我们上一篇介绍了停止等待协议（SW），即发送方每发送一个数据分组后，停止等待接收方的确认分组，当收到确认分组后才能发送下一个分组。由此可以看到，发送方每发送一个数据分组后，至少要等待一个收发双方之间的往返时间，当往返时间较大时，例如卫星链路，停止等待协议的信道利用率很低，若再出现超时重传，信道利用率将更低。

为了提高信道利用率，引出了回退N帧协议和选择重传协议，他们又是如何提高信道利用率的呢？本篇文章我们详细来讨论下。

一、回退N帧协议（GBN）

既然停止等待协议信道利用率这么低，发送一个确认一个，那么容易想到，我能不能在接收确认分组前，连续发送多个数据分组呢？

想一个问题：那我索性一下全部发完，静静等待接收端回复不就好了？

这样做引发的问题有点多，假如要发送的数据很大，一下全发的话，如何做分组的编号？我搞一个很大的数组来标识吗？显然不可能。如果不做分组编号，出现问题怎么办？全部重发？此外，无论是中间的路由器还是末端的接收者，他们的接收队列长度都是有限的，大量丢弃怎么办？

所以啊，一次性发送多个分组也是有讲究的，不能简单粗暴地做这件事。

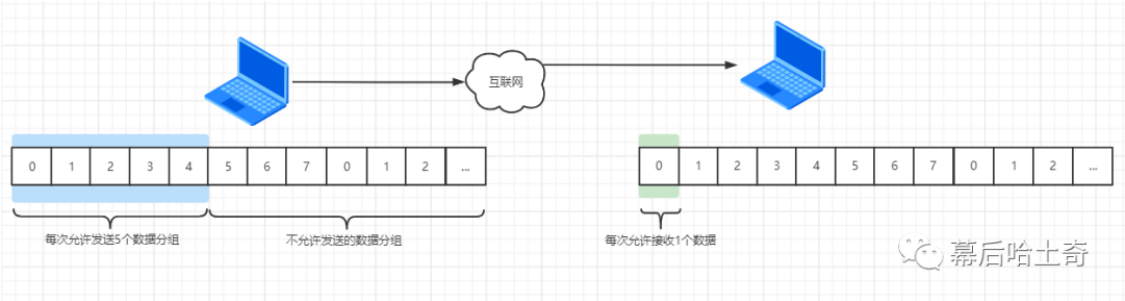
计算机网络的前辈们设计了窗口这个玩意，简单来说就是用来限制你批量发送或接收的分组数量，那么就可以利用发送窗口来限制发送方可连续发送数据分组的数量了。

具体如何来做呢？我们结合一个例子来学习下。

本文假设采用3个比特给数据分组编序号，**序号是0到7对发送的数据分组进行编号（还记得我们给停止等待协议编号只用1个bit就够了吗？）**。

发送方要维持一个发送窗口，序号落在发送窗口内的数据分组可被连续发送，而不必等接收方的相应确认分组后再发送。发送窗口的大小记为**Wt**，对于回退N帧协议，**Wt取值范围是(1,7]（通用公式为大于1、小于(2^N次)-1）**。

为什么要大于1比较好理解，如果Wt的值取为1，那么就是停止等待协议了，没有意义；当Wt的值超出取值范围上限呢？这会引起严重的错误，下面会进行说明，本例先取Wt的值为5。

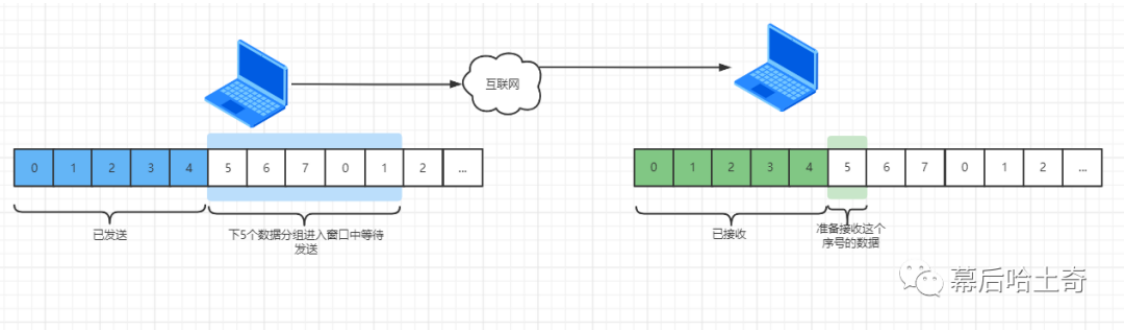


蓝色框为发送窗口，序号落在发送窗口内的数据分组允许发送，不在发送窗口内的数据分组不允许发送。

绿色框为接收窗口，接收窗口的尺寸记为Wr，对于回退N帧协议，其取值只能为1。序号落在接收窗口内的数据分组允许接收，不在接收窗口内的数据分组不允许接收。

我们来看看最简单的无差错情况。

第一次发送的前五个数据分组顺利按序到达了接收方，接收方每接收一个数组分组，接收窗口就向前滑动一个位置，并给发送方发送针对所接收分组的确认分组，即0-4号的确认分组返回，当发送方每接收一个数据分组，发送窗口也向前滑动一个位置，这样就会有新的序号落入了发送窗口。



发送方就可以把已正常处理的发送数据从缓存中删除了，接收方也可以将收到的数据分组择机交给上层应用处理。

我们需要停下来思考下，真的有必要每个数据分组的确认分组都要返回并逐一确认吗？计算机网络的前辈们觉得这个没必要，于是引入了累计确认的概念。

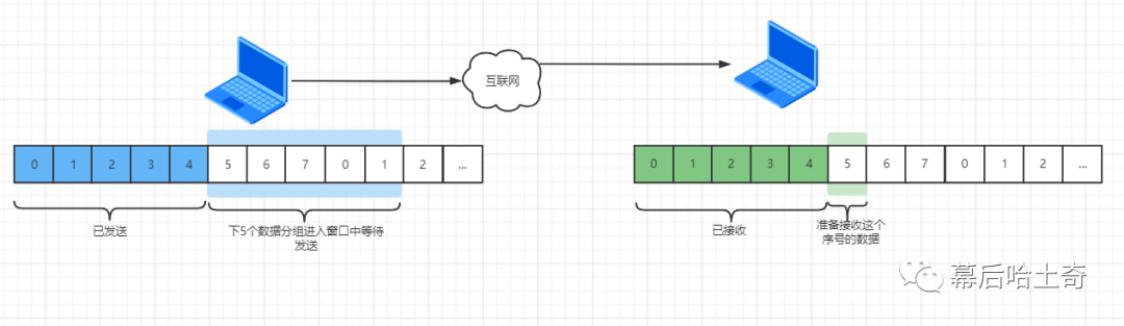
累计确认：接收方不一定要对收到的数据分组逐个发送确认，而是可以在收到几个数据分组后，对按序到达的最后一个数组分组发送确认，ACKn表示序号为n及以前的所有数据分组都已正确接收。

比如上一个例子，接收方在收到第4号分组后，给出ACK4返回给发送方，发送方收到ACK4后就知道0-4号分组都已正确接收，滑动窗口移动到下一个位置继续发送即可。

这样做的一个优点是，即使某个确认分组丢失，发送方也不一定需要重传，比如接收方给了ACK1和ACK4，其中ACK1丢失，只要ACK4没丢也不需要重发，如果ACK1到了ACK4没到，那么只需要重发2-4序号的数据即可。

但是累计确认也有一个小缺点：发送方无法及时知晓接收方正确接收的数据分组信息。

说完了理想情况，下面该来说说有差错的情况了。



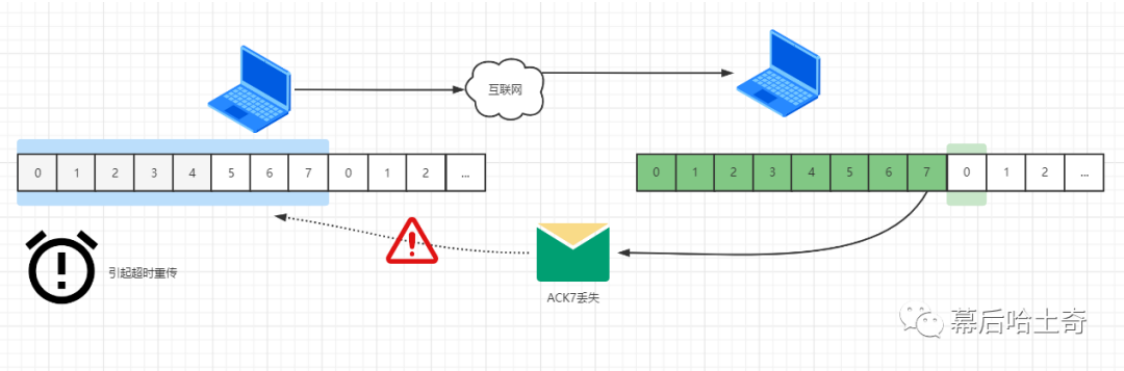
假设本次发送的序号为5的数据分组由于某种原因出现了误码，接收方通过差错码检测出了误码，选择将序号5的数据分组进行丢弃（或者返回NAK让发送方知晓分组存在误码），剩下的序号是6、7、0、1，无法与接收方的序号5进行匹配，剩下的4个分组将全部被丢弃，每丢弃一个分组就返回一个ACK4，这样发送方收到了若干个重复的ACK4后，就知道之前发送的数据分组出现了差错，于是可以不等待超时计时器超时就立刻重传，至于收到几个重复确认就立刻重传，由具体实现决定。

假设收到这若干重复ACK4后仍然不会触发发送方立刻重传，一段时间后，超时计时器出现超时，发送方将发送窗口内已发送过的这些数据分组全部重传。

在本例中，发送方发出了5、6、7、0、1五个数据分组，由于5号数据分组误码不被接收，其余的6、7、0、1四个数据分组受到了牵连也无法正常被接收，发送方仍然要重发这些数据分组，这就是所谓GO-BACK-N即回退N帧名字的由来。

可以看出来，当通信线路质量不好时，很容易出现“连坐”的情况，这样效率将会受到极大折扣。

下面回过头来说说当Wt取值大于窗口上限时，为什么会引起错误。本例中取Wt为8试试，假设发送方发送0-7序号的数据分组，接收方正确收到了这8个数据分组后，给出ACK7的累计确认，但是不幸的是ACK7在回传的时候丢失了。



由于是累计确认，发送方发出去8个数据分组没有收到任何的ACK消息，在一段时间后，引起发送方的超时重传，此时发送方仍然将当前0-7号数据分组发送出去，但是接收方已实际成功接收了0-7号分组，此时就会引发问题：**无法分辨新、旧数据分组！**

在这种情况下，接收方认为自己ACK7已经发出，准备处理后面新的数据了，但是其实发送方超时重发了，接收方收到这些超时重发来的数据分组，并不知道这些数据分组已经被正确接收过了，于是还会正常接收这些数据分组，造成分组重复这种传输差错。

反之，如果发送的数据分组数量在约定的范围以内，接收端可以判断出是发送方重传过来的数据，还是新的数据，这一点还请读者朋友们仔细推敲推敲。

至于Wt为9、10等情况，就不再赘述了，同样会引发相同的问题。

二、回退N帧协议小结

对于发送方：

- 发送方可将在窗口范围的所有数据全部发送出去。
- 发送窗口的Wt取值范围是1<Wt<=(2的N次方-1)，这个N是构成分组号的比特数量，当Wt=1时退化为停止等待协议，当Wt超出最大范围则会引发接收方无法分辨新、旧数据分组。
- 发送方只有收到对已发送的数据分组的确认分组时，发送窗口才能向前相应滑动。
- 发送方收到多个重复ACK分组后，可在重传计时器超时前尽早开始重传，由具体实现决定。
- 发送方发送窗口内某个已发送数据分组产生超时重传时，其后续在发送窗口内且已发送的数据分组也必须全部重传，这就是回退N帧名称的由来。

对于接收方：

- 接收方的接收窗口Wr=1，因此接收方只能按序接收数据分组。
- 接收方只接收【序号落在接收窗口内】且【无误码的数据分组】，并且将接收窗口向前滑动一个位置，与此同时给发送方发出相应的确认分组，为了减少开销，接收方不一定每接收一个按序到达且无误码的数据分组就给发送方发回一个确认分组，而是可以在连续收到好几个按序到达且无误码的数据分组后针对最后一个数据分组发送确认分组，这就是累计确认。
- 接收方收到未按序到达的数据分组，除丢弃外，还要对最近的最后一个按序接收的数据分组进行逐一确认，发回该数据分组重复的累计确认分组（比如上例中的ACK4确认分组），让接收方触发立即重传的机制。

可以看到，协议在工作过程中，发送窗口和接收窗口在不断向前滑动，因此回退N帧协议这类协议也称为滑动窗口协议。

也容易看出，回退N帧协议在数据分组无差错的情况下，其信道利用率比停止-等待协议要高上不少，但是也有很大缺陷，我们下面继续说。

三、选择重传协议（SR）

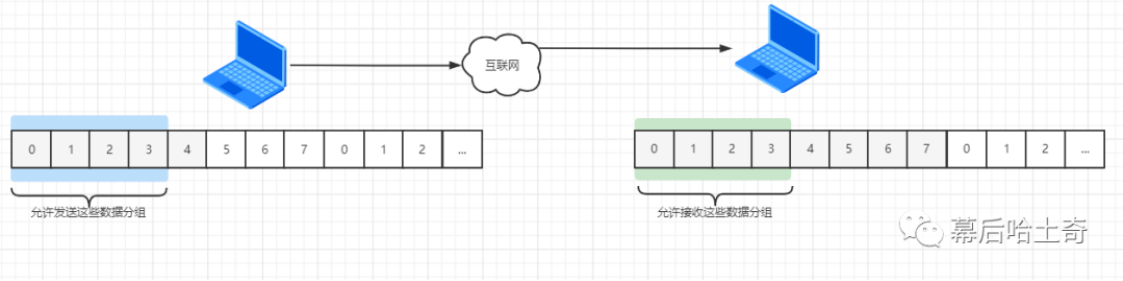
我们现在已经了解了回退N帧协议，我们注意到，**接收窗口Wr只能等于1，因此接收方只能按序接收正确到达的数据分组。**

回退N帧协议最大的问题就在这里：有一个数据分组的误码会导致后续多个数据分组不能被接收方按序接收而丢弃，尽管后续的分组都没有误码，导致发送方对这些数据分组的重传，极大浪费了通信资源。

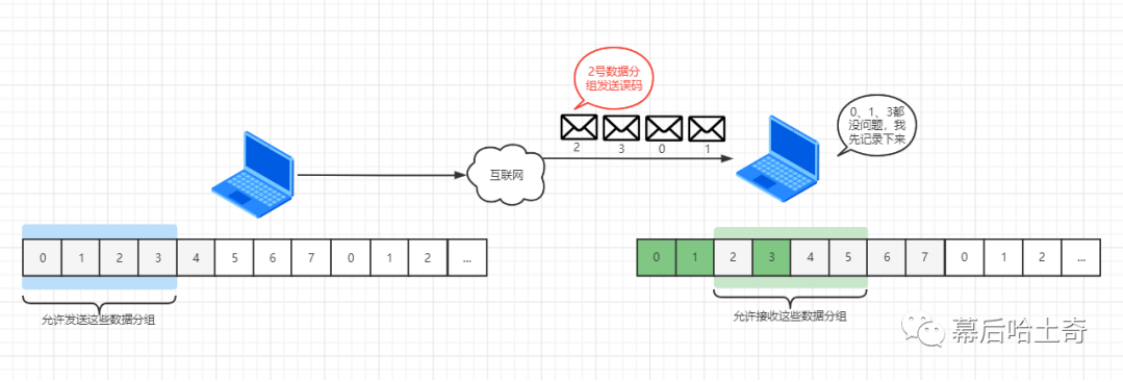
为了进一步提高性能，可设法只重传出现误码的数据分组，因此接收窗口的Wr不应只能等于1，以便接收方先收下【失序到达】且【无误码】且【序号落在接收窗口内】的那些数据分组，这样就可以针对误码的数据分组进行单独的处理，等接收方所缺的分组收齐后一起交付给上层应用处理，这就是选择重传协议名称的由来。由于是失序分组，且收到一个在接收窗口内的数据分组就要接收并发回确认分组，因此在选择重传协议中，接收方不能再采用累积确认，而需要对每个正确接收到的数据分组逐一确认。

下面我们来详细看看选择重传协议的工作原理。

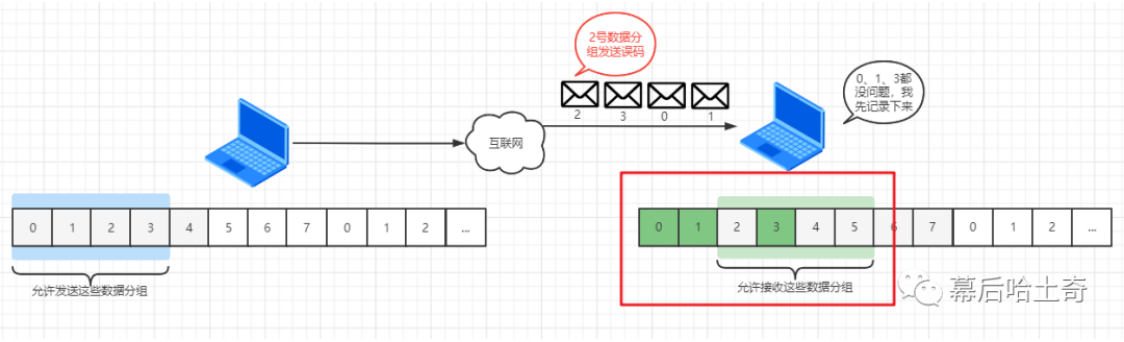
- 采用3个比特给数据分组进行编号，即序号为0-7。
- 发送窗口的Wt取值范围是1<Wt<=(2的(N-1)次方)，本例中N等于3，范围为1<Wt<=4，我们这里将Wt取值为4。
- 接收窗口Wr取值可跟Wt一样，本例中取值也为4。



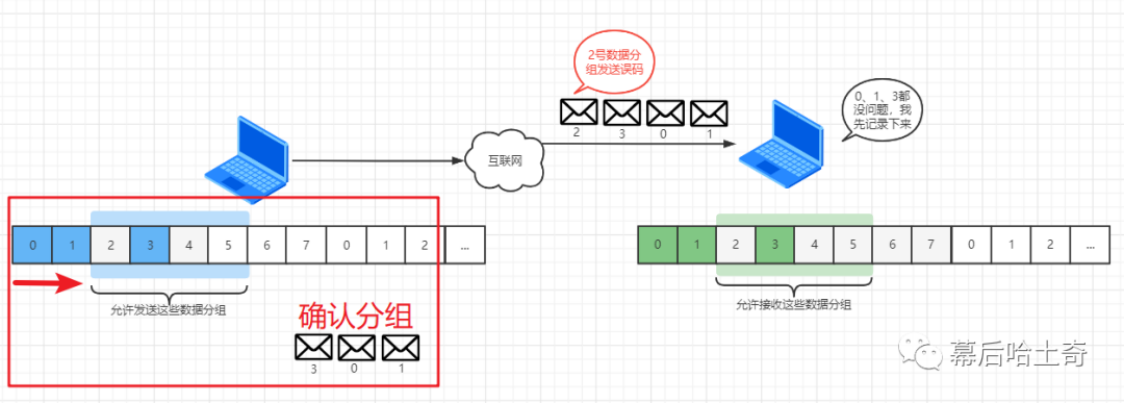
正常情况比较简单，两边窗口同时往前移动即可，**但是当某个数据分组出现误码了，如何处理呢？**



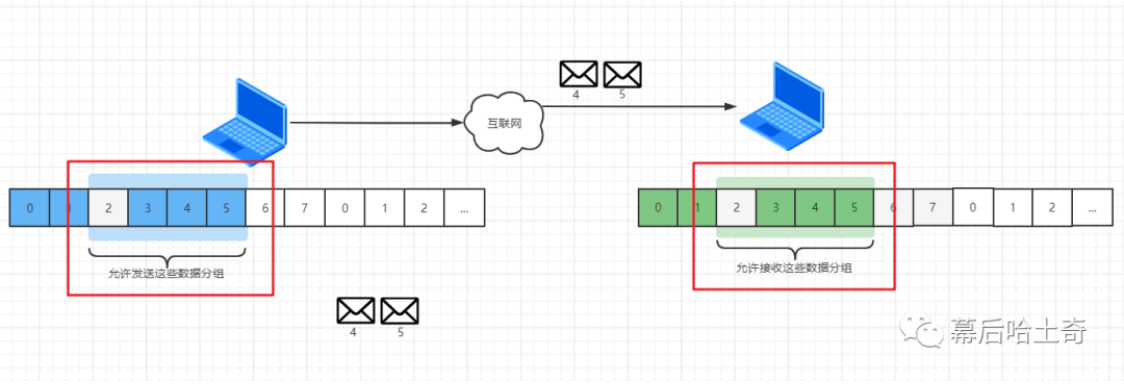
比如本例中序号为2的数据分组出现了误码，接收方判断是误码后进行丢弃，其他的数据分组可正常接收，0和1号分组是正常接收的，那么滑动窗口往后移动两个位置：



并且针对0、1、3分别给出确认分组，发送方进行窗口滑动，不过由于2号分组误码，因此滑动窗口只能往后移动两个位置：

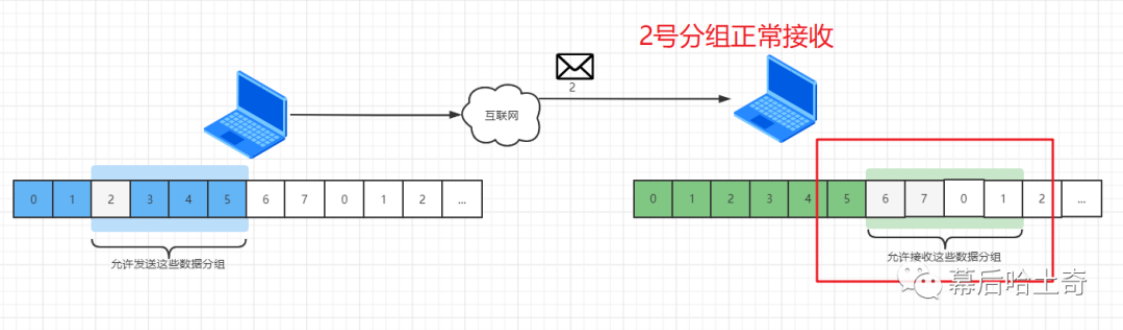


下面发送方将继续发送4、5号数据分组，2号分组需要等待超时重发，可能会发生4、5号数据分组先得到确认的情况：

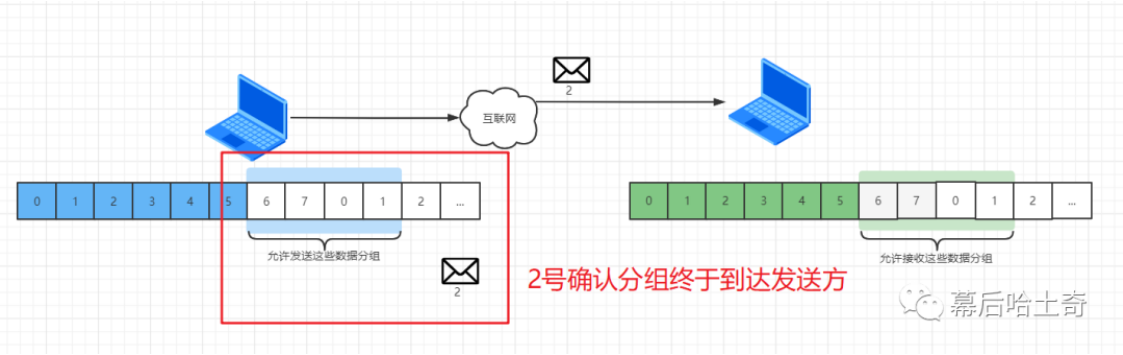


由于2号数据分组此时还未重发，所以发送窗口和接收窗口都不能往后滑动！此时**发送方已记录3、4、5数据分组都已正常发送和接收，因此不会触发其重发**。

等待一段时间后，2号数据分组重发，接收方收到2号分组后，发回2号确认分组，接收窗口往后滑动4个位置：



若发送方收到2号确认分组，那么发送窗口也往后移动4个位置：



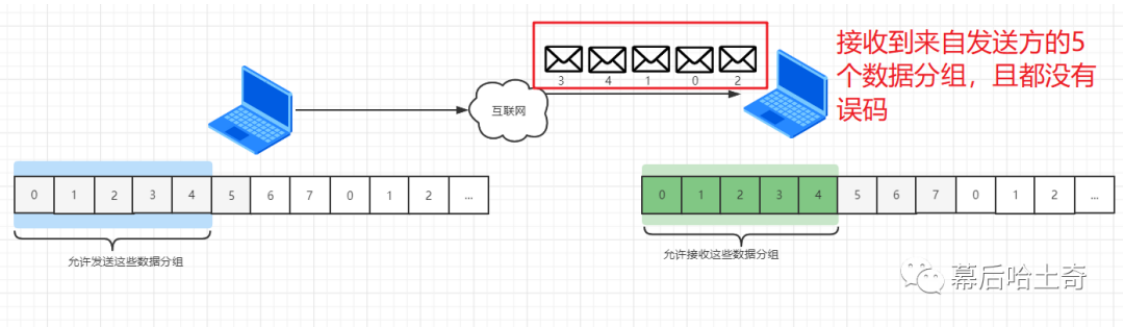
下面就有6、7、0、1号数据分组落入发送窗口，那么就继续如上的过程发送即可。

请注意Wt的取值范围是：1<Wt<=(2的(N-1)次方)，当大于这个范围时，则会造成接收方无法分辨新、旧数据分组的问题。

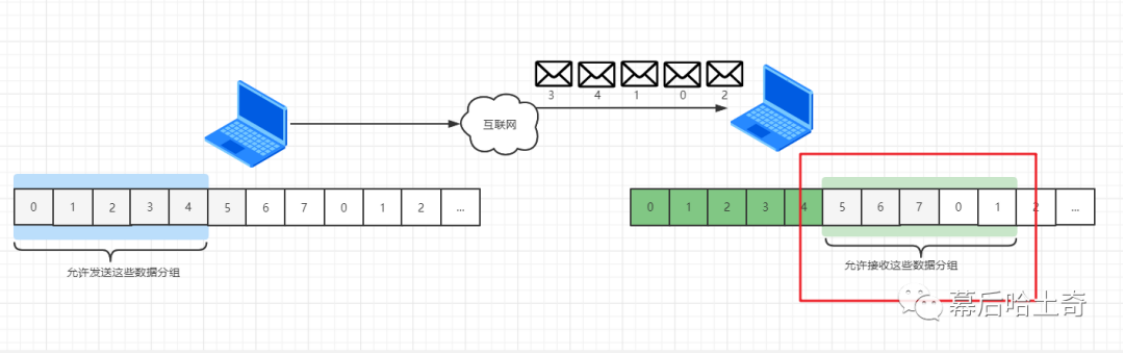
Wr的取值范围应满足：1<Wr<=Wt，当Wr=1则与回退N帧协议一样了，当Wr大于等于Wt，则没有任何意义。

下面来看看超出范围时为什么会造成接收方无法分辨新、旧数据分组的问题。

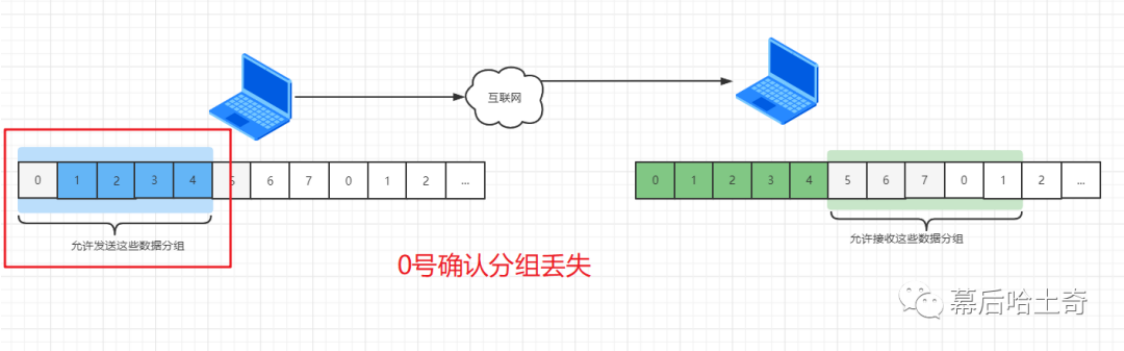
本例中Wt最大可设置为4，当我们设置为5，Wr仍然等于Wt。



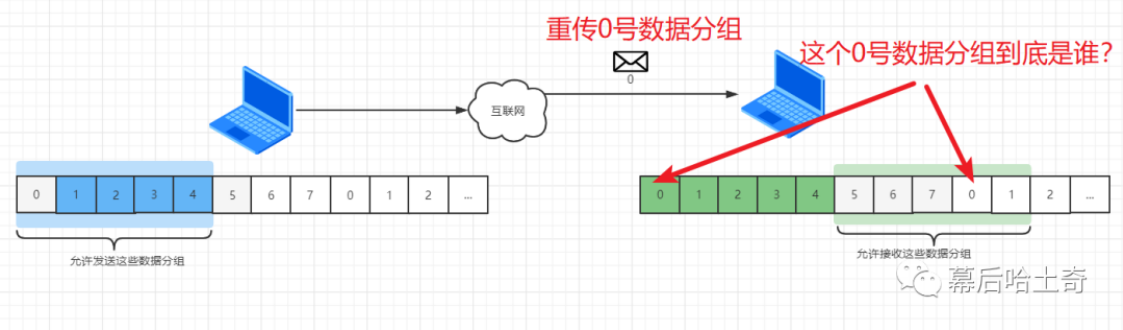
回复ACK确认分组，接收方移动接收窗口到下一组位置：



当0号数据分组的确认分组丢失，1-4号数据分组的确认分组安全到达，发送方滑动窗口不能往后滑动：



这样引起发送方对0号数据分组的重传，此时0号数据分组到达接收方后，无法分辨到底是老数据还是新数据：



四、选择重传协议小结

发送方：

- 发送方窗口Wt取值范围是：1<Wt<=(2的(N-1)次方)，当为1时，则退化为停止等待协议，若超出最大范围，则可能会造成接收方无法分辨新、旧数据。
- 发送方可在未收到接收方确认分组的情况下，将落在发送窗口内的多个数据分组全部发送出去。
- 发送方只有按序收到对已发送数据分组的确认时，发送窗口才能向前滑动，当收到未按序到达的确认分组时，对其进行记录，防止重发，发送窗口不能向前滑动。

接收方：

- 接收方窗口Wr取值范围是1<Wr<=Wt。当为1时，退化为停止等待协议，若超出最大范围，则没有任何意义，一般情况下，在选择重传协议中，Wt和Wr相等。
- 接收方可接收未按序到达但没有误码并且序号落在接收窗口内的数据分组，为了使发送方仅重传出现差错的分组，接收方不能再采用累积确认，而需要对每个正确接收到的数据分组逐一确认。
- 接收方只有在按序接收数据分组后，接收窗口才能向前滑动。

可以看到，选择重传协议相比回退N帧协议，采取【逐个分组确认】+【扩大接收窗口】+【提前缓存失序数据分组】，避免“连坐”情况的发生，只用针对某个数据分组进行重传即可。