作非常相似（对发送方的遏制），但是它们显然是针对完全不同的原因而采取的措施。不幸的是，许多作者把这两个术语混用，理解力强的读者会明智地区分这两种情况。现在我们来讨论 TCP 如何提供流量控制服务的。为了能从整体上看问题，我们在本节都假设 TCP 是这样实现的，即 TCP 接收方丢弃失序的报文段。

TCP 通过让*发送方*维护一个称为**接收窗口**（receive window）的变量来提供流量控制。通俗地说，接收窗口用于给发送方一个指示——该接收方还有多少可用的缓存空间。因为 TCP 是全双工通信，在连接两端的发送方都各自维护一个接收窗口。我们在文件传输的情况下研究接收窗口。假设主机 A 通过一条 TCP 连接向主机 B 发送一个大文件。主机 B 为该连接分配了一个接收缓存，并用 RcvBuffer 来表示其大小。主机 B 上的应用进程不时地从该缓存中读取数据。我们定义以下变量：

- LastByteRead：主机 B 上的应用进程从缓存读出的数据流的最后一个字节的编号。
- LastByteRcvd：从网络中到达的并且已放入主机 B 接收缓存中的数据流的最后一个字节的编号。

由于 TCP 不允许已分配的缓存溢出，下式必须成立：

$$\text{LastByteRcvd} - \text{LastByteRead} \leqslant \text{RcvBuffer}$$

接收窗口用 rwnd 表示，根据缓存可用空间的数量来设置：

$$\text{rwnd} = \text{RcvBuffer} - [\text{LastByteRcvd} - \text{LastByteRead}]$$

由于该空间是随着时间变化的，所以 rwnd 是动态的。图 3-38 对变量 rwnd 进行了图示。

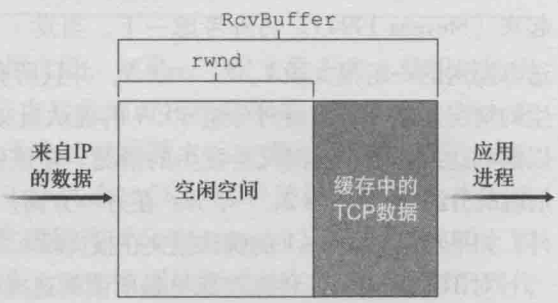

图 3-38 接收窗口（rwnd）和接收缓存（RcvBuffer）

连接是如何使用变量 rwnd 提供流量控制服务的呢？主机 B 通过把当前的 rwnd 值放入它发给主机 A 的报文段接收窗口字段中，通知主机 A 它在该连接的缓存中还有多少可用空间。开始时，主机 B 设定 rwnd = RcvBuffer。注意到为了实现这