自动重传请求（*Automatic Repeat-reQuest，ARQ*）

一、停止等待ARQ协议

这个协议说白了就是对于收到数据包后的确认机制，基本过程如下：

*1. 发送方A向接收方B发送数据分组M1；

*2. 接收方B收到分组M1后想A发送M1的确认；

*3. A接收到B发送的对于M1后的确认之后再发送分组M2；

*4. 若中间的某一个分组丢失或是延迟，则A就会使用超时计时器进行超时重传，每个分组发送时都会保留其副本并且设置超时计时器；

这里对于超时重传机制需要三点说明：

**1. A发送自己的每个分组，必须暂时保存已发送分组的一个副本，直到收到该副本的确认；

**2. 分组和确认分组都必须编号，这样才能对分组进行区分；

**3. 超时计时器设置的时间应当比数据在分组传输的平均往返时间更长一些；

上面的机制保证了在不可靠的信道上时间可靠的传输，这种机制称为自动重传请求(ARQ)，即重传的请求是自动的，不需要接收方向发送方请求重传某个分组。下面的几个图来说明超时重传的机制：

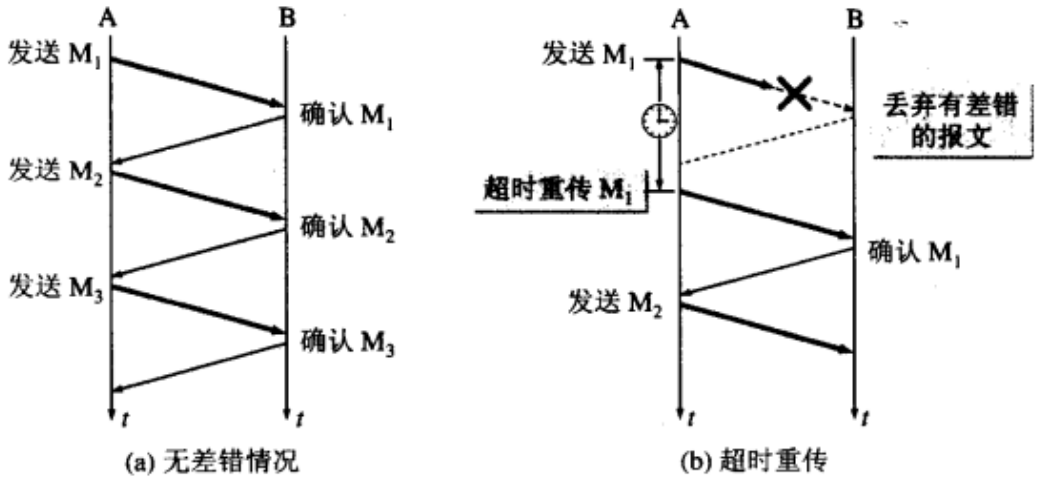


图 5-9　停止等待协议

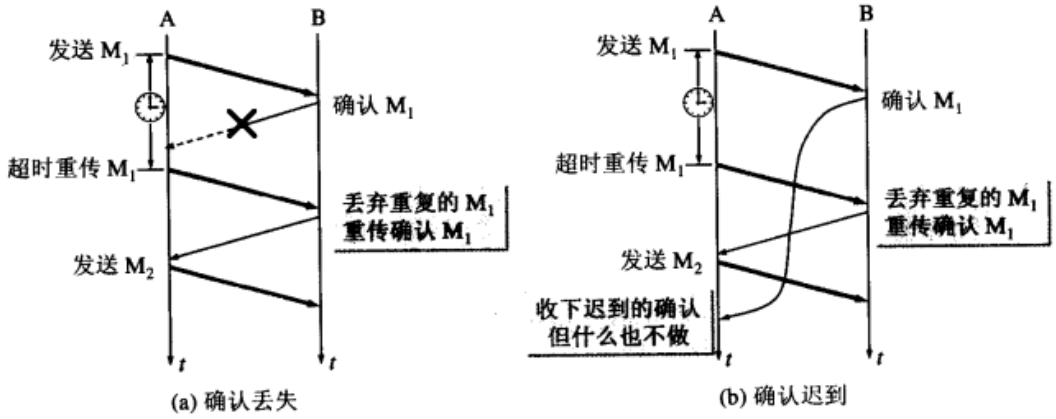
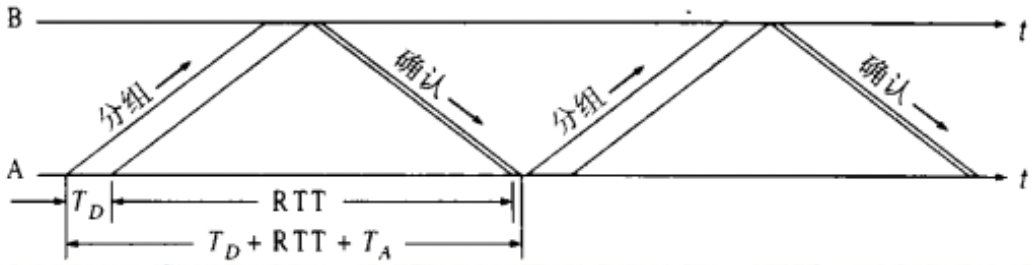


图 5-10　确认丢失(a)和确认迟到(b)

停止等待协议缺点是信道利用率太低!



信道利用率：U = TD / (TD + RTT + TA)

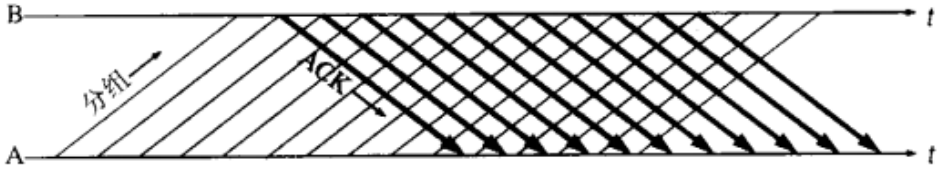
二、回退n帧ARQ协议（GBN go-back-n）

发送方使用滑动窗口发送，但是接收放不维护一个对应的滑动窗口，无法缓存比如2号包丢失后，3号包的数据，

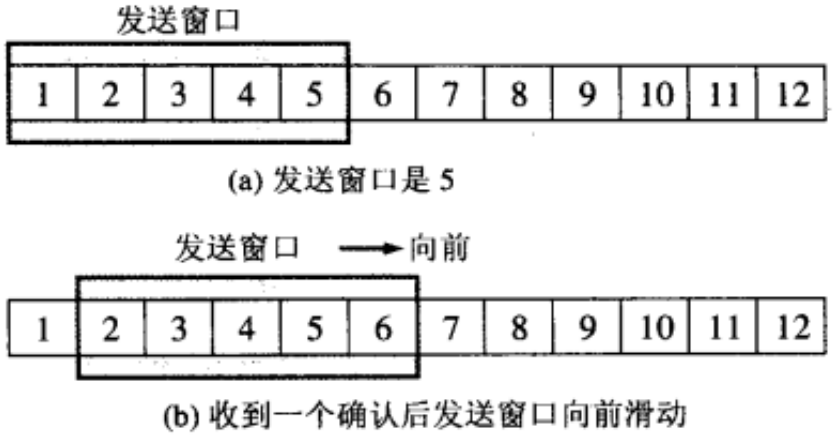
所以在2号包丢失以后，通知发送端，将2号及其之后的包全部重发。

由于停止等待ARQ协议信道利用率太低，所以需要使用连续ARQ协议来进行改善。这个协议会连续发送一组数据包，然后再等待这些数据包的ACK。

发送方采用流水线传输。流水线传输就是发送方可以连续发送多个分组，不必每发完一个分组就停下来等待对方确认。如下图所示：



连续ARQ协议通常是结合滑动窗口协议来使用的，发送方需要维持一个发送窗口，如下图所示：



图（a）是发送方维持的发送窗口，它的意义是：位于发送窗口内的5个分组都可以连续发送出去，而不需要等待对方的确认，这样就提高了信道利用率。

连续ARQ协议规定，发送方每收到一个确认，就把发送窗口向前滑动一个分组的位置。例如上面的图（b），当发送方收到第一个分组的确认，就把发送窗口向前移动一个分组的位置。如果原来已经发送了前5个分组，则现在可以发送窗口内的第6个分组。

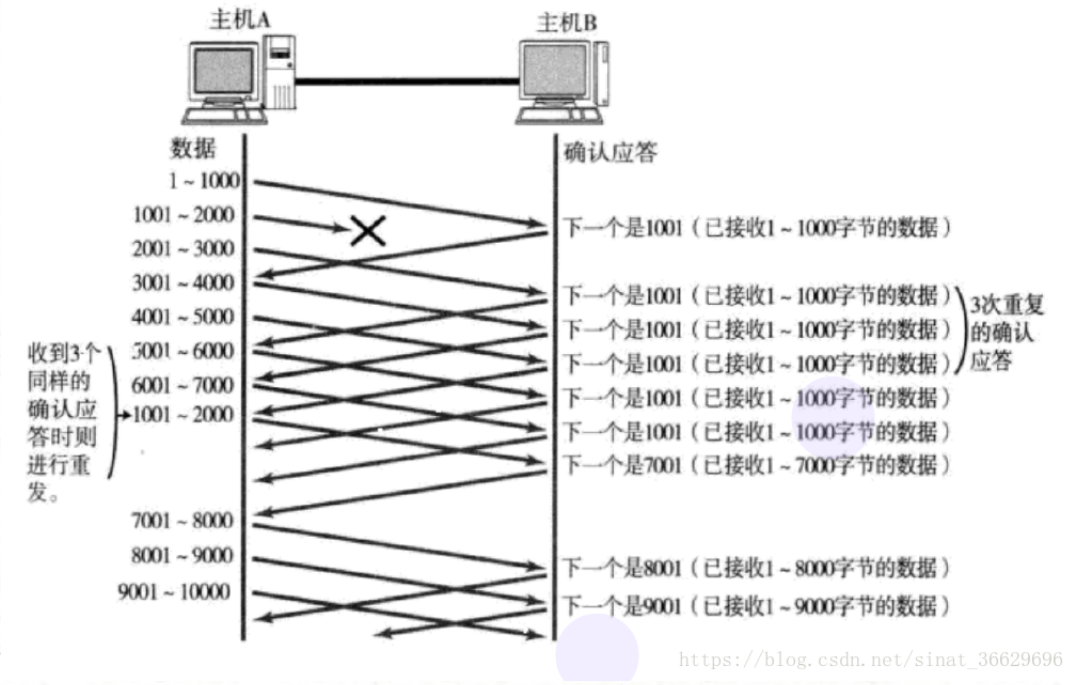
接收方一般都是采用累积确认的方式。也就是说接收方不必对收到的分组逐个发送确认。而是在收到几个分组后，对按序到达的最后一个分组发送确认。如果收到了这个分组确认信息，则表示到这个分组为止的所有分组都已经正确接收到了。

累积确认的优点是容易实现，即使确认丢失也不必重传。但缺点是，不能正确的向发送方反映出接收方已经正确收到的所以分组的信息。比如发送方发送了前5个分组，而中间的第3个分组丢失了，这时候接收方只能对前2个发出确认。而不知道后面3个分组的下落，因此只能把后面的3个分组都重传一次，这种机制叫Go-back-N（回退N），表示需要再退回来重传已发送过的N个分组。

三、选择重传ARQ协议（SR selective-report）

与GBN不同点在于，它不仅仅在发送端维护了一个滑动窗口，还在服务端也维护了一个滑动窗口，用于记录由发送端窗口发送端数据。
此时，丢失哪个包便可以直接发送请求给客户端，要求重传该包

发送数据包的过程与连续ARQ协议相似，只是重传的数据并不是某个数据包之后的都重传，仅仅重传丢失的。



如上图所示
当某一段报文丢失之后, 发送端会一直收到 1001 这样的ACK, 就像是在提醒发送端“我想要的是 1001”
如果发送端主机连续三次收到了同样一个“1001”这样的应答, 就会将对应的数据 1001 - 2000 重新发送
这个时候接收端收到了 1001 之后, 再次返回的ACK就是7001了
因为2001 - 7000接收端其实之前就已经收到了, 被放到了接收端操作系统内核的接收缓冲区中.

四、滑动窗口算法
其实以上介绍的停止等待协议，连续ARQ协议（GBN），选择重传协议（SR），都是基于滑动窗口的具体实现。
停止等待协议是滑动窗口大小为1；

GBN ARQ协议是在发送端维护一个大小为N的发送窗口，在接收端维护一个大小为1的窗口；

SR ARQ协议是在发送端与接收端各自维护一个大小大于1的窗口。

问题：具体窗口的大小如何定义呢？可参见tcp拥塞控制，但是我相信那片文章也没说清楚