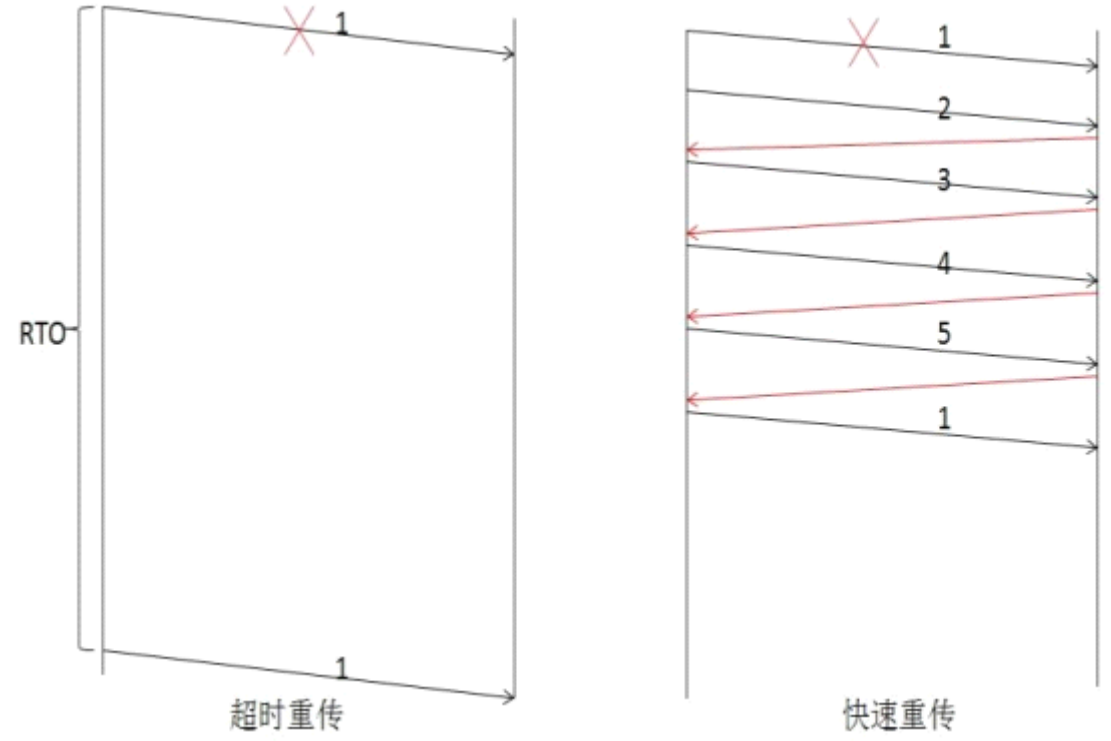
前一篇《NAS 性能优化之一》发布后，一位读者对 smb 和 smb2的不同工作方式很感兴趣。为了检验“叫外卖”（如果对这个词感到困惑，请参考前一篇）的方式能提高多少效率，他在 NAS 和作为客户端的 Windows 7上都启用了 smb2，果然看到读写性能大幅度提升。

熟悉网络的读者可能会存疑：在局域网里的往返时间(RTT)很短，读写的总时间其实大多消费在服务器的响应上。smb2的改进看起来只节省了 RTT，可能达到大幅度提升（比如数倍）的效果吗？这个质疑完全正确，但这位读者的实验结果也是真实的。怎么解释这个矛盾呢？答案就在 TCP 协议上。本文将详解 TCP 中影响 NAS 性能的各个因素，包括对以上矛盾的解释。

在逐条分析之前，先让我们复习一下 TCP 的重传机制，因为后面会多次谈及它。当有 TCP 包在网络上丢失时，TCP 有两种机制来实现重传：超时重传和快速重传。下图展示了这两种情况：

图1，发送方只给接收方传送了一个包。不幸的是这个包在网络上丢失了，所以发送方迟迟等不到来自接收方的确认。在经过一段时间（RTO）之后，发送方认为该包已经丢失了，所以重新传了一次。这个机制就是超时重传。RTO 能达到数百毫秒，这在计算机世界可以算“浪费很长时间”了，NAS 对一个读写请求的响应也就几个毫秒。除此之外，超时重传还会使 TCP 发送窗口降到最小，更是雪上加霜。如果网络中有超过0.1%的超时重传，我们就能看到明显的性能问题。减少超时重传对提高性能有明显的改善。

图2，发送方要给接收方传送5个包，不幸的是第一个就丢失了。由于这个发送窗口>=5个 MTU，所以发送方在没有接收方确认的情况下继续发送了四个包。接收方在收到这些包的时候，可以通过包号知道第一个包丢失了。所以收到第 n 个时（n=2,3,4,5），接收方就发一个“收到n，但1还没收到呢”给发送方（如下图的红线所示）。发送方在收到四个“但是1还没收到呢”的消息后，意识到1可能已经丢失了，就赶紧重传一个。这个机制称为快速重传。由于这个过程没有等待时间，所以对性能影响较小。实现超时重传的条件是发送方在丢了一个包后，接下来还有4个或以上包可以传。



明白了这两个机制后，我们再逐条分析 TCP 对性能的影响因素：

1， TCP 滑动窗口：如果要把10块砖从 A 地搬到 B 地，你是一次搬一块，总共搬10次，还是一口气搬10块呢？在力气允许的条件下，自然是一口气搬完速度快，因为节省了往返时间。网络传输也是如此，如果有10个 TCP 包要传，在带宽允许的情况下应该一起发送，而不是发一个就等确认，然后再发下一个。举个例子，假如往返时间 RTT 是2毫秒，那10个包逐个传至少要花20毫秒；一起传就只需要2毫秒多一点，对性能的提高是显而易见的。除此之外，在发生丢包的时候，大窗口可以提高快速重传的概率，减少了超时重传。比如10个包一起传时，前6个包中任何一个丢失都可以由接下来的4个包触发快速重传。而每次传4个包是永远等不到快速重传的机会的。

2， 多线程：在一个 TCP session 里，如果存在多个线程，也可以在丢包时提高快速重传的概率。还记得《NAS 性能优化之一》里关于 smb2的“叫外卖”图片吗？在第一个请求没有完成的情况下，就可以发送第二个请求。如果第一个请求有丢包，那第二个请求的包可以帮忙凑满四个，从而触发快速重传。本文开头提到的读者在测试 smb2时得到大幅度的性能提升，很可能就得益于此。除了 SMB2和 NFS 协议，EMC 免费提供的 EMCOPY 工具也能在 SMB 中实现多线程拷贝。

3，超时重传时间（RTO）：这是一个动态值。RFC 规定了计算该值的方法，但是结果比较大，已经不适用当今的网络环境了。有些 NAS（比如 Celerra）提供一个设置，允许强制把该值改小。

4， Jumbo Frame：中文好像翻译为巨帧。就是把 MTU 增大到9000，从而减少 TCP 头和 IP 头在一个网络包中所占的比例。理论上这是能提高性能的，但是实际效果却不一定。因为大包的丢失概率更大一点，而且包数少了，就更有可能发生超时重传。

5， 网络拥塞：除了网络配置出错（比如两端的 speed/duplex 不符合），另外一个导致丢包的因素就是网络发生拥塞。如何避免呢？最有效最简单的方法当然是购买更高端的 switch，但这也是最难被接受的建议。有一个将就的办法，就是人为的把 TCP 滑动窗口强制在拥塞点以下。宁愿每次少传一点，也不要丢包。有些 NAS(比如 Celerra)提供了强制最大滑动窗口的设置，但是要确定一个合适的拥塞点比较麻烦，需要抓大量的包分析。

写到这里，突然想到可以写几篇如何利用 Wireshark 分析网络包的。不过，这个读者群应该比 NAS 还小很多吧？