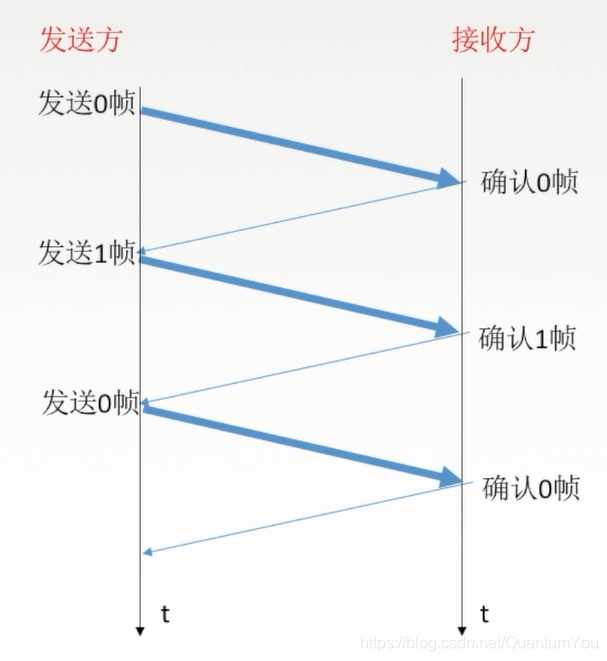
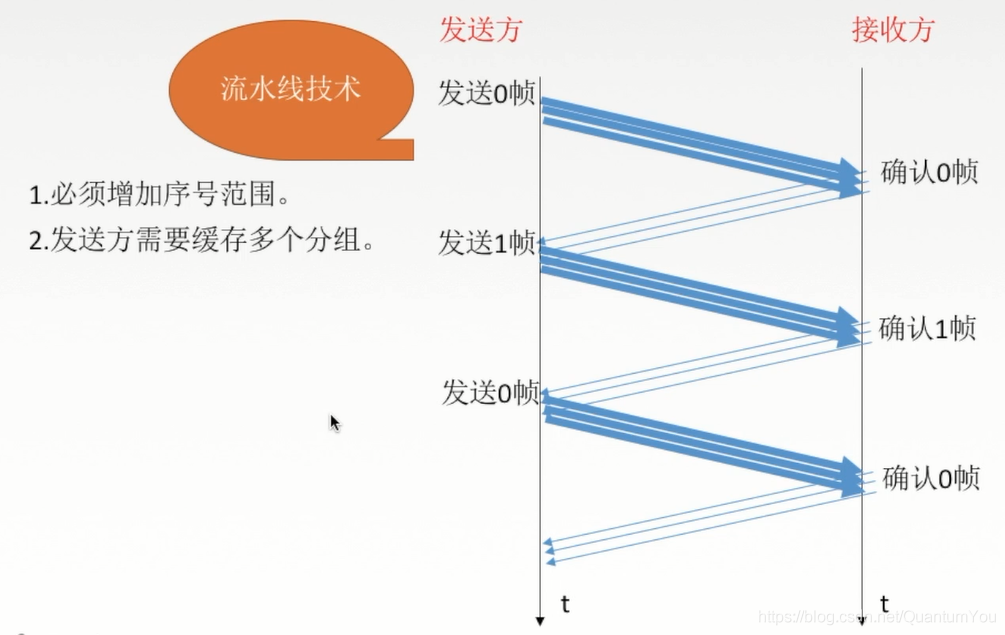
喜欢专研自己所热爱的知识，以及项目完成的那一份喜悦感  
大三学年利用好时间 （985 靠齐） 学习策略

@[toc]

# 停等协议的弊端

* 发送方发送速度很快，但是需要接受到确认才能继续发送（导致发送方一直处于等待状态）  
    
  **改进方法**：  
  

# 后退N帧协议

## 滑动窗口



## GBN发送方必须响应的三件事

GBN (Go-Back-N)

**1、上层调用**

* 上层要发送数据时，发送方先检査发送窗口是否已满，如果未满，则产生一个帧并将其发送；如果窗口已满，发送方只需将数据返回给上层，暗示上层窗口已满。上层等一会再发送。（实际实现中，发送方可以缓存这些媺据，窗口不满时再发送帧）。

**2、收到了一个ACK**

* GBN协议中，对n号帧的确认采用**累积确认**的方式，标明接收方已经收到n号帧和它之前的全部帧。

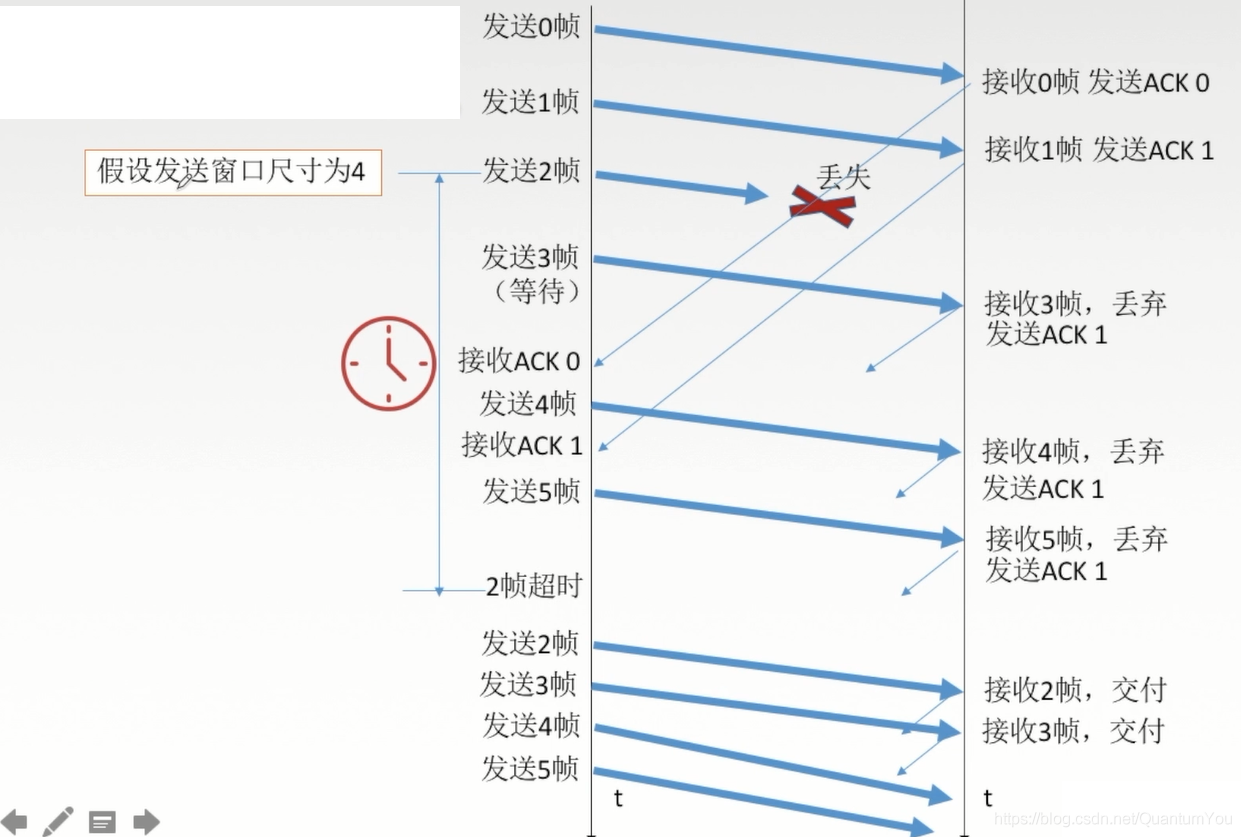
**3、超时事件**

* 协议的名字为后退N帧/回退N帧，来源于出现丢失和时延过长帧时发送方的行为。就像在停等协议中一样，定时器将再次用于恢复数据帧或确认帧的丢失。如果岀现超时，发送方重传所有已发送但未被确认的帧。

## GBN接收方要做的事

* **正确时**：如果正确收到n号帧，并且按序，那么接收方为n帧发送一个ACK，并将该帧中的数据部分交付给上层。
* **非正确时**：其余情况都丢弃帧，并为最近按序接收的帧重新发送ACK.接收方无需缓存仼何失序帧，只需要维护一个信息： expectedseqnum（下一个按序接收的帧序)

## 运行中的GBN



## 滑动窗口长度

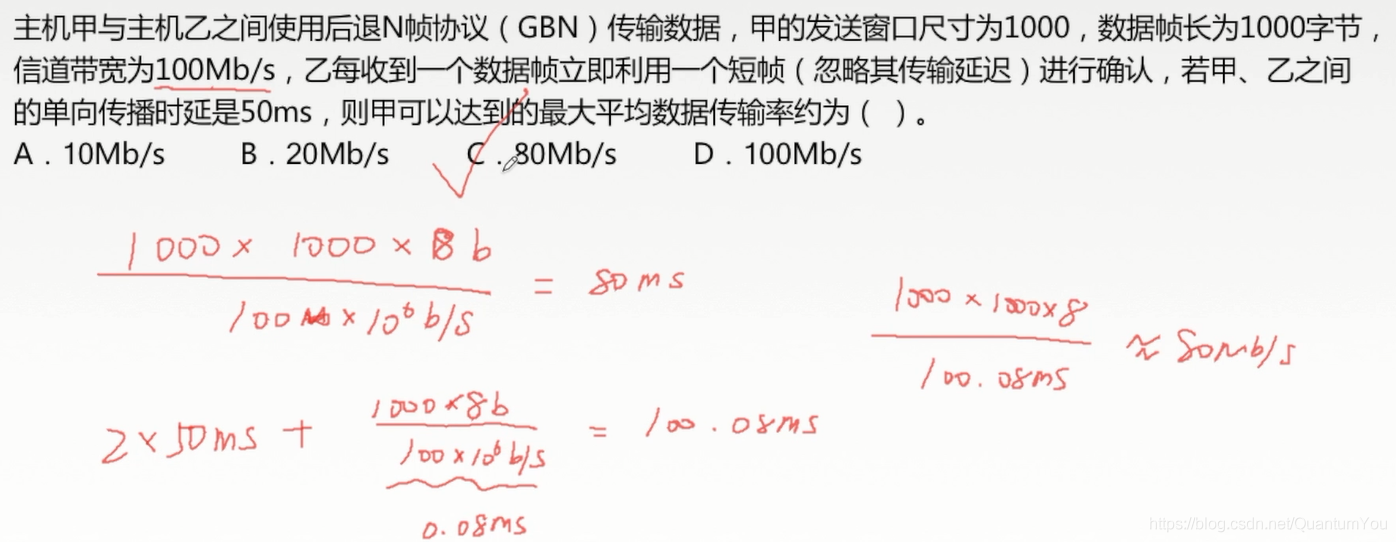
* 若采用n个比特对帧编号，那么发送窗口的尺寸W应满足1≤≤2n-1. 因为发送窗口尺寸过大，就会使得接收方无法区别新帧和旧帧

## GBN 协议重点总结

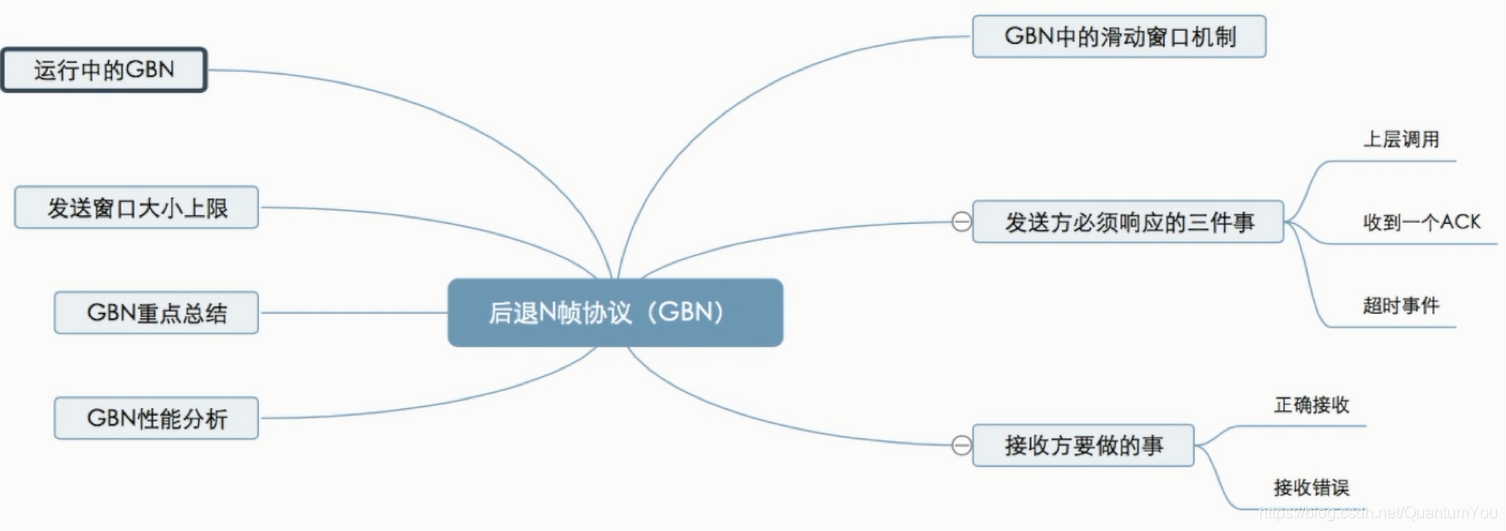
1. 累积确认（偶尔捎带确认）
2. 接收方只按顺序接收帧，不按序无情丢弃
3. 确认序列号最大的、按序到达的帧
4. 发送窗口最大为2n-1，接收窗口大小为1

## GBN协议性能问题

## 习题



## 小结思维导图

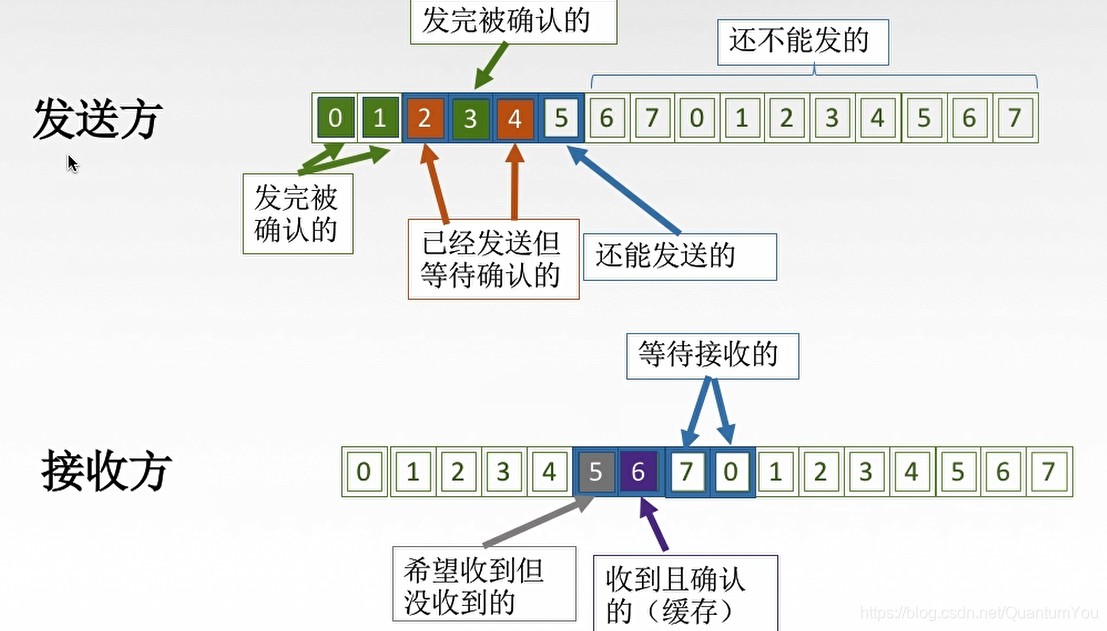


# 选择重传协议（SR）

## GBN 协议的弊端

* 累积确认 ---> 批量重传
* 解决办法：设置单个确认，同时加大接受窗口，设置接受缓存，缓存乱序到达的帧。

## 选择重传协议中的滑动窗口



## SR发送方必须响应的三件事

**1、上层的调用**

* 从上层收到数据后，SR发送方检査下一个可用于该帧的序号，如果序号位于发送窗口内，则发送数据帧；否则就像GBN一样，要么将数据缓存，要么返回给上层之后再传输。

**2、收到了一个ACK**

* 如果收到ACK,加入该帧序号在窗口内，则SR发送方将那个被确认的帧标记为已接收。如果该帧序号是窗口的下界（最左边第一个窗口对应的序号），则窗口向前移动到具有最小序号的未确认帧处。如果窗口移动了，并且有序号在窗口内的未发送帧，则发送这些帧

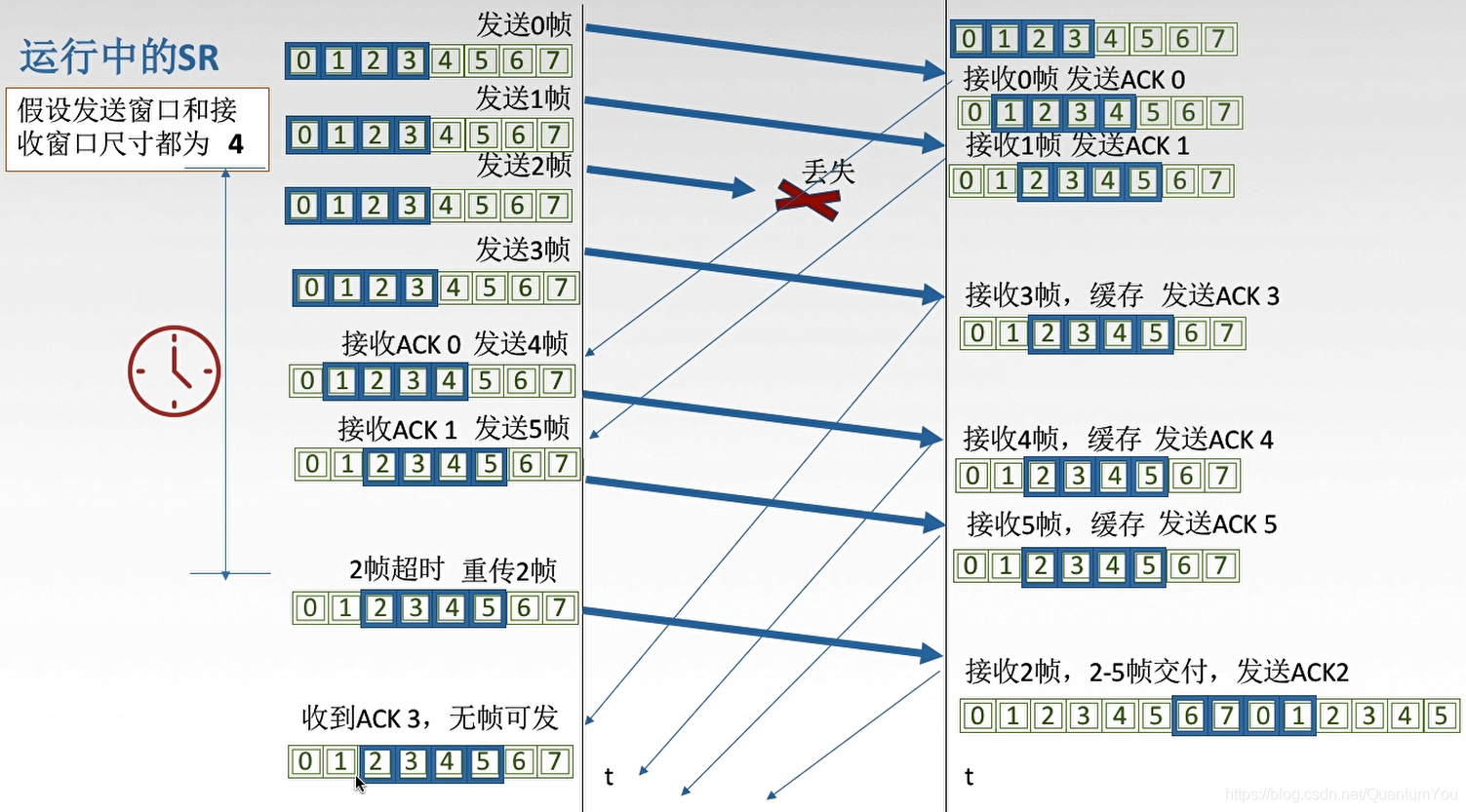
**3、超时事件**

* 每个帧都有自己的定时器，一个超时事件发生后只重传一个帧

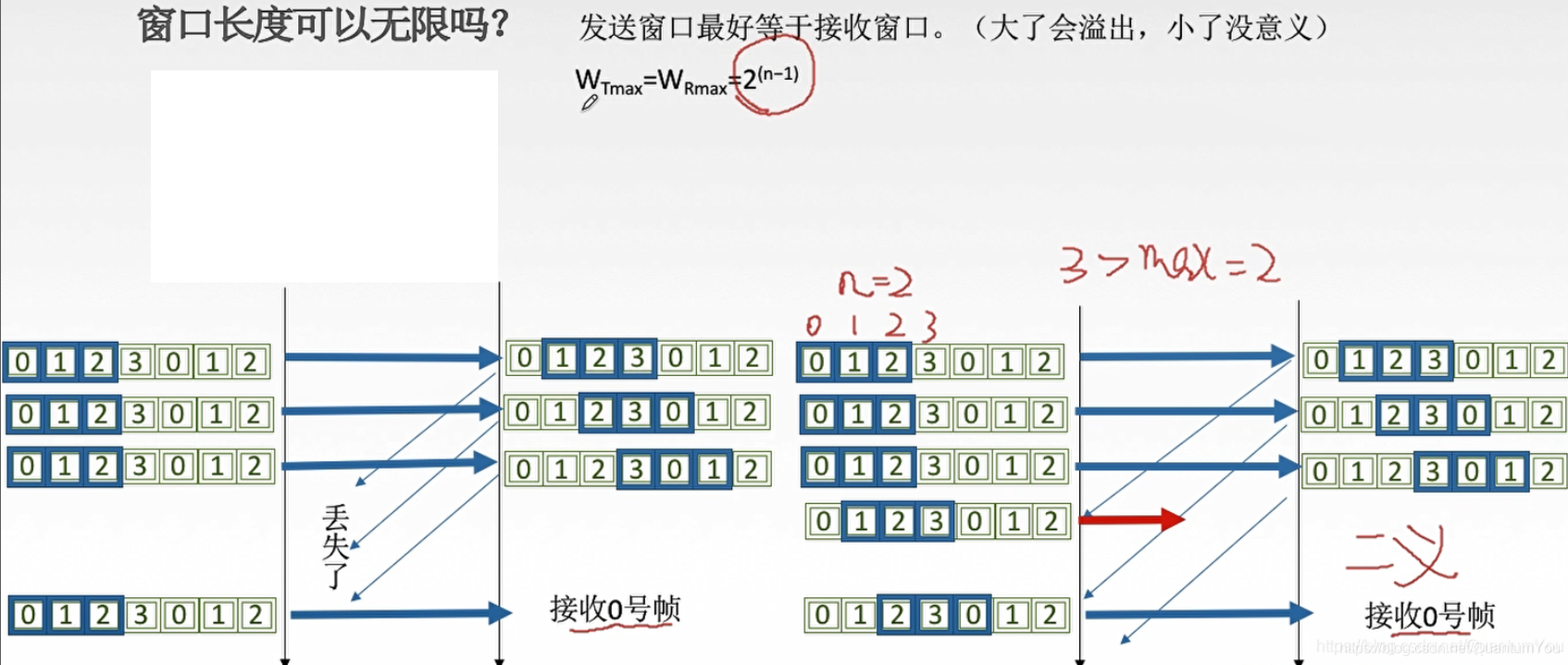
## SR 接收方要做的事情

* 来者不拒（窗口內的帧）  
  SR接收方将**确认一个正确接收的帧**而不管其是否按序。失序的帧将被缓存，并返回给发送方一个该帧的确认帧【收谁确认谁】，直到所有帧（即序号更小的帧）皆被收到为止，这时才可以将一批帧按序交付给上层，然后向前移动滑动窗口。
* 如果收到了窗口序号外（小于窗口下界）的帧，就返回一个ACK其他情况，就忽略该帧。

## 运行中SR



## 滑动窗口长度



* 其中上述 的 n 为比特位 比如 0，1，2，3 共有 4 位，即 22

## SR 协议总结

1、对数据逐一确认，收一个确认一个  
2、只重传出错帧  
3、接收方有缓存  
4、 = 2n-1

## 小结思维导图

