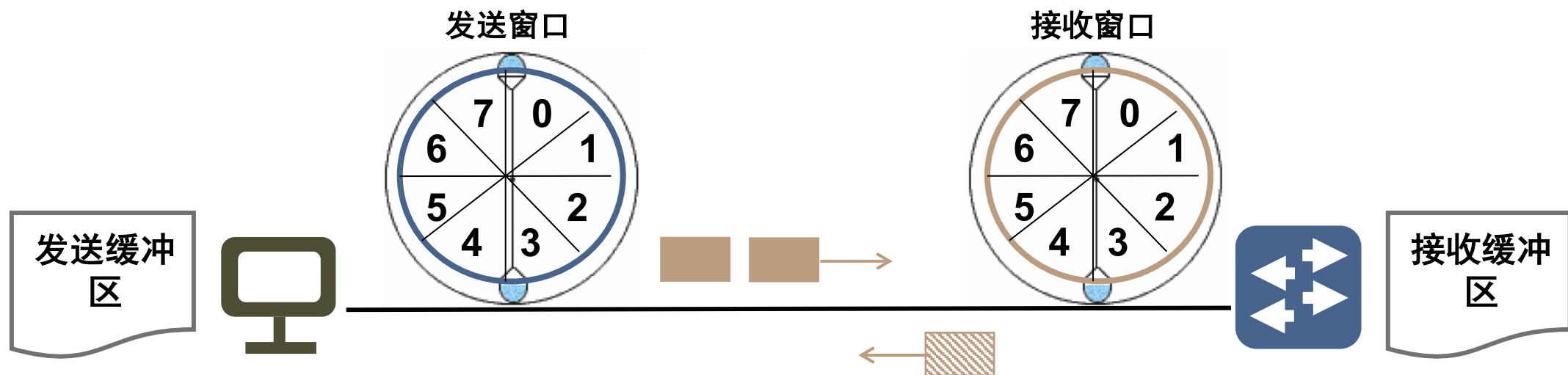


基于滑动窗口机制的 差错控制



基于滑动窗口机制的差错控制



- 可连续发送由发送窗口指定的多个帧
- 每发出一个帧启动一个计时器
- 在计时器超时后仍未收到来自接收方的确认则重发该帧

- 每收到一个数据帧检查序号是否落在接收窗口, 是则接收, 否则丢弃
- 检验该帧传输是否出错
- 如果传输无误且允许接收则给发送方反馈一个肯定确认

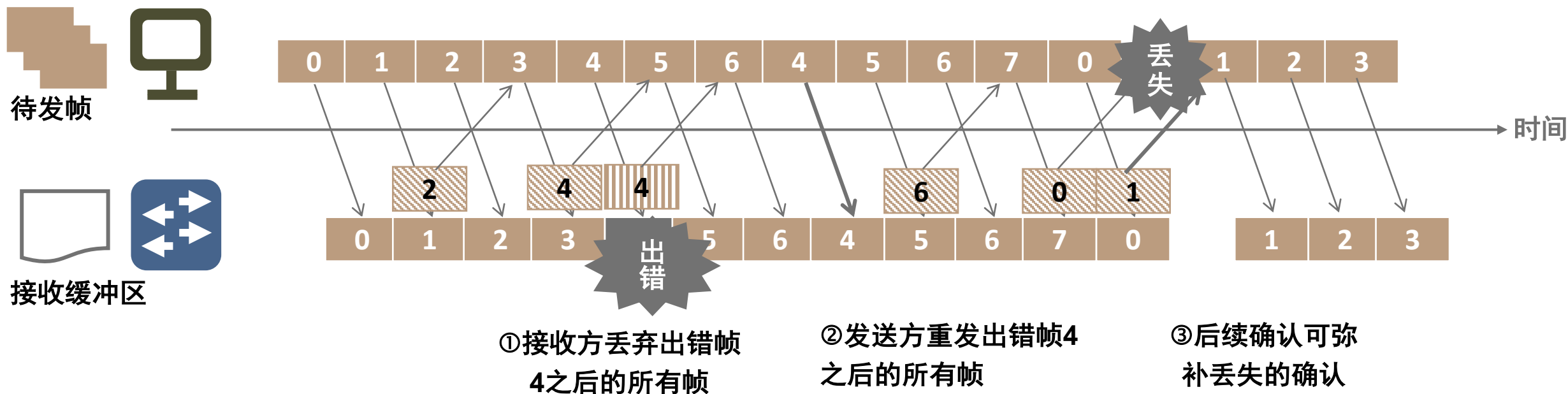


回退-N ARQ基本思想

顺序收发方式：接收方只能按照帧的序号接收数据帧。

回退N控制策略

- 发送方连续发出N个帧，接收方以流水线方式顺序接收各个帧，并进行差错检测。
- 一旦某个帧有错，则丢弃该帧和它之后所收到的所有帧。



N：发送窗口大小



否定确认



肯定确认



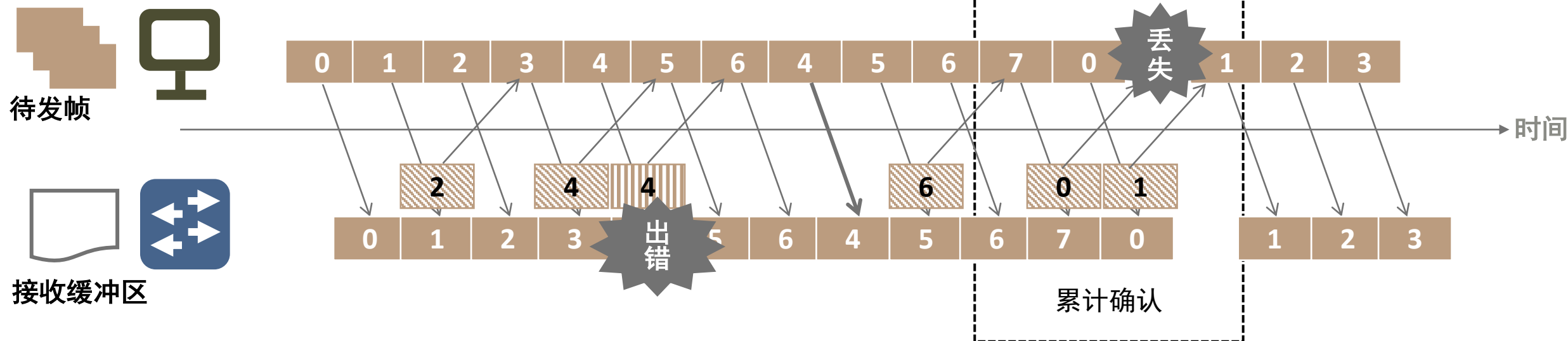
数据帧



北京大学

回退-N中的累计确认

1 肯定确认表示已经正确接收直到0的所有帧，因而可弥补丢失的肯定确认 0



N : 发送窗口大小



否定确认



肯定确认





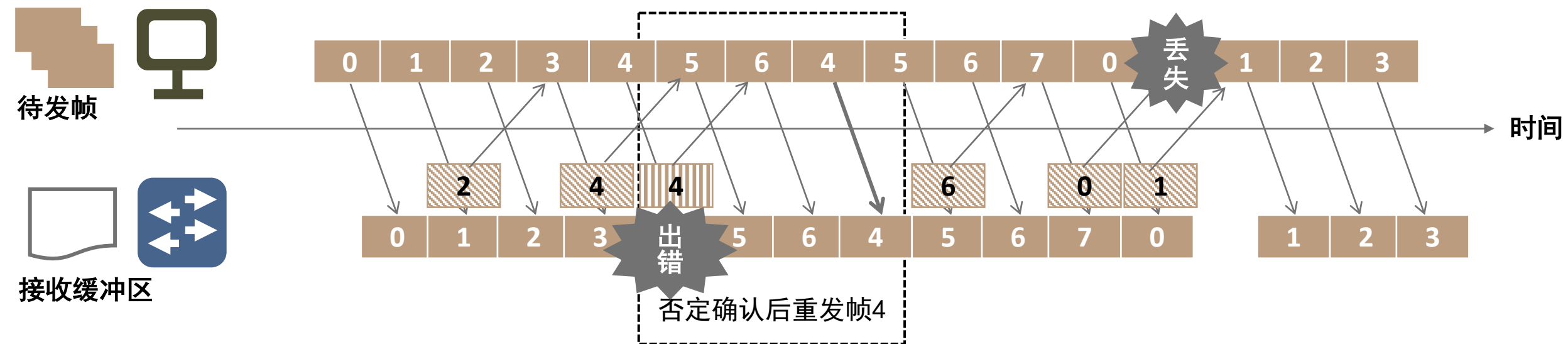
数据帧



北京大学

回退-N ARQ中的否定确认

-  4 • 肯定确认表示已经正确接收直到3的所有帧，期待接收帧4
-  4 • 否定确认表示帧4出错，要求重发帧4



回退-N ARQ的发送窗口大小

假设：序号用n位表示

- 序号空间：0, 1, 2, 3, $2^n - 1$
- 模 $m = 2^n$
- 最大序号 $S_{\max} = m - 1 = 2^n - 1$

？ 发送窗口N取多大合适

回退N-ARQ控制，最大发送窗口

$$N = 2^n - 1$$

- 窗口越大，发送方在接收方确认帧返回之前可发送的数据帧越多
- 接收方必须分配更多的资源和更大的缓冲区来应付入境数据帧



回退-N ARQ特性

优点

消除了停一等ARQ的等待确认时间。

缺点

正确帧的重发无疑浪费了信道。

回退-N ARQ 特点

- 要求每一帧的确认在其后第N个帧尚未结束发送之前到达
- 发送方必须有存放N帧的缓存以便出错时重发
- 接收方只要求存放一帧大小的缓冲区
- 要求全双工链路

若第一帧的确认在第N帧发完以前尚未到达，表明信道往返时延较大，可加大帧的长度或增加N的值，否则发送方要空等。



选择重传ARQ基本思想

乱序收发方式：接收方可以接收顺序出错但传输无错的数据帧。

选择重传控制策略

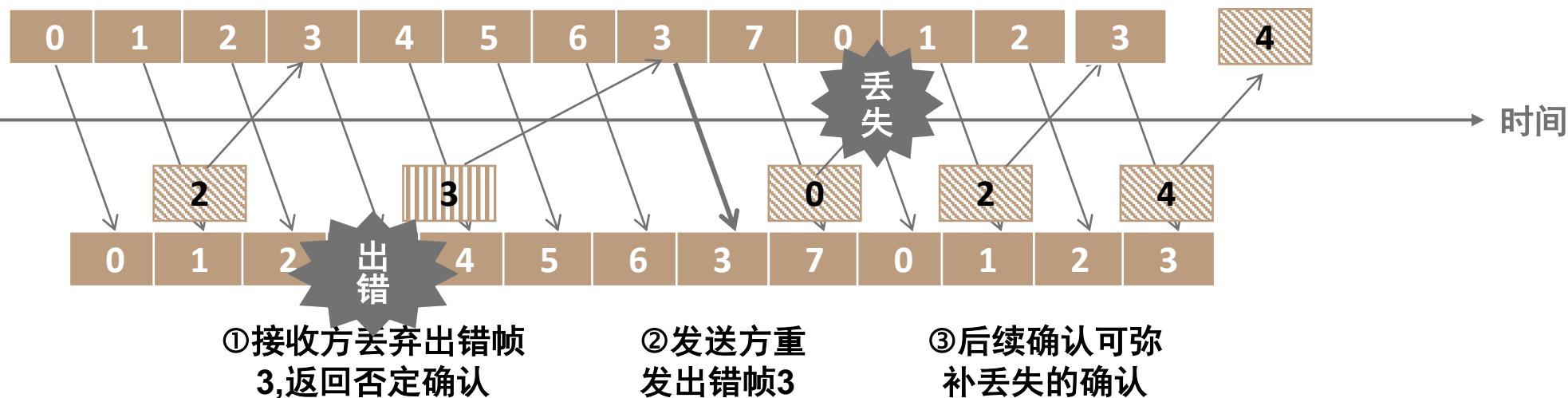
- 发送方可连续发送N个帧，接收方收到出错帧时给发送方反馈出错帧序号，要求发送方只重发出错的帧；
- 出错帧后续到达的帧被保存，待重发后的出错帧到达目的地，接收方把接收到的帧重新排序上交给网络层。



待发帧



接收缓冲区



N：发送窗口大小



否定确认



肯定确认



数据帧

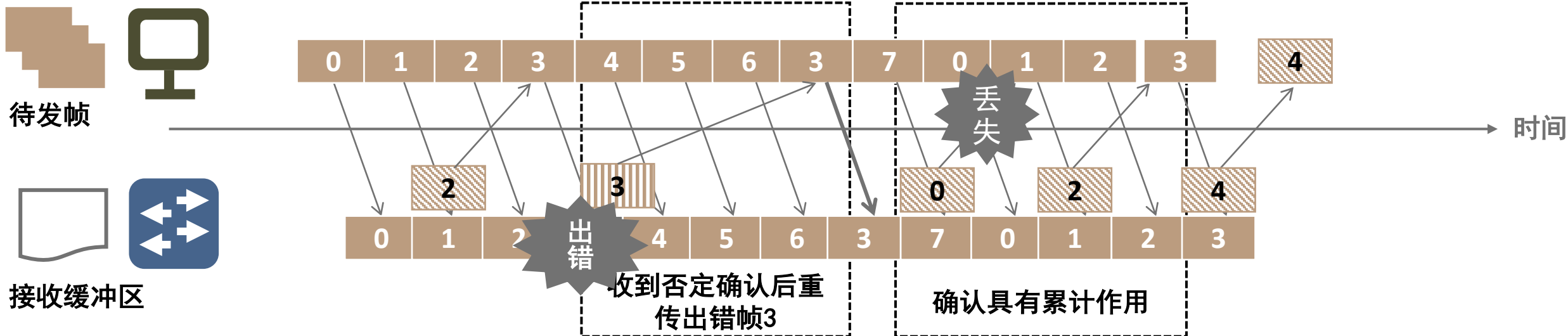


北京大学

选择重传ARQ中的累计和否定确认

3 发送方针对否定确认采取的措施与回退N不同，发送方仅发送出错的帧3。

2 肯定确认表示已经正确接收直到1的所有帧，因而可弥补丢失的肯定确认 0



选择重传ARQ的滑动窗口大小

假设：序号用n位表示

- 序号空间：0, 1, 2, 3, $2^n - 1$
- 模 $m = 2^n$
- 最大序号 $S_{\max} = m - 1 = 2^n - 1$

?

选择重传的发送窗口
可以和回退N的发送
窗口一样

选择重传ARQ控制，最大发送窗口

$$N = 2^{n-1}$$

- 发送方要能处理所有可能的出错情况（错1个帧、2个帧、。。。、N个帧）
- 重传时要确保接收方不会发生接收错误



选择重传ARQ特性

优点

- 消除了发送方的等待确认时候
- 信道利用率高

缺点

- 实现复杂
- 接收方要有存放N个帧的缓冲区

选择重传ARQ特点

- 发送方必须有存放N帧数据的缓存
- 接收方必须有足够存储空间以便缓存(N-1)个帧
- 接收方的接收顺序可能会打乱原发送顺序
- 要求全双工的工作链路

若第一帧的确认在第N帧发完以前尚未到达，表明信道往返时延较大，回退N可加大帧的长度或增加N的值，但选择重传只能加大帧长而不能增加N，否则接收方会发送接收错误。



两种ARQ控制的比较

回退N 的ARQ控制

- 因只能接收正确顺序的数据帧，相对控制简单
- 对接收方的缓存空间要求小，只要有存放一个数据帧的大小即可
- 要重传出错帧后续的所有数据帧，浪费网络带宽

以时间和带宽为代价

选择重传的ARQ控制

- 接收方的控制因乱序接收数据帧而复杂
- 对接收方的缓存空间要求大，必须具备和发送方一样大的缓冲区
- 只重传出错的帧，相对节省网络带宽

以空间和复杂性为代价

