概述了 TCP 拥塞控制后，现在是我们考虑广受赞誉的 **TCP 拥塞控制算法**（TCP congestion control algorithm）细节的时候了，该算法首先在［Jacobson 1988］中描述并且在［RFC 5681］中标准化。该算法包括 3 个主要部分：①慢启动；②拥塞避免；③快速恢复。慢启动和拥塞避免是 TCP 的强制部分，两者的差异在于对收到的 ACK 做出反应时增加 cwnd 长度的方式。我们很快将会看到慢启动比拥塞避免能更快地增加 cwnd 的长度（不要被名称所迷惑！）。快速恢复是推荐部分，对 TCP 发送方并非是必需的。

1. 慢启动

当一条 TCP 连接开始时，cwnd 的值通常初始置为一个 MSS 的较小值［RFC 3390］，这就使得初始发送速率大约为 MSS/RTT。例如，如果 MSS = 500 字节且 RTT = 200ms，则得到的初始发送速率大约只有 20kbps。由于对 TCP 发送方而言，可用带宽可能比 MSS/RTT 大得多，TCP 发送方希望迅速找到可用带宽的数量。因此，在**慢启动**（slow-start）状态，cwnd 的值以 1 个 MSS 开始并且每当传输的报文段首次被确认就增加 1 个 MSS。在图 3-51所示的例子中，TCP 向网络发送第一个报文段并等待一个确认。当该确认到达时，TCP 发送方将拥塞窗口增加一个 MSS，并发送出两个最大长度的报文段。这两个报文段被确认，则发送方对每个确认报文段将拥塞窗口增加一个 MSS，使得拥塞窗口变为 4 个 MSS，并这样下去。这一过程每过一个 RTT，发送速率就翻番。因此，TCP 发送速率起始慢，但在慢启动阶段以指数增长。

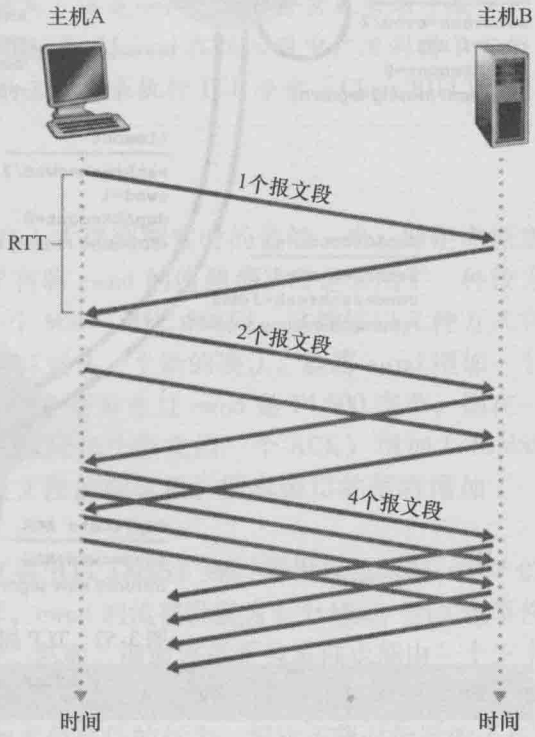

图 3-51 TCP 慢启动

但是，何时结束这种指数增长呢？慢启动对这个问题提供了几种答案。首先，如果存在一个由超时指示的丢包事件（即拥塞），TCP 发送方将 cwnd 设置为 1 并重新开始慢启动过程。它还将第二个状态变量的值 ssthresh（“慢启动阈值”的速记）设置为 cwnd/2，即当检测到拥塞时将 ssthresh 置为拥塞窗口值的一半。慢启动结束的第二种方式是直接与ssthresh的值相关联。因为当检测到拥塞时 ssthresh 设为 cwnd 的值一半，当到达或超过 ssthresh的值时，继续使 cwnd 翻番可能有些鲁莽。因此，当 cwnd 的值等于 ssthresh 时，结束慢启动并且 TCP 转移到拥塞避免模式。我们将会看到，当进入拥塞避免模式时，TCP 更为谨慎地增加 cwnd。最后一种结束慢启动的方式是，如果检测到 3 个冗余 ACK，这时 TCP 执行一种快速重传（参见 3.5.4 节）并进入快速恢复状态，后面将讨论相关内容。慢启动中的 TCP 行为总结在图 3-52 中的 TCP 拥塞控制的 FSM 描述中。慢启动算法最早源于［Jacobson 1988］；在［Jain 1986］中独立地提出了一种类似于慢启动的方法。