报文段，并把新的过期时间设置为1.5秒。如果1.5秒后定时器又过期了，则TCP将再次重传该报文段，并把过期时间设置为3.0秒。因此，超时间隔在每次重传后会呈指数型增长。然而，每当定时器在另两个事件（即收到上层应用的数据和收到ACK）中的任意一个启动时，TimeoutInterval由最近的EstimatedRTT值与DevRTT值推算得到。

这种修改提供了一个形式受限的拥塞控制。（更复杂的TCP拥塞控制形式将在3.7节中学习。）定时器过期很可能是由网络拥塞引起的，即太多的分组到达源与目的地之间路径上的一台（或多台）路由器的队列中，造成分组丢失或长时间的排队时延。在拥塞的时候，如果源持续重传分组，会使拥塞更加严重。相反，TCP使用更文雅的方式，每个发送方的重传都是经过越来越长的时间间隔后进行的。当我们在第5章学习CSMA/CD时，将看到以太网采用了类似的思路。

3. 快速重传

超时触发重传存在的问题之一是超时周期可能相对较长。当一个报文段丢失时，这种长超时周期迫使发送方延迟重传丢失的分组，因而增加了端到端时延。幸运的是，发送方通常可在超时事件发生之前通过注意所谓冗余ACK来较好地检测到丢包情况。**冗余ACK**（duplicate ACK）就是再次确认某个报文段的ACK，而发送方先前已经收到对该报文段的确认。要理解发送方对冗余ACK的响应，我们必须首先看一下接收方为什么会发送冗余ACK。表3-2总结了TCP接收方的ACK生成策略［RFC 5681］。当TCP接收方收到一个具有这样序号的报文段时，即其序号大于下一个所期望的、按序的报文段，它检测到了数据流中的一个间隔，这就是说有报文段丢失。这个间隔可能是由于在网络中报文段丢失或重新排序造成的。因为TCP不使用否定确认，所以接收方不能向发送方发回一个显式的否定确认。相反，它只是对已经接收到的最后一个按序字节数据进行重复确认（即产生一个冗余ACK）即可。（注意到在表3-2中允许接收方不丢弃失序报文段。）

表3-2 产生TCP ACK的建议［RFC 5681］

事件	TCP接收方动作
具有所期望序号的按序报文段到达。所有在期望序号及以前的数据都已经被确认	延迟的ACK。对另一个按序报文段的到达最多等待500ms。如果下一个按序报文段在这个时间间隔内没有到达，则发送一个ACK
具有所期望序号的按序报文段到达。另一个按序报文段等待ACK传输	立即发送单个累积ACK，以确认两个按序报文段
比期望序号大的失序报文段到达。检测出间隔	立即发送冗余ACK，指示下一个期待字节的序号（其为间隔的低端的序号）
能部分或完全填充接收数据间隔的报文段到达	倘若该报文段起始于间隔的低端，则立即发送ACK

因为发送方经常一个接一个地发送大量的报文段，如果一个报文段丢失，就很可能引起许多一个接一个的冗余ACK。如果TCP发送方接收到对相同数据的3个冗余ACK，它把这当作一种指示，说明跟在这个已被确认过3次的报文段之后的报文段已经丢失。（在课后习题中，我们将考虑为什么发送方等待3个冗余ACK，而不是仅仅等待一个冗余ACK。）一旦收到3个冗余ACK，TCP就执行**快速重传**（fast retransmit）［RFC 5681］，即在该报文段的定时器过期之前重传丢失的报文段。对于采用快速重传的TCP，可用下列代码片段代替图3-33中的ACK收到事件：