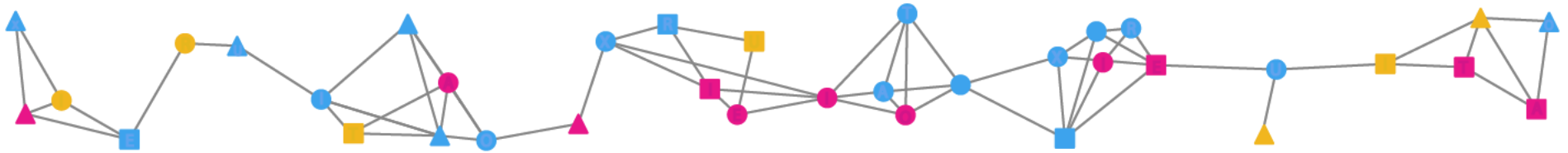


计算机网络

第三章 传输层

谢瑞桃
xie@szu.edu.cn
rtxie.github.io
计算机与软件学院
深圳大学

第三章讲解内容

1. 传输层概述与UDP
   - 需求/服务/协议、多路复用/分解、UDP协议
2. 可靠传输
   - 可靠传输基础知识、TCP可靠传输
3. TCP
   - 报文段结构、超时间隔、流量控制、连接管理
4. TCP拥塞控制
   - 网络拥塞、TCP拥塞控制、吞吐量分析

传输层需求、服务和协议

应用层需求	传输层服务	UDP	TCP
为运行在不同主机上的进程之间提供逻辑通信	进程间交付	✓	✓
检测报文段是否出错	差错检测	✓	✓
解决丢包、差错问题	可靠传输	✗	✓
解决乱序问题	按序交付	✗	✓
解决接收缓存溢出问题	流量控制	✗	✓
应对网络拥塞	拥塞控制	✗	✓

可靠传输讲解内容

- 解决差错问题
- 解决丢包问题
- 停等协议与流水线协议
- 滑动窗口协议
- 回退N步
- 选择重传
- TCP可靠传输

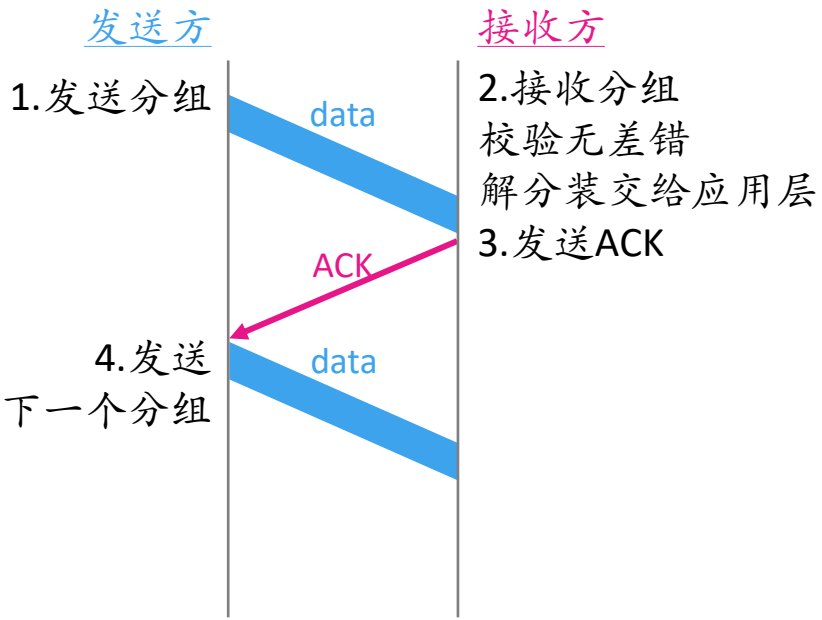
可靠传输基础知识

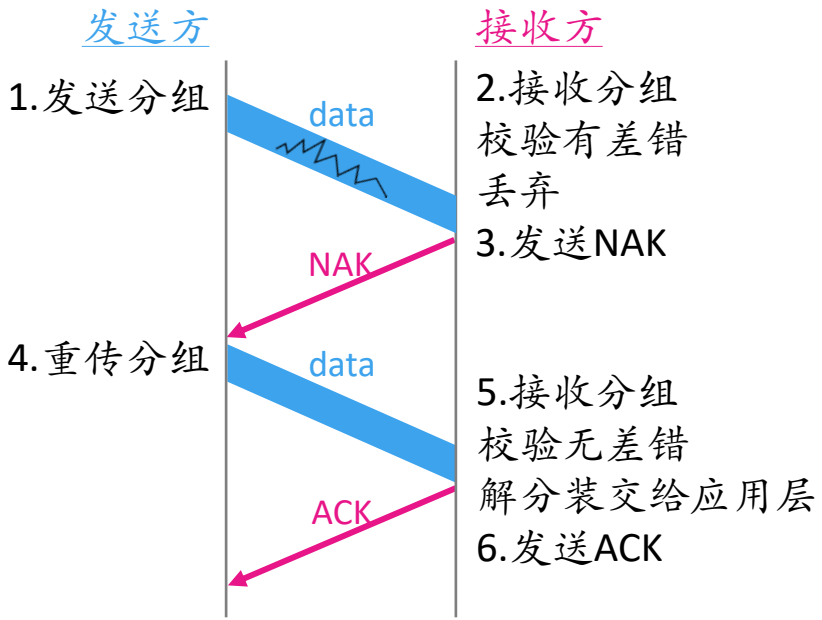
可靠传输—解决差错问题

- 先假设底层信道不丢包，但可能产生差错
- 校验和差错检测
- 检测出差错以后，如何获得正确的报文段呢?
- 解决办法：确认Acknowledgment（ACK）机制

可靠传输—解决差错问题

- 无差错：肯定确认
- 有差错：否定确认

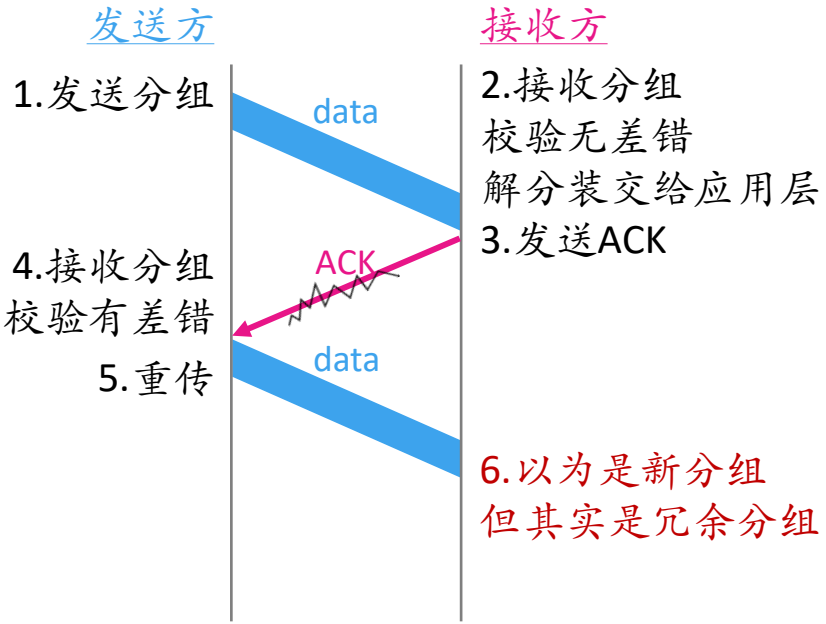




ACK: Acknowledgement
NAK: Negative Acknowledgement

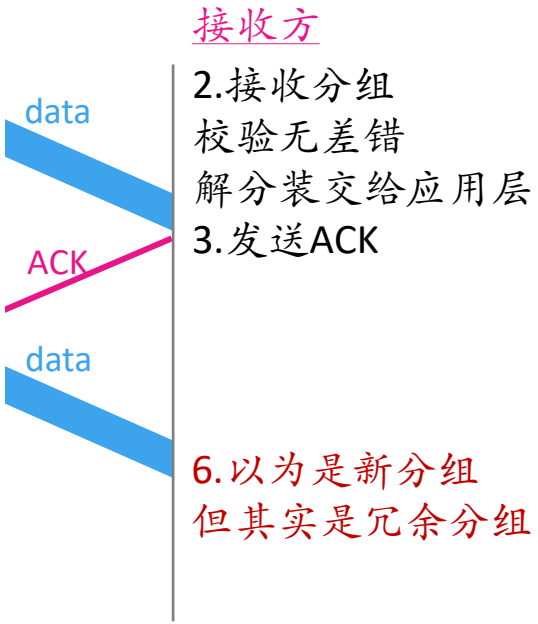
可靠传输—解决差错问题

- 可是，ACK和NAK分组会不会出错呢？
- 当然会，所以也要对确认分组进行差错检测，如果校验发现有差错，就重传分组。



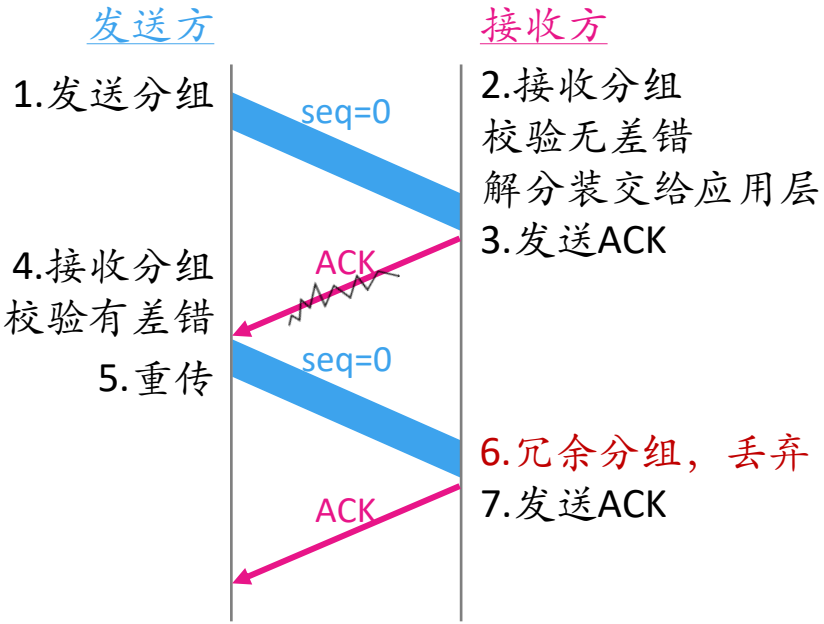
可靠传输—解决差错问题

- 可是，ACK和NAK分组会不会出错呢？
- 当然会，所以也要对确认分组进行差错检测，如果校验发现有差错，就重传分组。
- 误会的原因是：接收方无法知晓发送方的情况。



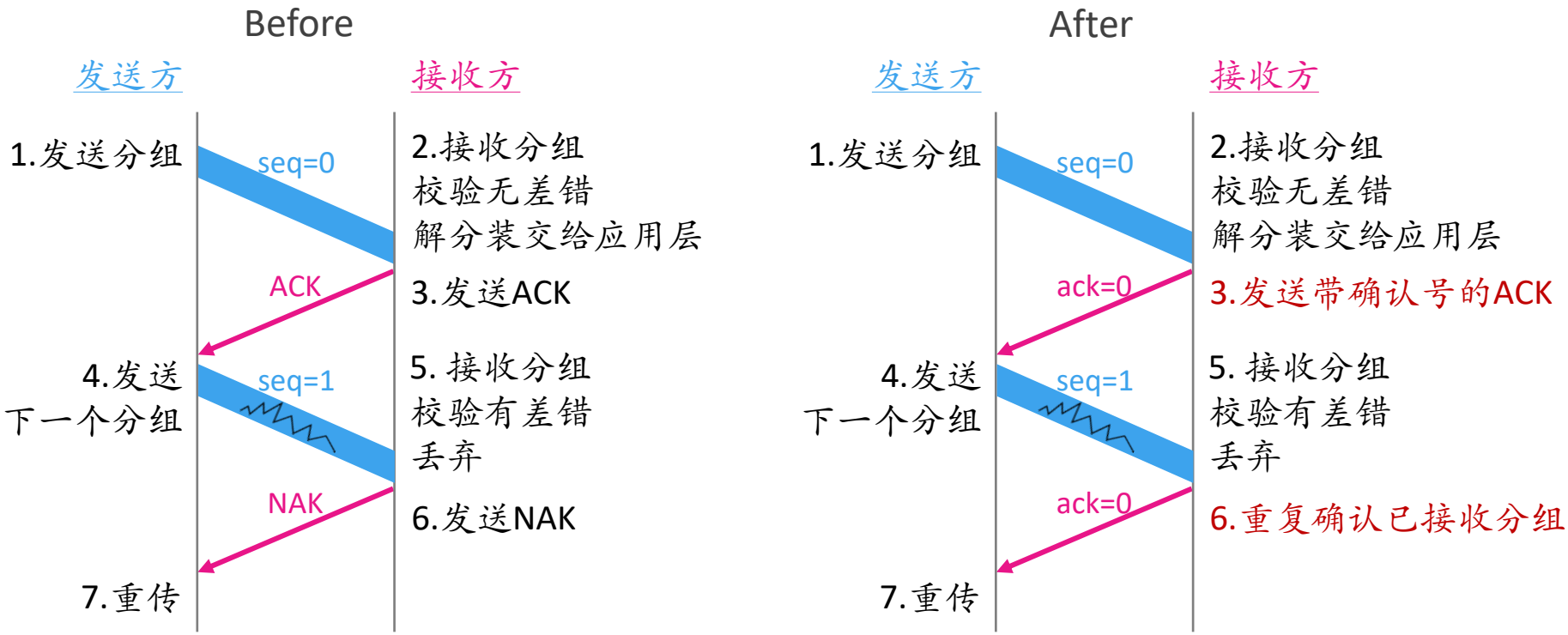
可靠传输—解决差错问题

- 如何区分冗余分组?
- 解决方法：序号



可靠传输—解决差错问题

- 引入确认序号，可以代替NAK。
- 比如收到受损分组时，对上次正确接收的分组发送一个ACK。



第三章知识点汇总

- 理解以下为了解决差错问题而提出的技术
- 校验和、序号、ACK分组和重传

可靠传输讲解内容

- 解决差错问题
- 解决丢包问题
- 停等协议与流水线
- 滑动窗口协议
- 回退N步
- 选择重传
- TCP可靠传输

可靠传输—解决丢包问题

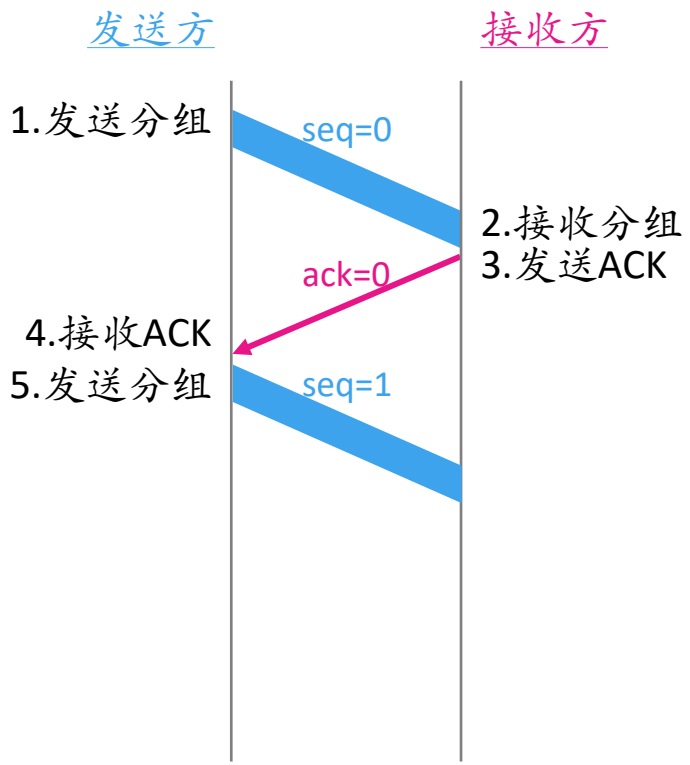
- 假设底层信道丢包
- 怎么检测丢包?
- 丢包后怎么做?
- 注意：数据分组和确认分组都可能会丢失
- 我们让发送方负责检测和恢复丢包工作（这是一种方法但不是唯一的方法）

可靠传输—解决丢包问题

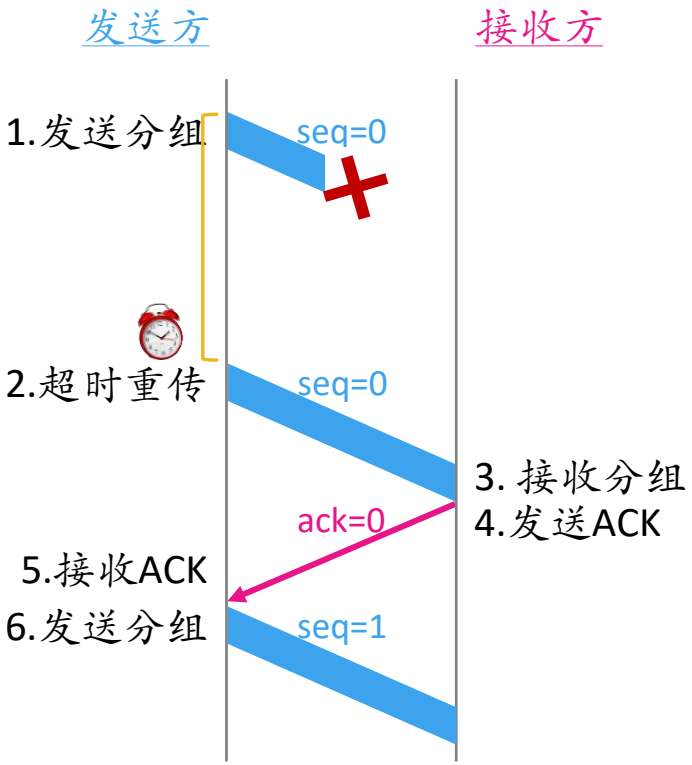
- 怎么检测丢包?
  - 解决方法：定时器机制
  - 发送方等待足够的时间，如果还未收到ACK，则认定为丢包
  - 如果定时器时长太短，误判丢包
  - 如果定时器时长太长，等待时间就很长
- 丢包后怎么做?
  - 重传

可靠传输—解决丢包问题

- 无丢包操作

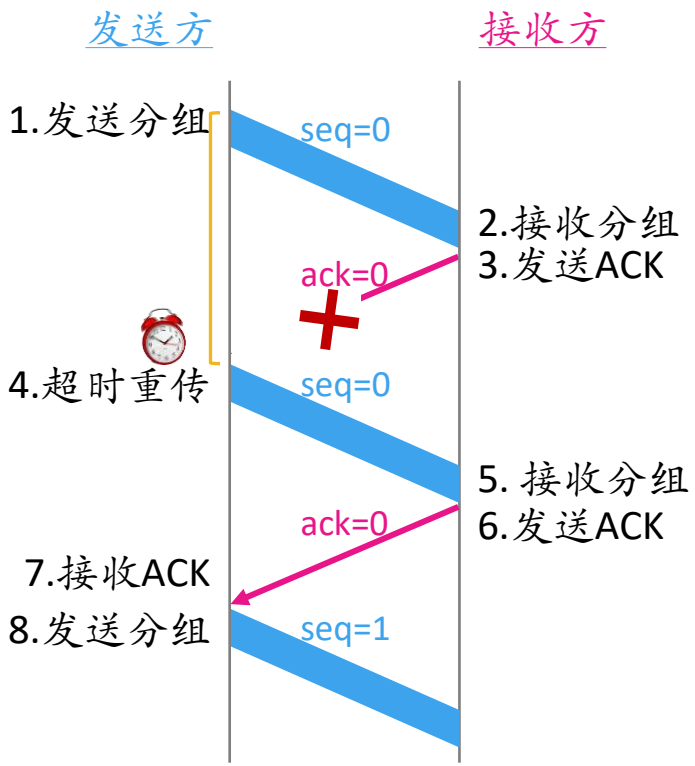


- 分组丢失



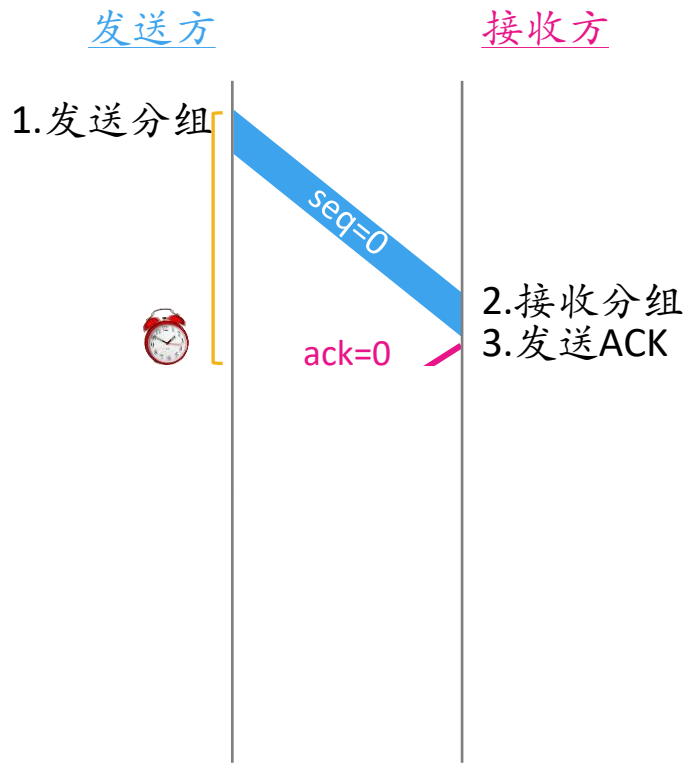
可靠传输—解决丢包问题

- ACK丢失



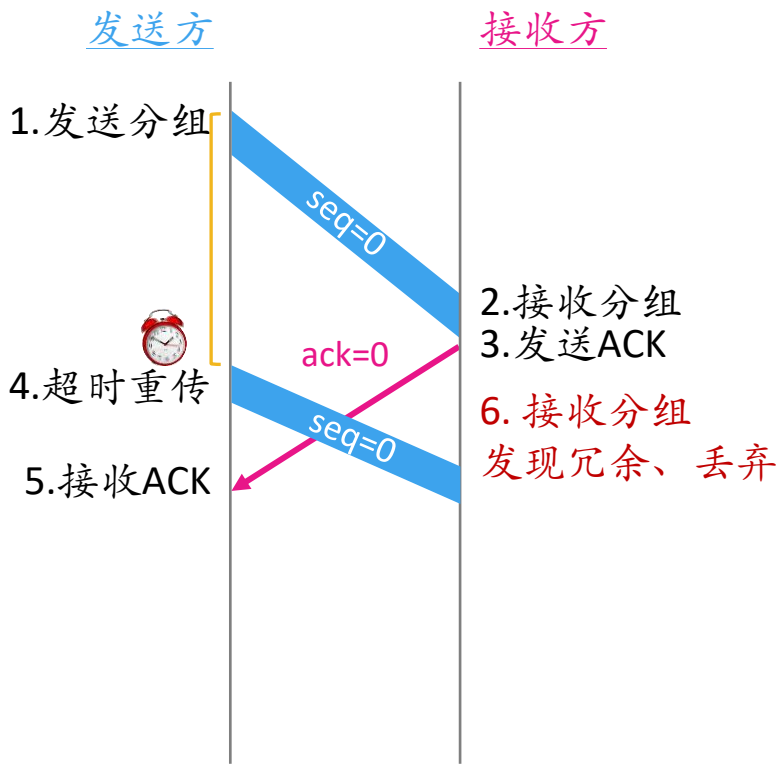
可靠传输—解决丢包问题

- 过早超时



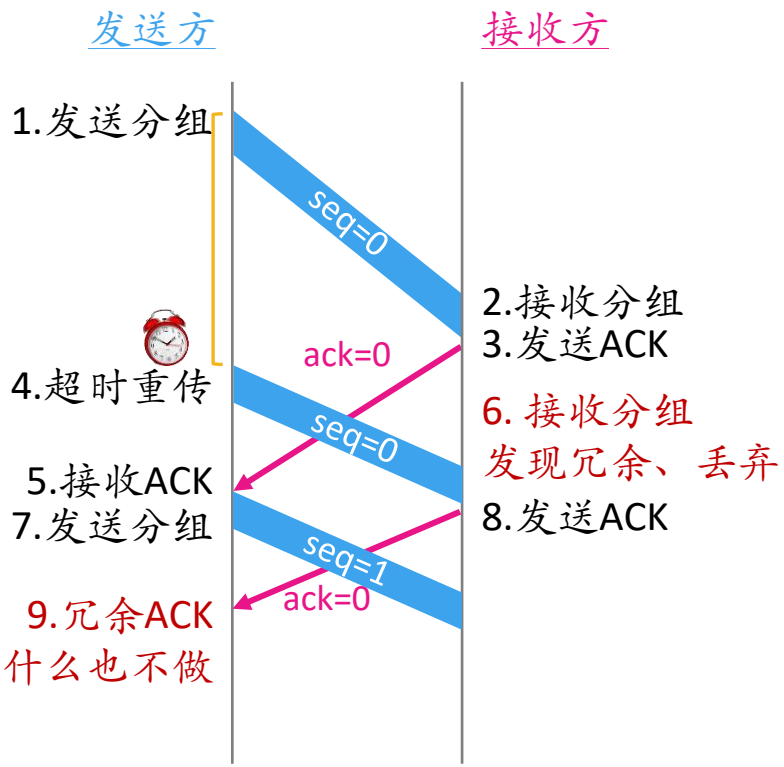
可靠传输—解决丢包问题

- 过早超时



可靠传输—解决丢包问题

- 过早超时



- 总结：
- 分组丢失、ACK丢失、过度延迟，这三种情况在发送方看起来是一样的，动作都是重传
- 序号可以检测冗余

第三章知识点汇总

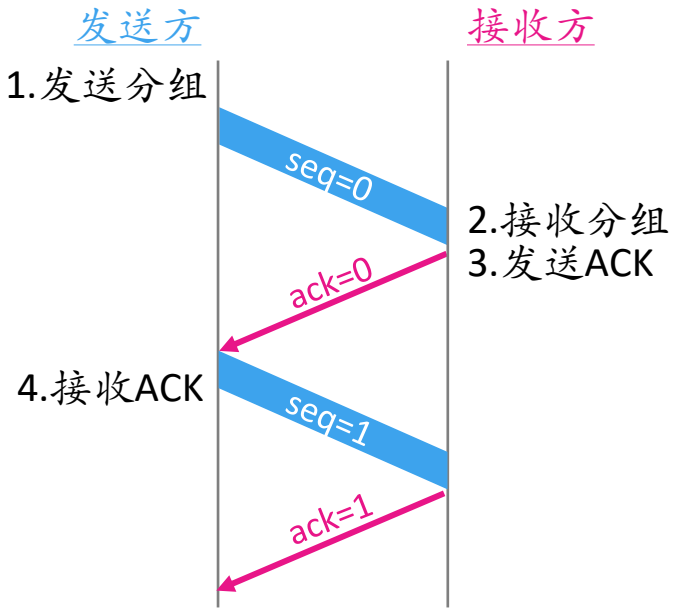
- 理解为了解决丢包问题提出的定时器机制
- 理解不同情况发生时发送方和接收方的处理逻辑

可靠传输讲解内容

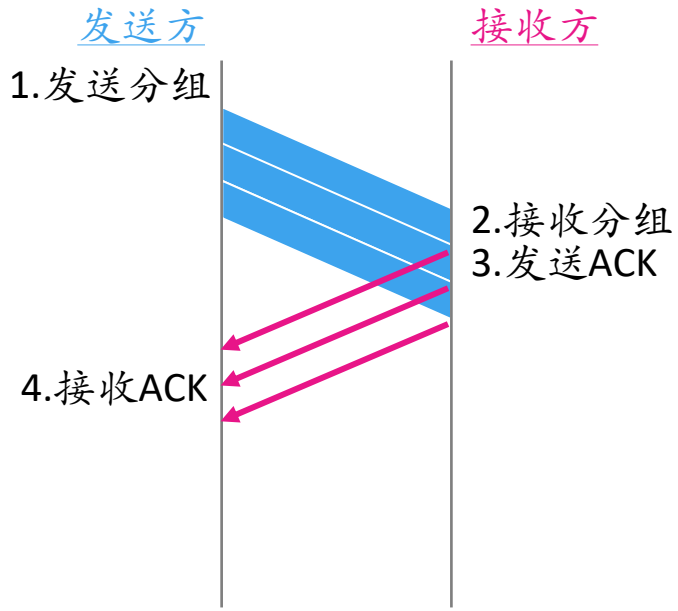
- 解决差错问题
- 解决丢包问题
- 停等协议与流水线协议
- 滑动窗口协议
- 回退N步
- 选择重传
- TCP可靠传输

可靠传输协议

- 停等协议
- 网络里只有一个未确认的分组

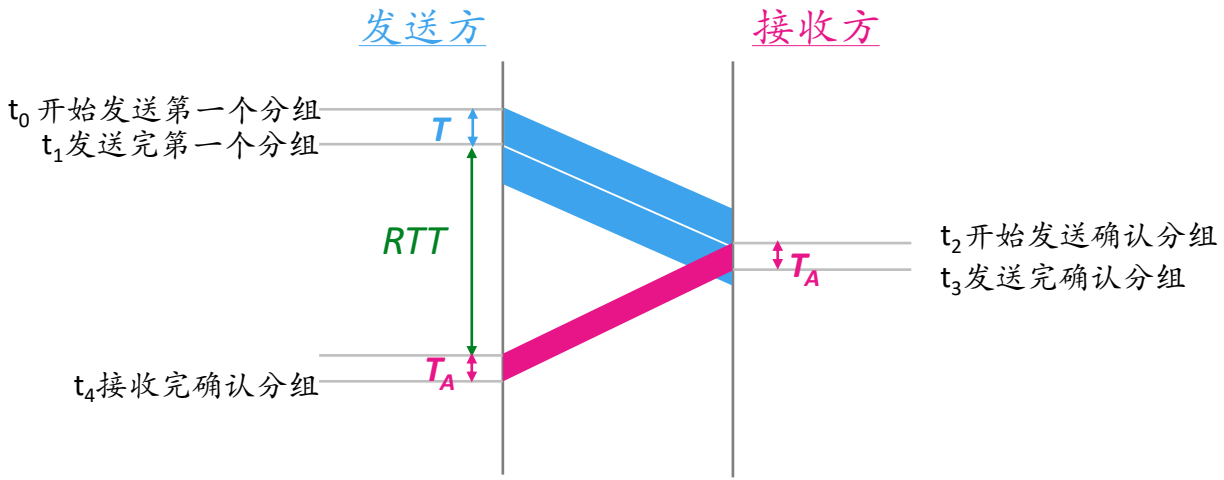


- 流水线协议
- 网络里有N个未确认的分组



信道利用率

$$
\begin{aligned}
&\text{信道利用率} \\
&= \frac{\text{信道被占用的时间}}{\text{周期}} \\
&= \frac{n\text{个分组的传输时间}}{\text{传输一个分组和它所对应确认的时间}} \\
&= \frac{nT}{T + RTT + T_A}
\end{aligned}
$$

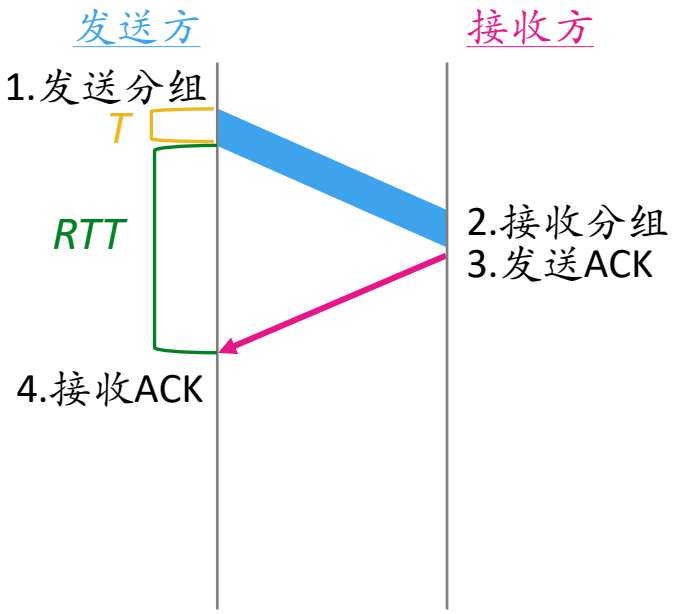


可靠传输—性能

- 忽略ACK传输时间的信道利用率：

$$\frac{T}{RTT+T}$$

- 停等协议



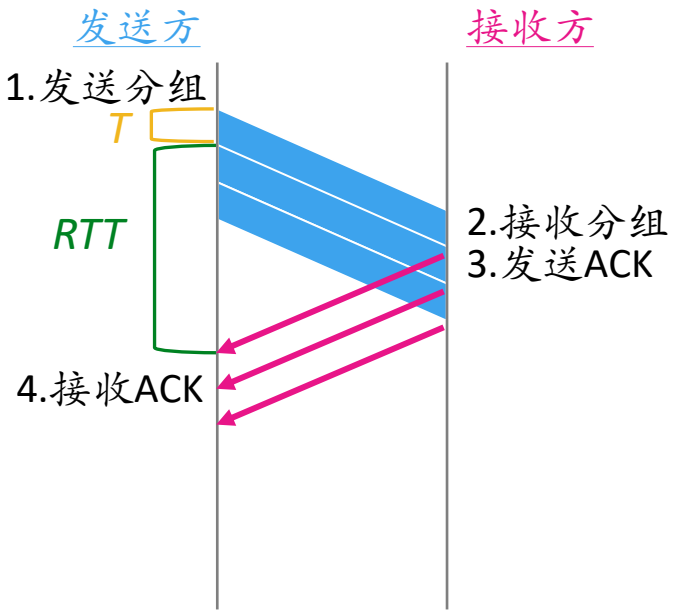
可靠传输—性能

- 忽略ACK传输时间的信道利用率：

$$\frac{3T}{RTT+T}$$

- 流水线提高了信道利用率

- 流水线



习题

- 两个主机之间的距离是L千米，帧长为K比特，单位距离的传播时延为t秒/千米，它们之间的信道容量为R比特/秒，假设处理时延和确认帧的传输时延可以忽略，那么当使用流水线协议时，使得信道利用率最大化的分组数是(　　)。

- A. $\frac{2LtR+K}{K}$
- B. $\frac{2LtR}{K}$
- C. $\frac{2LtR+2K}{K}$
- D. $\frac{2LtR+K}{2K}$

第三章知识点汇总

- 掌握可靠传输的信道利用率的计算方法

可靠传输—流水线

- 必须增加序号范围
- 如何处理差错、丢失、延时过大、乱序等问题?
- 滑动窗口协议
  - 回退N步
  - 选择重传

可靠传输讲解内容

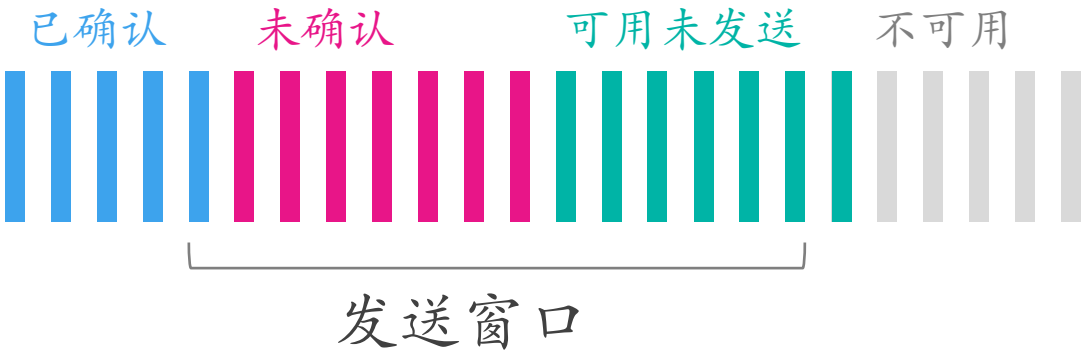
- 解决差错问题
- 解决丢包问题
- 停等协议与流水线协议
- 滑动窗口协议
- 回退N步
- 选择重传
- TCP可靠传输

滑动窗口协议

- 允许未确认的分组数 > 1
- 允许发送的序号范围，称为发送窗口

滑动窗口协议

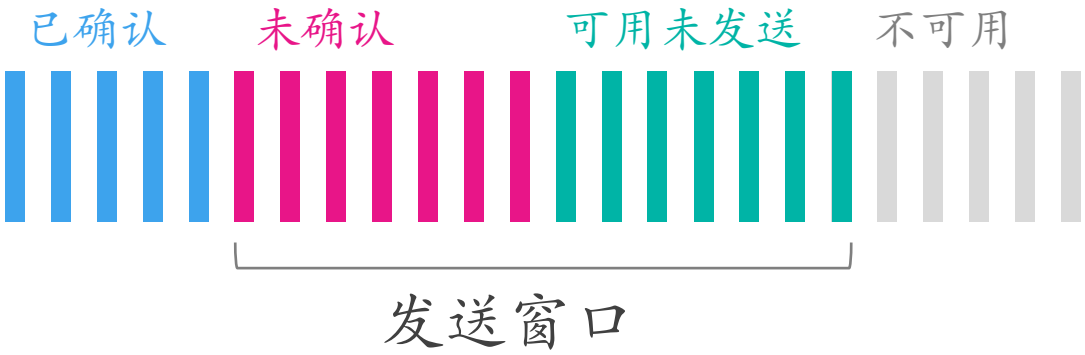
- 允许未确认的分组数 > 1
- 允许发送的序号范围，称为发送窗口



图示：发送方的序号空间

滑动窗口协议

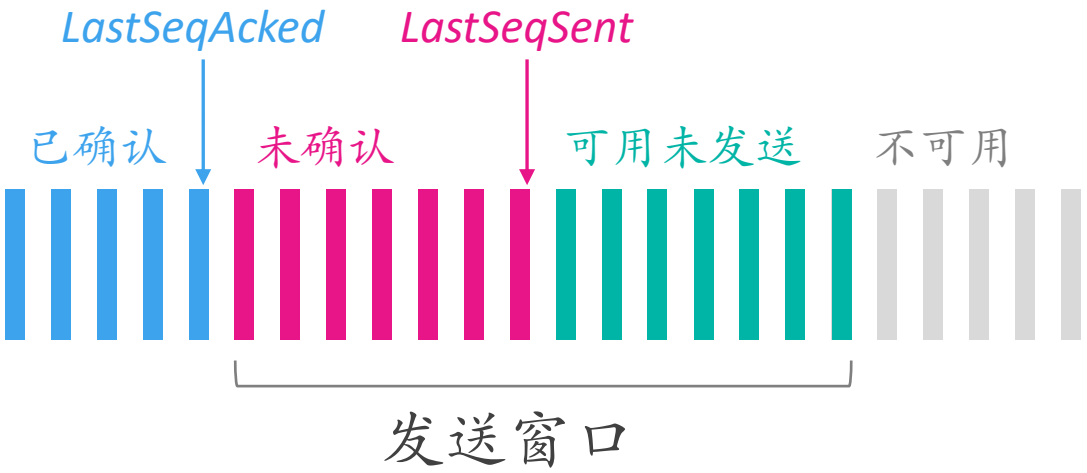
- 允许未确认的分组数 > 1
- 允许发送的序号范围，称为发送窗口



图示：发送方的序号空间

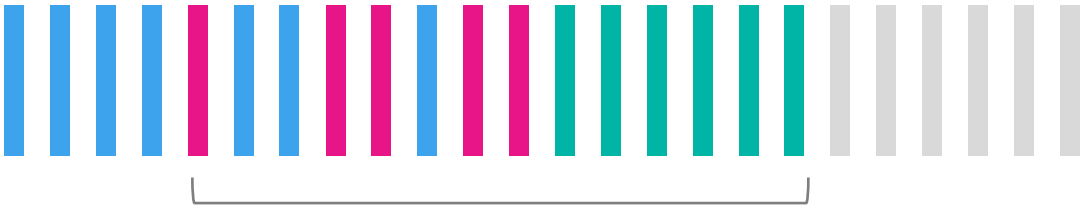
滑动窗口协议

- 允许未确认的分组数 > 1
- 允许发送的序号范围，称为发送窗口
- *LastSeqSent – LastSeqAcked <= wnd*



图示：发送方的序号空间

滑动窗口协议

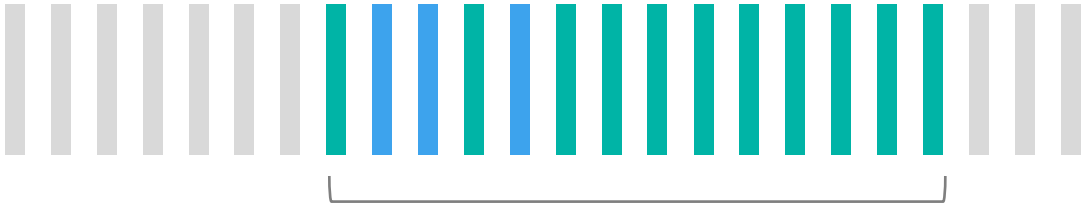


已确认
未确认
可用未发送
不可用

发送窗口

图示：发送方看到的序号空间

- 允许接收的序号范围，称为接收窗口

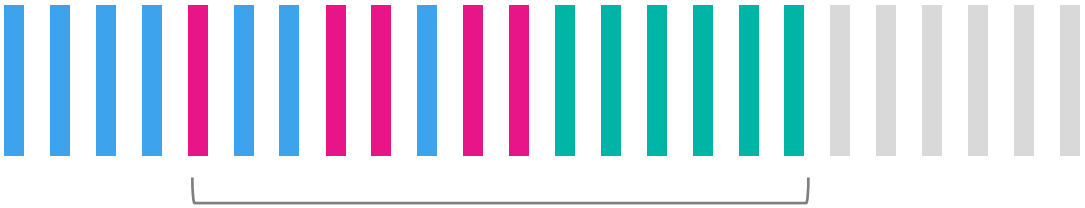


失序(已缓存)
可接收(窗口内)
不可用

接收窗口

图示：接收方看到的序号空间

滑动窗口协议

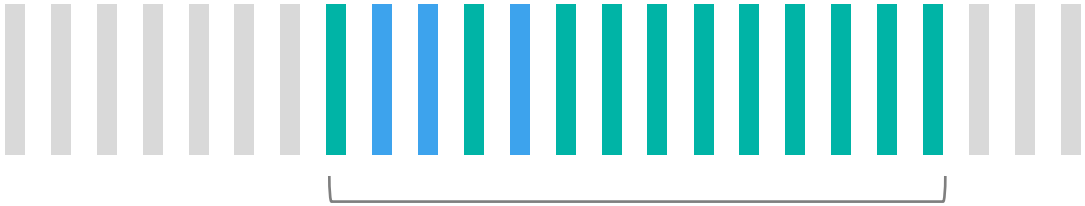


已确认
未确认
可用未发送
不可用

发送窗口

图示：发送方看到的序号空间

发送方和接收方是独立的，不同步的



失序(已缓存)
可接受(窗口内)
不可用

接收窗口

图示：接收方看到的序号空间

滑动窗口协议

- 如果序号空间为[0, M-1]
- 发送窗口 + 接收窗口 <= M

- 例如：
  - 当M = 4，接收窗口是1，则发送窗口最大是3.
  - 当M = 8，接收窗口是4，则发送窗口最大是4.

滑动窗口协议

- 如果发送窗口 + 接收窗口 > M，则会发生接收端错将重传分组当成新分组的错误事故。

滑动窗口协议

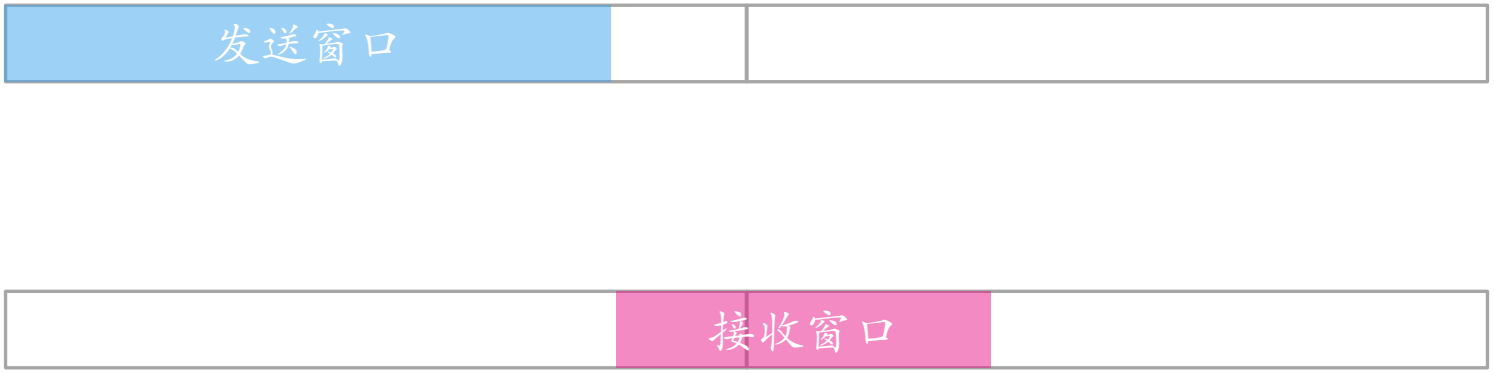
- 如果发送窗口 + 接收窗口 > M，则会发生接收端错将重传分组当成新分组的错误事故。

- 1、发送端发送窗口里的所有分组

滑动窗口协议

- 如果发送窗口 + 接收窗口 > M，则会发生接收端错将重传分组当成新分组的错误事故。

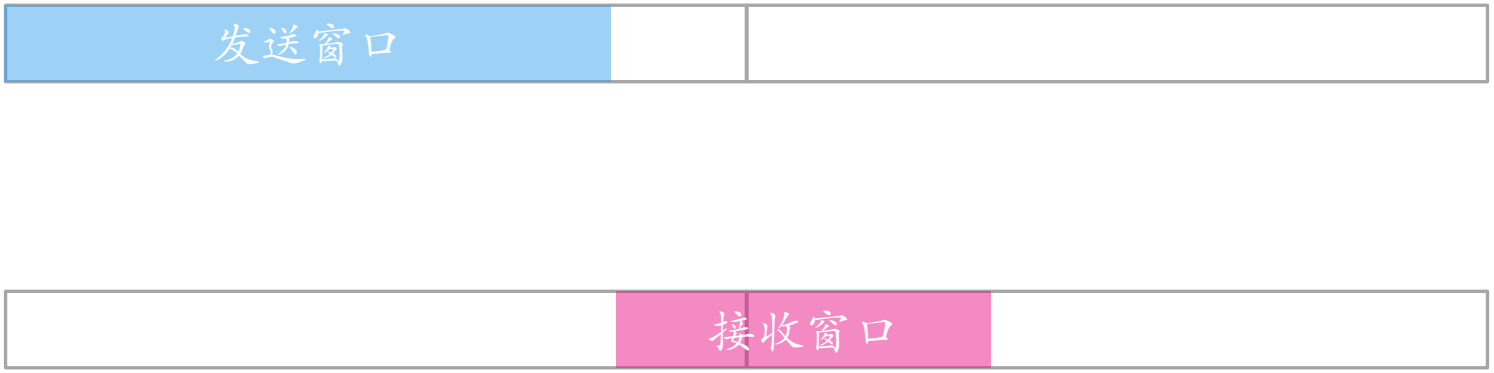
- 2、接收端正常接收所有分组，滑动窗口



滑动窗口协议

- 如果发送窗口 + 接收窗口 > M，则会发生接收端错将重传分组当成新分组的错误事故。

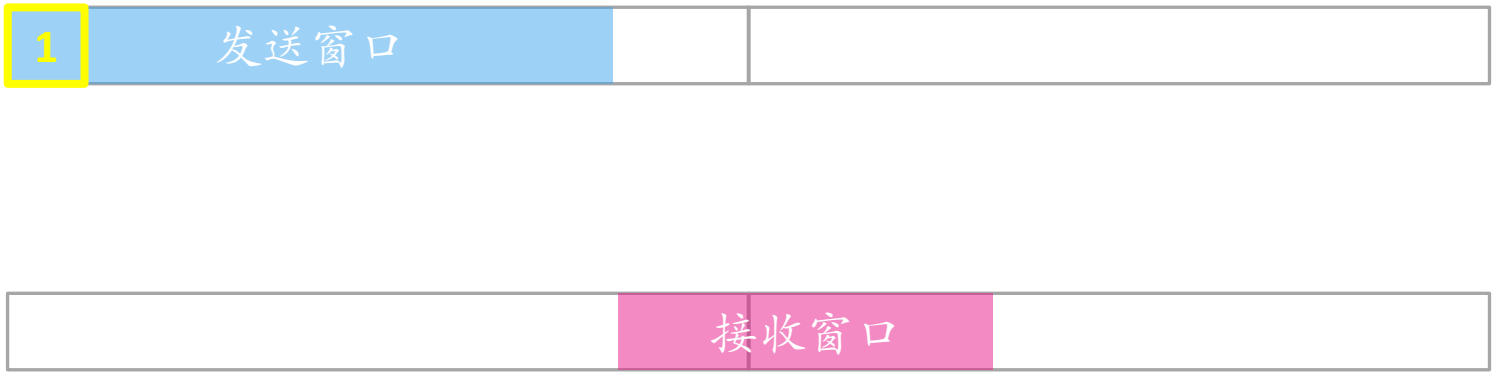
- 2、接收端正常接收所有分组，滑动窗口
- 注意此时接收窗口覆盖了部分发送窗口的序号



滑动窗口协议

- 如果发送窗口 + 接收窗口 > M，则会发生接收端错将重传分组当成新分组的错误事故。

- 3、假如序号1的确认分组丢失了，导致超时重传



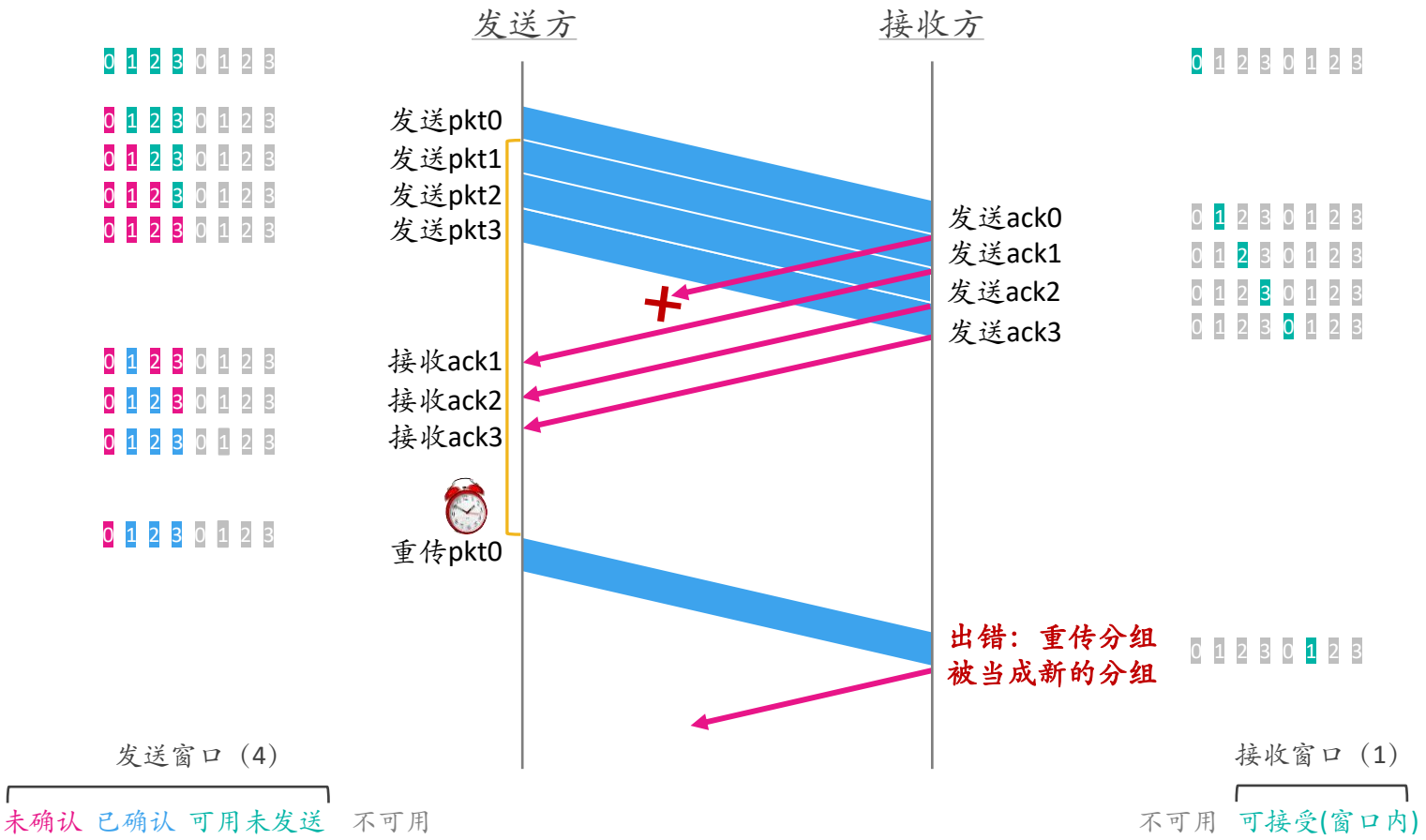
滑动窗口协议

- 如果发送窗口 + 接收窗口 > M，则会发生接收端错将重传分组当成新分组的错误事故。

- 4、接收端错误地将该分组当成了新的分组

滑动窗口协议

- 举例：当M为4，接收窗口为1，发送窗口为4时会出错



第三章知识点汇总

- 了解滑动窗口协议的工作原理
- 了解发送窗口、接收窗口、序号空间之间的关系

可靠传输讲解内容

- 解决差错问题
- 解决丢包问题
- 停等协议与流水线协议
- 滑动窗口协议
- 回退N步
- 选择重传
- TCP可靠传输

回退N步

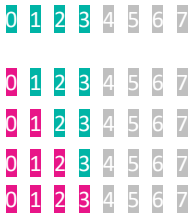
- 允许未确认的分组数最大为N
- 接收方不缓存失序分组，接收窗口为1，设计简单
- 失序分组都被丢弃，需要重传
- 超时：重传窗口中的所有分组
- 累积确认：对序号为n的分组的确认，表示接收方已正确接收到序号小于等于n的所有分组。
- 定时器：针对整个窗口，每次窗口滑动就重启定时器

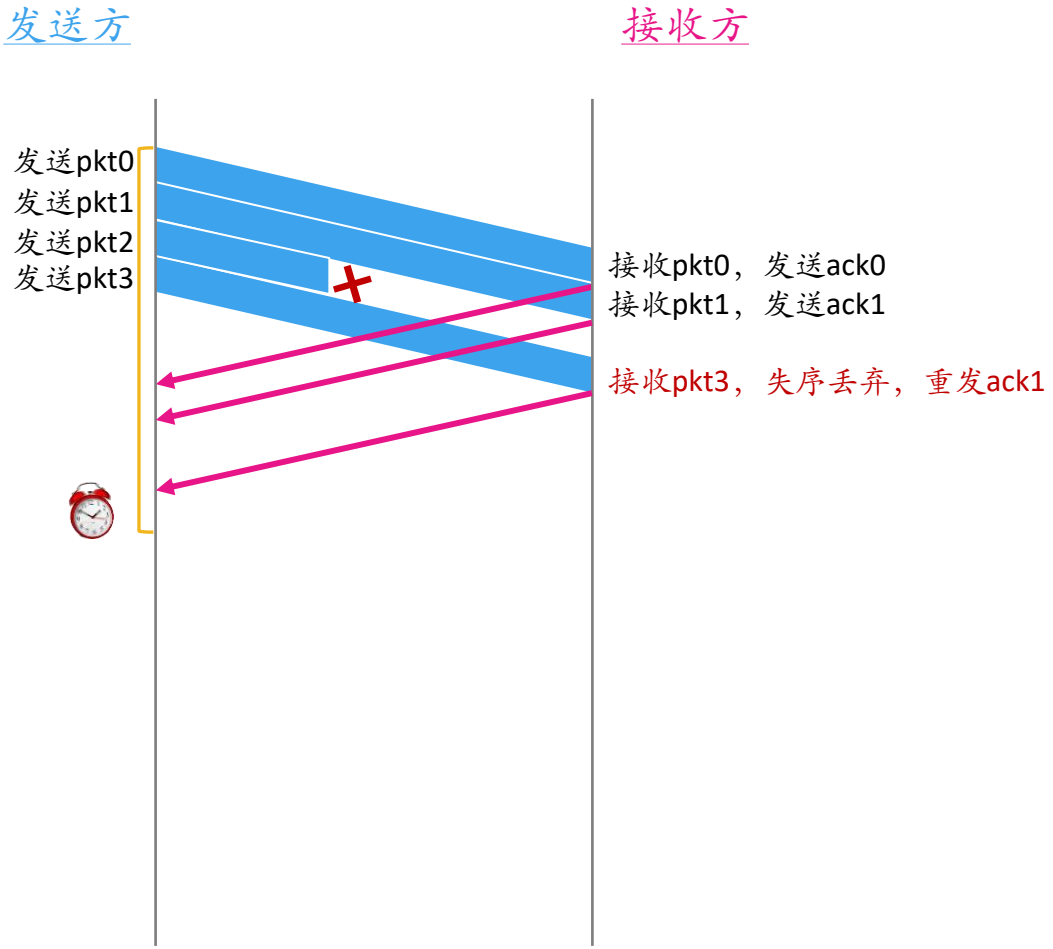
回退N步

- 序号空间 = M
- 接收窗口 = 1
- 发送窗口 + 接收窗口 <= M
- 发送窗口 <= M - 1

回退N步

- 举例： N=4

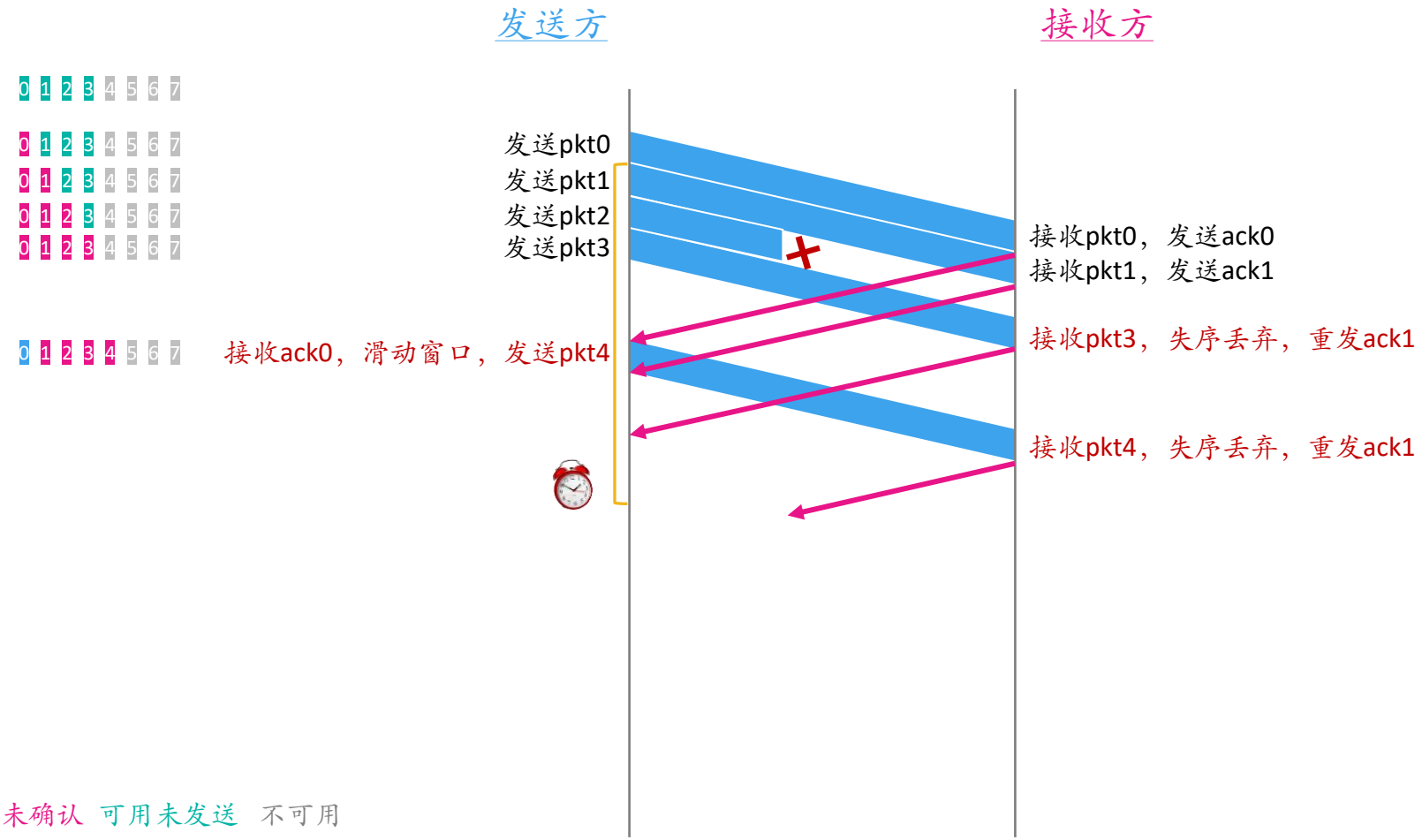




已确认 未确认 可用未发送 不可用

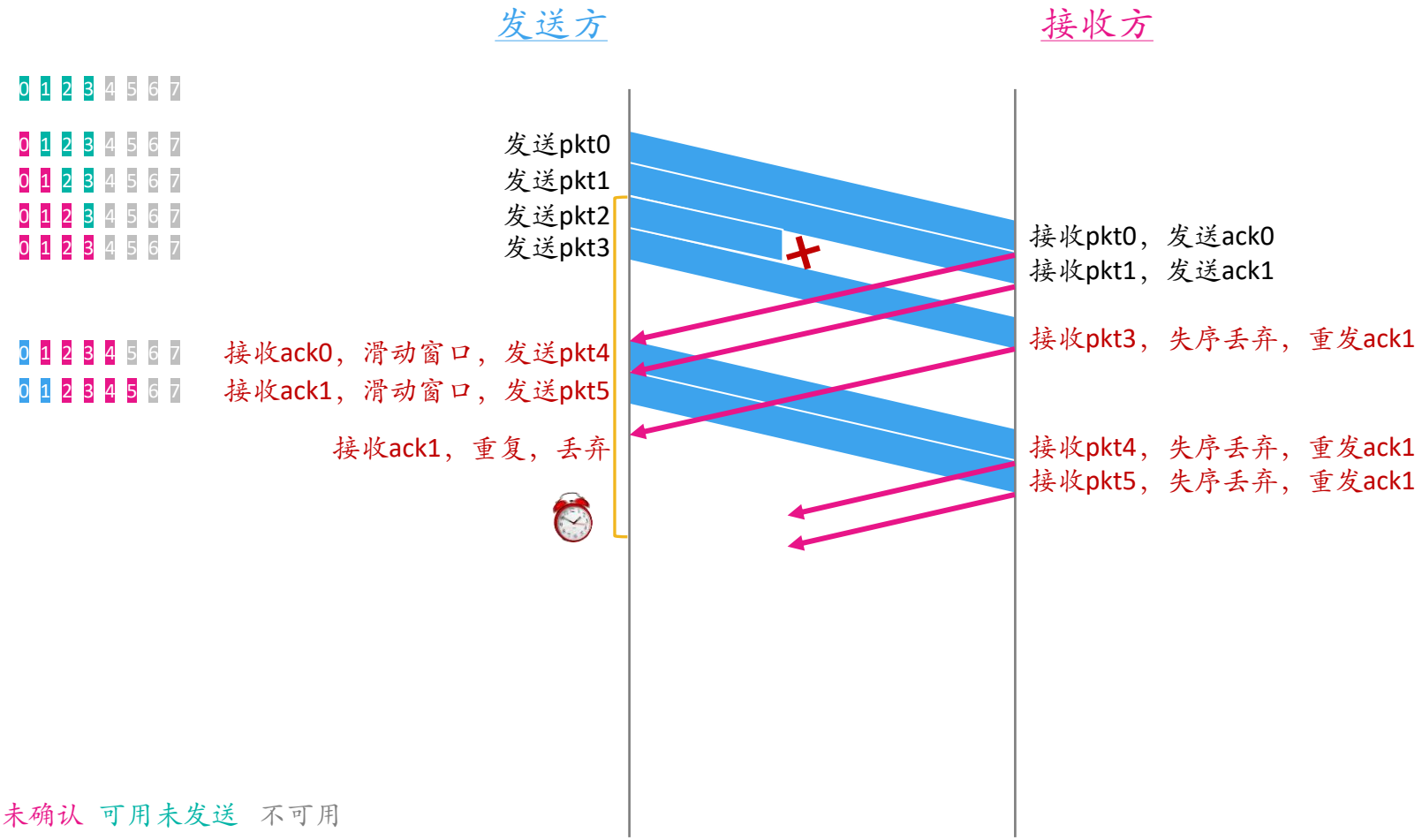
回退N步

- 举例： N=4


发送方
接收方
0 1 2 3 4 5 6 7
0 1 2 3 4 5 6 7
0 1 2 3 4 5 6 7
0 1 2 3 4 5 6 7
0 1 2 3 4 5 6 7
发送pkt0
发送pkt1
发送pkt2
发送pkt3
接收pkt0，发送ack0
接收pkt1，发送ack1
接收pkt3，失序丢弃，重发ack1
接收pkt4，失序丢弃，重发ack1
0 1 2 3 4 5 6 7
接收ack0，滑动窗口，发送pkt4
已确认 未确认 可用未发送 不可用

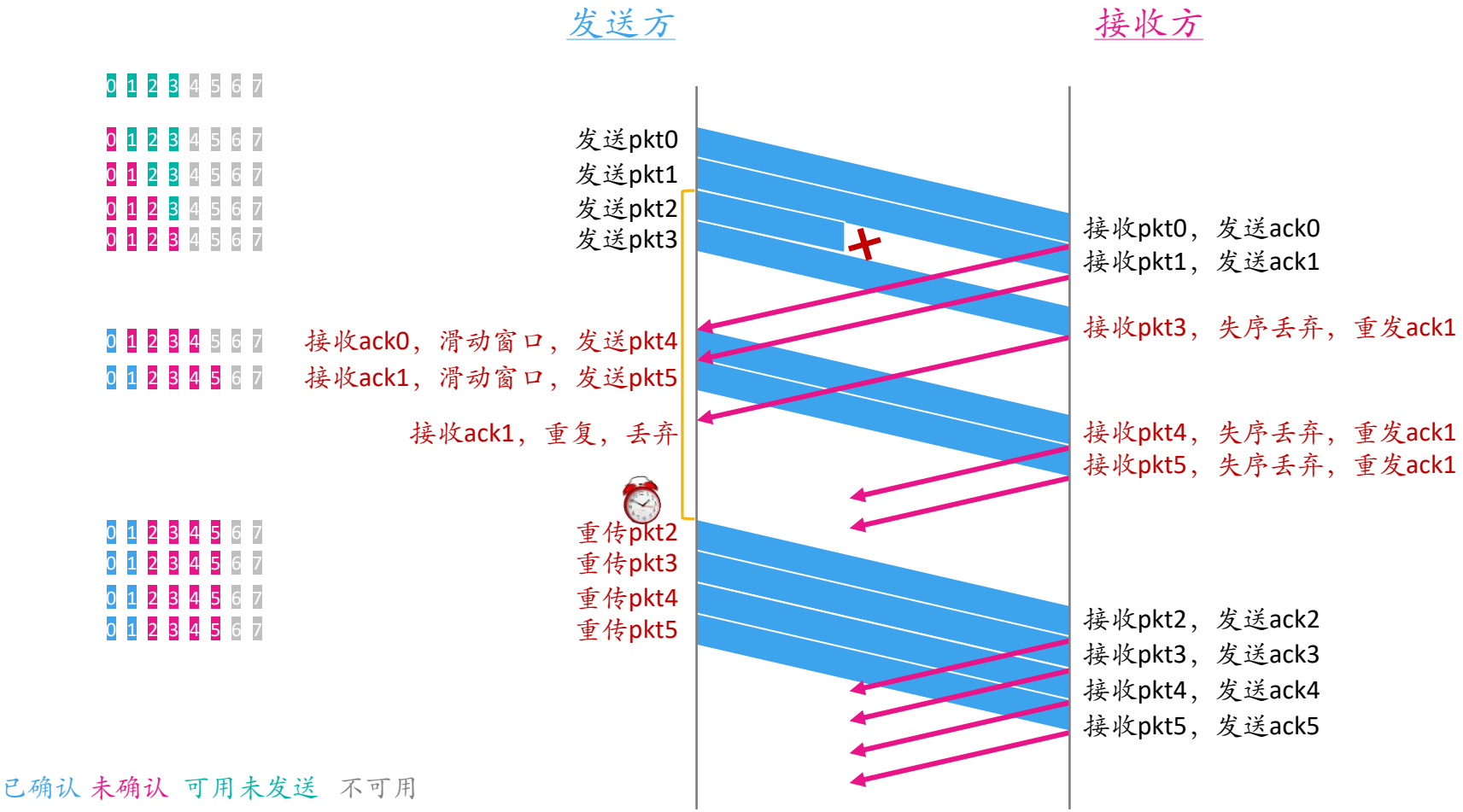
回退N步

- 举例： N=4


发送方
接收方
0 1 2 3 4 5 6 7
发送pkt0
发送pkt1
发送pkt2
发送pkt3
接收pkt0，发送ack0
接收pkt1，发送ack1
接收pkt3，失序丢弃，重发ack1
接收ack0，滑动窗口，发送pkt4
接收ack1，滑动窗口，发送pkt5
接收ack1，重复，丢弃
接收pkt4，失序丢弃，重发ack1
接收pkt5，失序丢弃，重发ack1
已确认 未确认 可用未发送 不可用

回退N步

- 举例： N=4


发送方
接收方
0 1 2 3 4 5 6 7
发送pkt0
发送pkt1
发送pkt2
发送pkt3
接收pkt0，发送ack0
接收pkt1，发送ack1
接收pkt3，失序丢弃，重发ack1
接收ack0，滑动窗口，发送pkt4
接收ack1，滑动窗口，发送pkt5
接收pkt4，失序丢弃，重发ack1
接收pkt5，失序丢弃，重发ack1
接收ack1，重复，丢弃
重传pkt2
重传pkt3
重传pkt4
重传pkt5
接收pkt2，发送ack2
接收pkt3，发送ack3
接收pkt4，发送ack4
接收pkt5，发送ack5
已确认 未确认 可用未发送 不可用

回退N步

- 允许未确认的分组数最大为*N*
- 接收方不缓存失序分组，接收窗口为1，设计简单
- 失序分组都被丢弃，需要重传
- 超时：重传窗口中的所有分组
- 累积确认：对序号为*n*的分组的确认，表示接收方已正确接收到序号小于等于*n*的所有分组。
- 定时器：针对整个窗口，每次窗口滑动就重启定时器

第三章知识点汇总

- 理解回退N步（滑动窗口）协议的工作原理

可靠传输讲解内容

- 解决差错问题
- 解决丢包问题
- 停等协议与流水线协议
- 滑动窗口协议
- 回退N步
- 选择重传
- TCP可靠传输

选择重传

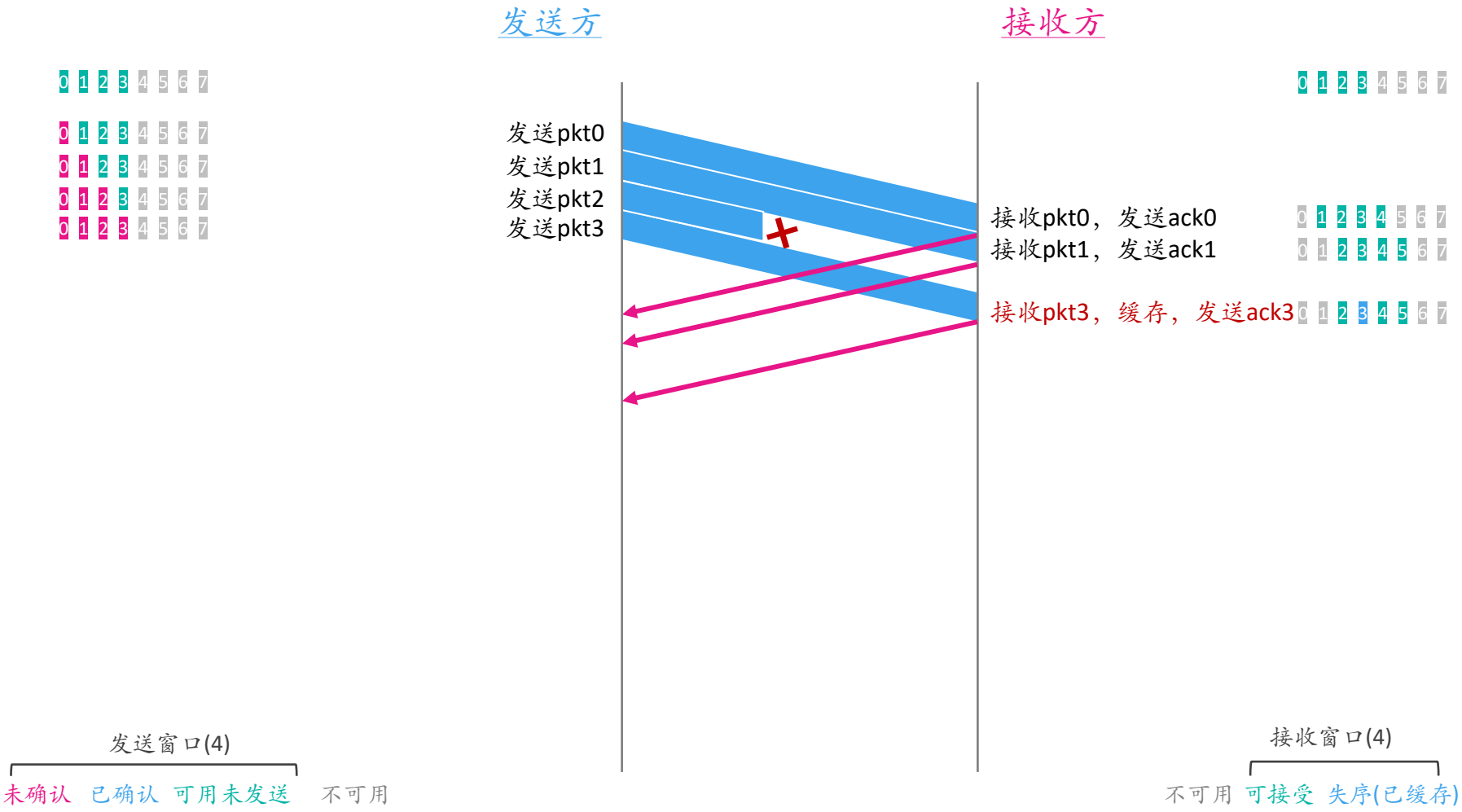
- 允许未确认的分组数 > 1
- 接收方缓存失序分组，接收窗口>1，设计复杂
- 失序分组被缓存，不需要重传
- 超时：只重传没收到确认的分组
- 逐个确认：对序号为*n*的分组的确认，表示接收方已正确接收该分组。
- 定时器：针对每个分组

选择重传

- 序号空间 = M
- 发送窗口 + 接收窗口 <= M
- 接收窗口 = 发送窗口
- 发送窗口 <= M/2

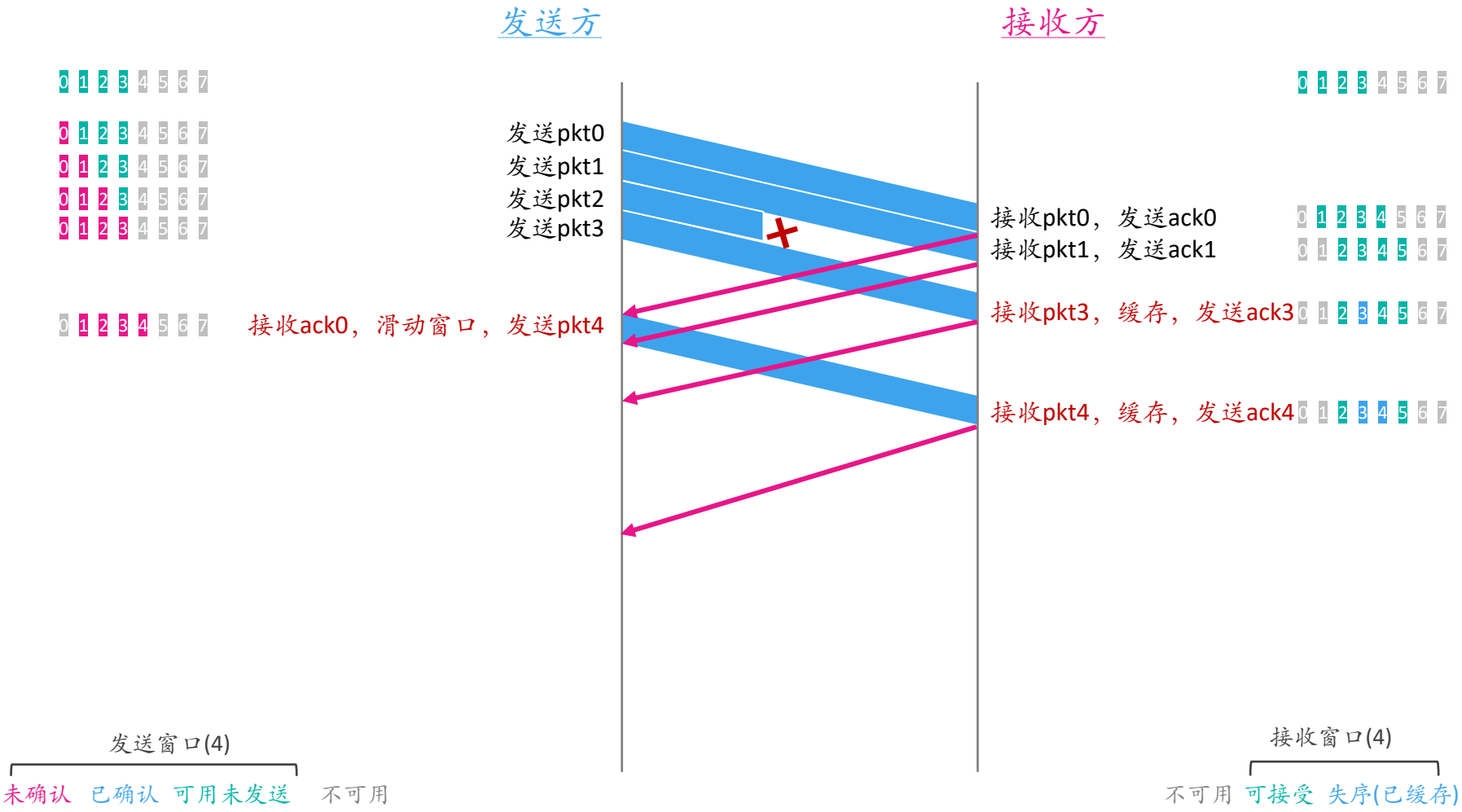
选择重传

- 举例：


发送方
接收方
0 1 2 3 4 5 6 7
0 1 2 3 4 5 6 7
0 1 2 3 4 5 6 7
0 1 2 3 4 5 6 7
0 1 2 3 4 5 6 7
发送pkt0
发送pkt1
发送pkt2
发送pkt3
接收pkt0，发送ack0
接收pkt1，发送ack1
接收pkt3，缓存，发送ack3
0 1 2 3 4 5 6 7
0 1 2 3 4 5 6 7
0 1 2 3 4 5 6 7
0 1 2 3 4 5 6 7
发送窗口(4)
未确认 已确认 可用未发送 不可用
接收窗口(4)
不可用 可接受 失序(已缓存)

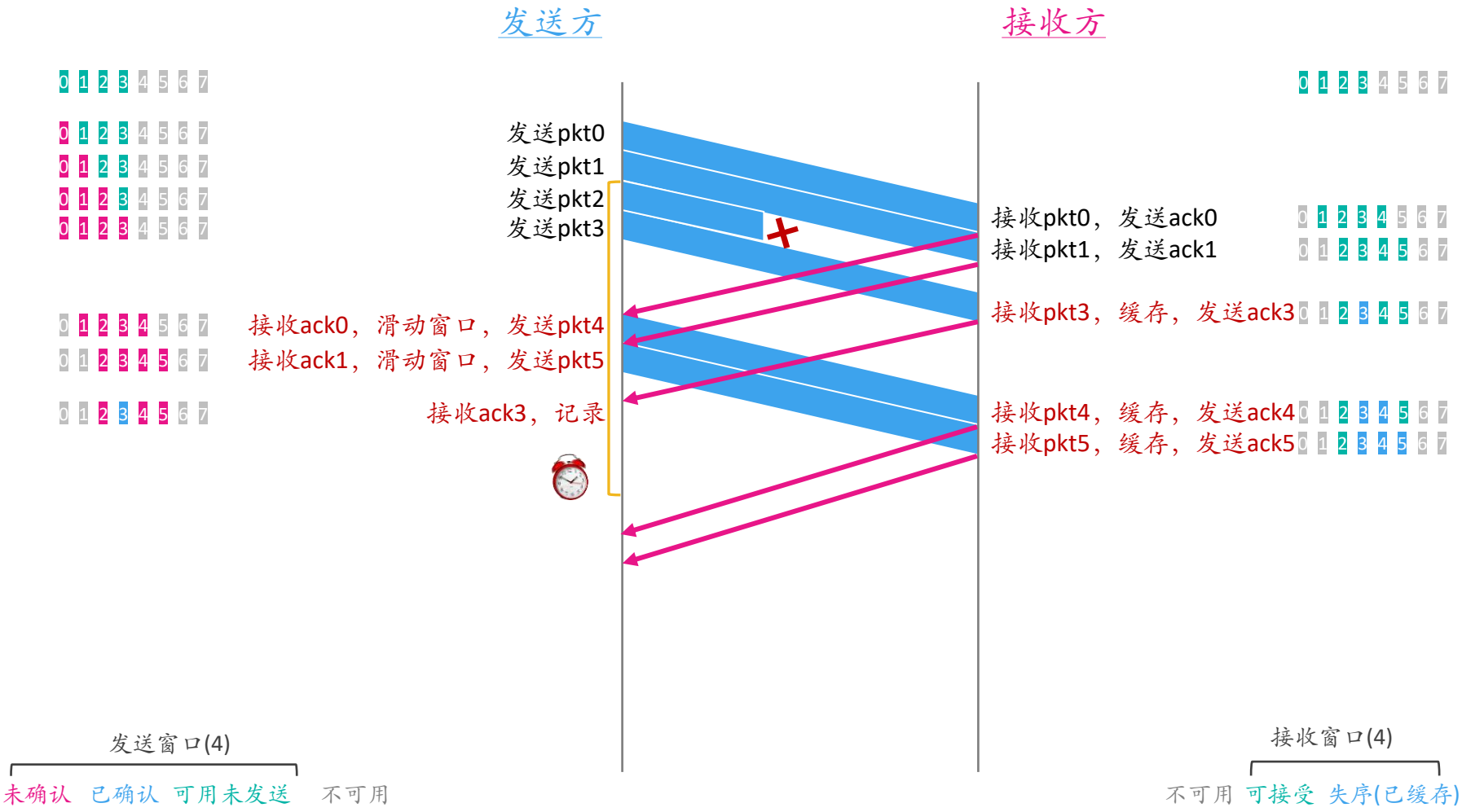
选择重传

- 举例：


发送方
接收方
0 1 2 3 4 5 6 7
发送pkt0
发送pkt1
发送pkt2
发送pkt3
接收pkt0，发送ack0
接收pkt1，发送ack1
接收pkt3，缓存，发送ack3
接收pkt4，缓存，发送ack4
接收ack0，滑动窗口，发送pkt4
发送窗口(4)
未确认 已确认 可用未发送 不可用
接收窗口(4)
不可用 可接受 失序(已缓存)

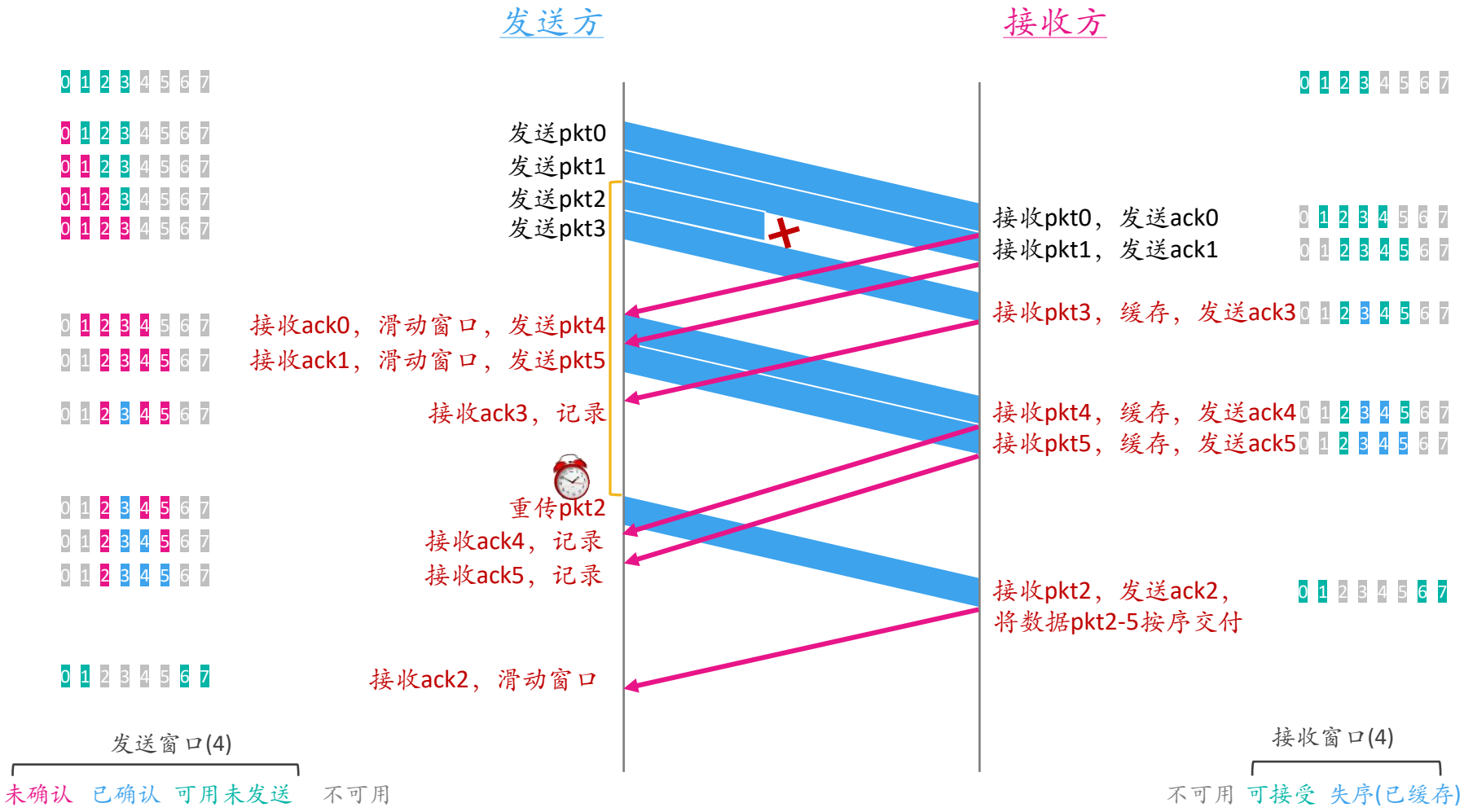
选择重传

- 举例：


发送方
接收方
发送pkt0
发送pkt1
发送pkt2
发送pkt3
接收pkt0，发送ack0
接收pkt1，发送ack1
接收pkt3，缓存，发送ack3
接收ack0，滑动窗口，发送pkt4
接收ack1，滑动窗口，发送pkt5
接收pkt4，缓存，发送ack4
接收pkt5，缓存，发送ack5
接收ack3，记录
发送窗口(4)
未确认 已确认 可用未发送 不可用
接收窗口(4)
不可用 可接受 失序(已缓存)

选择重传

- 举例：


发送方
接收方
发送pkt0
发送pkt1
发送pkt2
发送pkt3
接收pkt0，发送ack0
接收pkt1，发送ack1
接收pkt3，缓存，发送ack3
接收ack0，滑动窗口，发送pkt4
接收ack1，滑动窗口，发送pkt5
接收pkt4，缓存，发送ack4
接收pkt5，缓存，发送ack5
接收ack3，记录
重传pkt2
接收ack4，记录
接收ack5，记录
接收pkt2，发送ack2，
将数据pkt2-5按序交付
接收ack2，滑动窗口
发送窗口(4)
未确认 已确认 可用未发送 不可用
接收窗口(4)
不可用 可接受 失序(已缓存)

选择重传

- 允许未确认的分组数 > 1
- 接收方缓存失序分组，接收窗口>1，设计复杂
- 失序分组被缓存，不需要重传
- 超时：只重传没收到确认的分组
- 逐个确认：对序号为*n*的分组的确认，表示接收方已正确接收该分组。
- 定时器：针对每个分组

回退N步vs选择重传

- 允许未确认的分组数＞1
- 接收方不缓存失序分组，接收窗口=1，设计简单
- 超时：重传窗口中的所有分组
- 累积确认：对序号为n的分组的确认，表示接收方已正确接收到序号小于等于n的所有分组。
- 定时器：针对整个窗口

- 允许未确认的分组数＞1
- 接收方缓存失序分组，接收窗口>1，设计复杂
- 超时：只重传没收到确认的分组
- 逐个确认：对序号为n的分组的确认，表示接收方已正确接收该分组。
- 定时器：针对每个分组

第三章知识点汇总

- 理解选择重传协议的工作原理
- 理解回退N步与选择重传两种机制的区别

可靠传输讲解内容

- 解决差错问题
- 解决丢包问题
- 停等协议与流水线协议
- 滑动窗口协议
- 回退N步
- 选择重传
- TCP可靠传输

TCP可靠传输

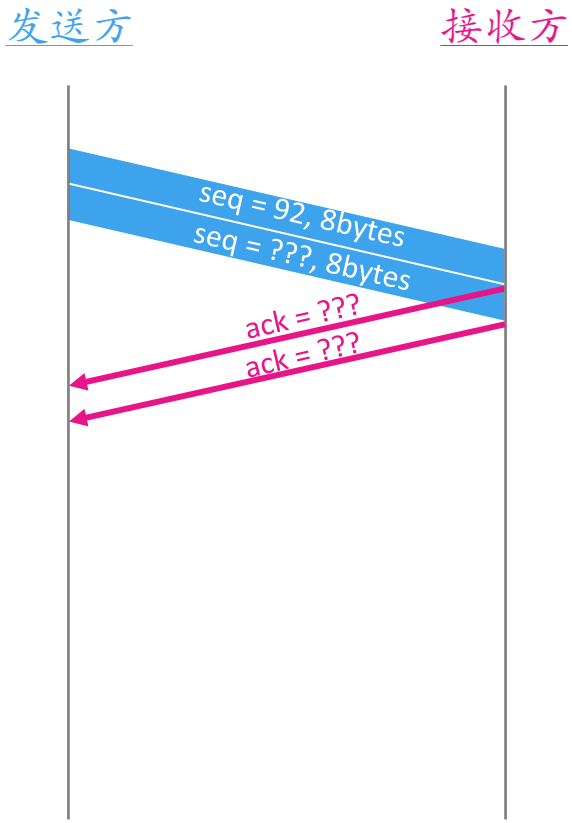
- 确保一个进程发送的字节流和其通信进程接收的字节流完全相同
- 无损坏、无间隙、非冗余、按序

TCP可靠传输

- 利用了校验和、确认、序号、定时器、流水线等技术
- 序号：基于字节流，报文段中数据第一个字节在字节流中的位置
- 确认号：已确认接收数据的最大序号加一，含义是：期望从对方收到的下一个字节的序号
- 确认机制：累积确认（类似回退N步）
- 重传：只重传丢失或损坏的分组（类似选择重传）
- 定时器：针对窗口中最早未确认的分组
- 回退N步和选择重传的混合体

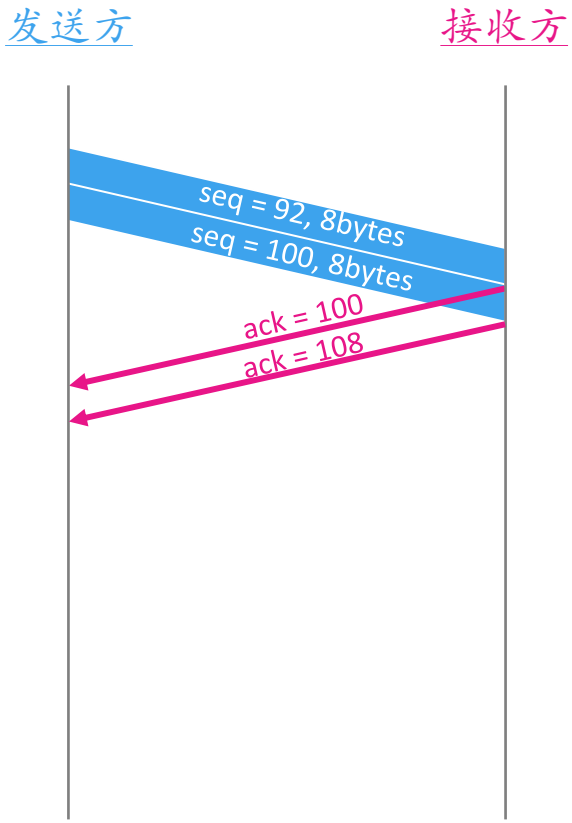
TCP可靠传输

- 无丢包



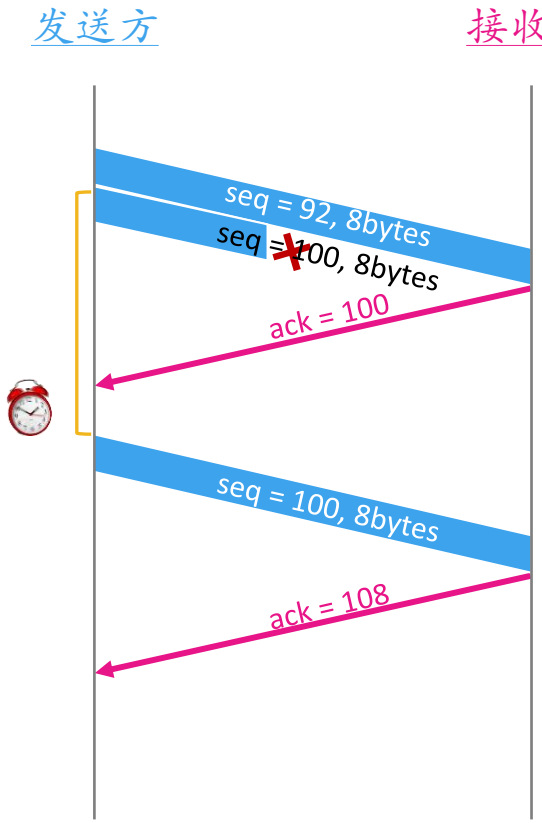
TCP可靠传输

- 无丢包



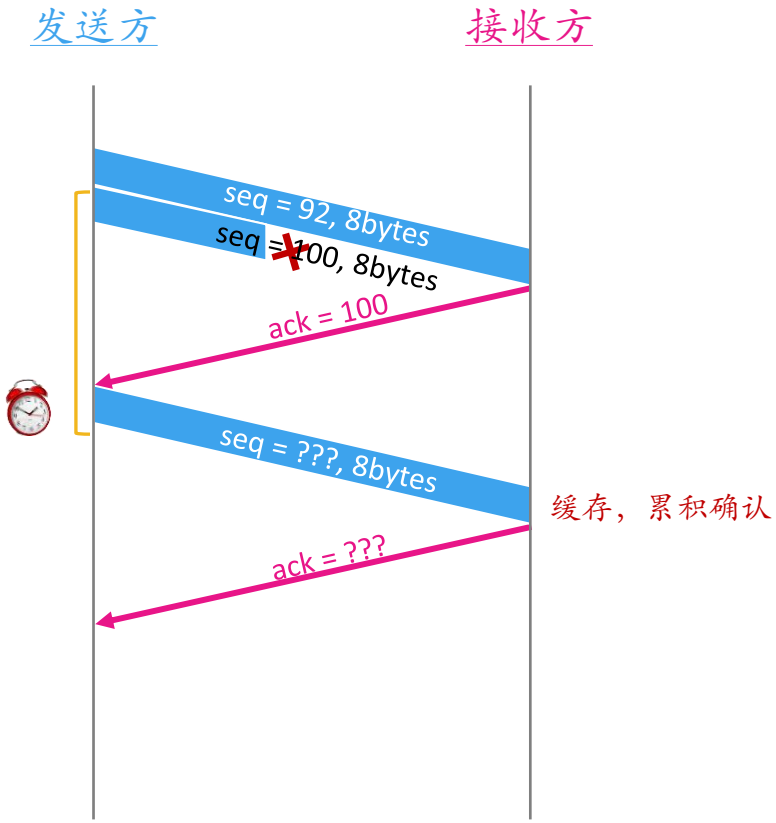
TCP可靠传输

- 数据丢失
  - 假设没有其他数据发送



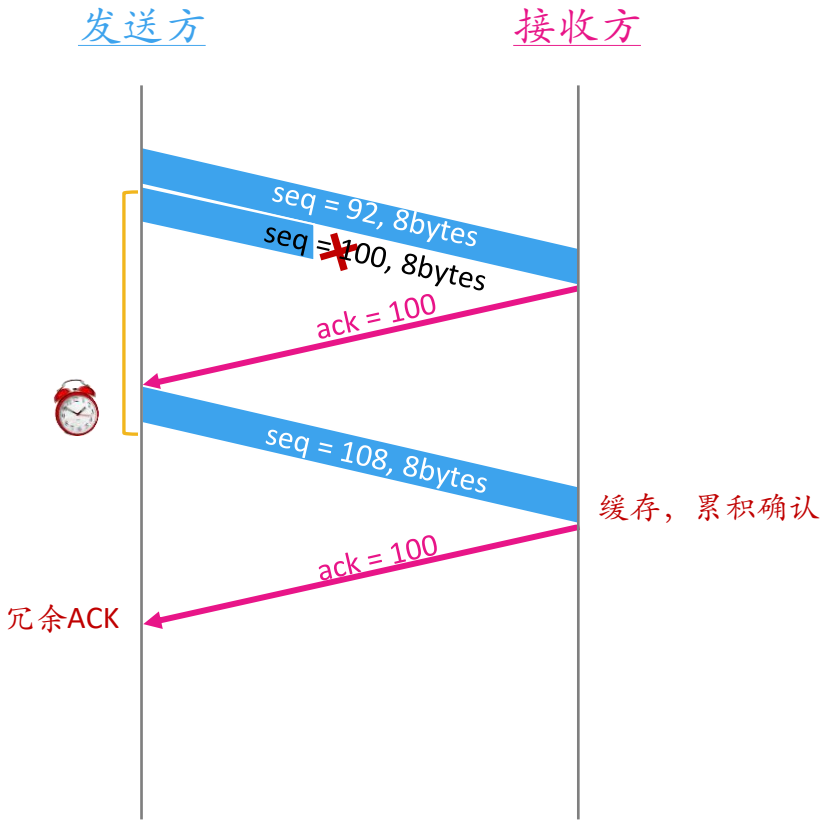
TCP可靠传输

- 数据丢失
  - 假设还有数据发送



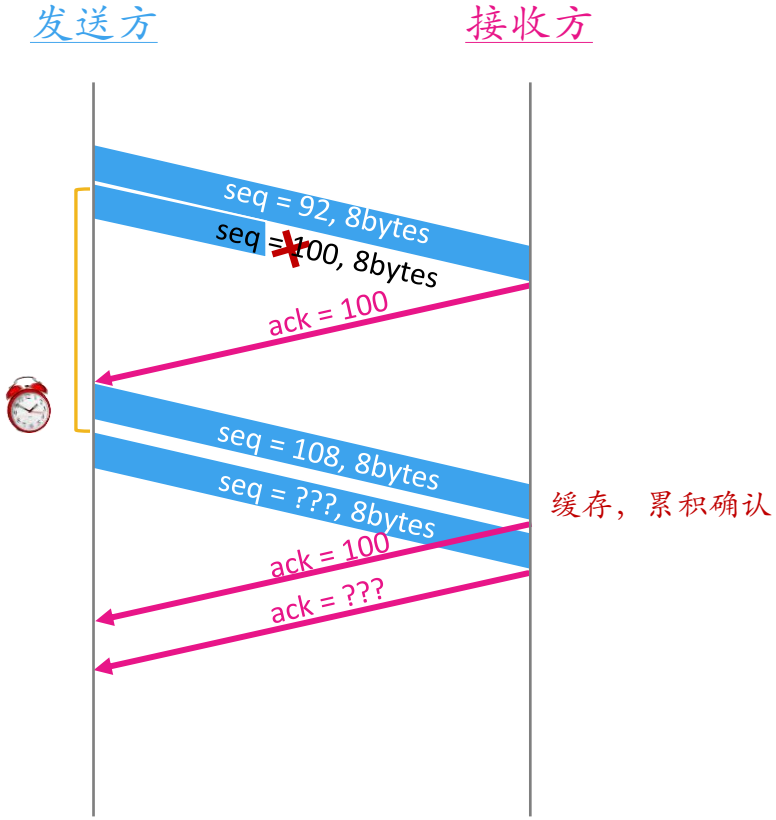
TCP可靠传输

- 数据丢失
  - 假设还有数据发送



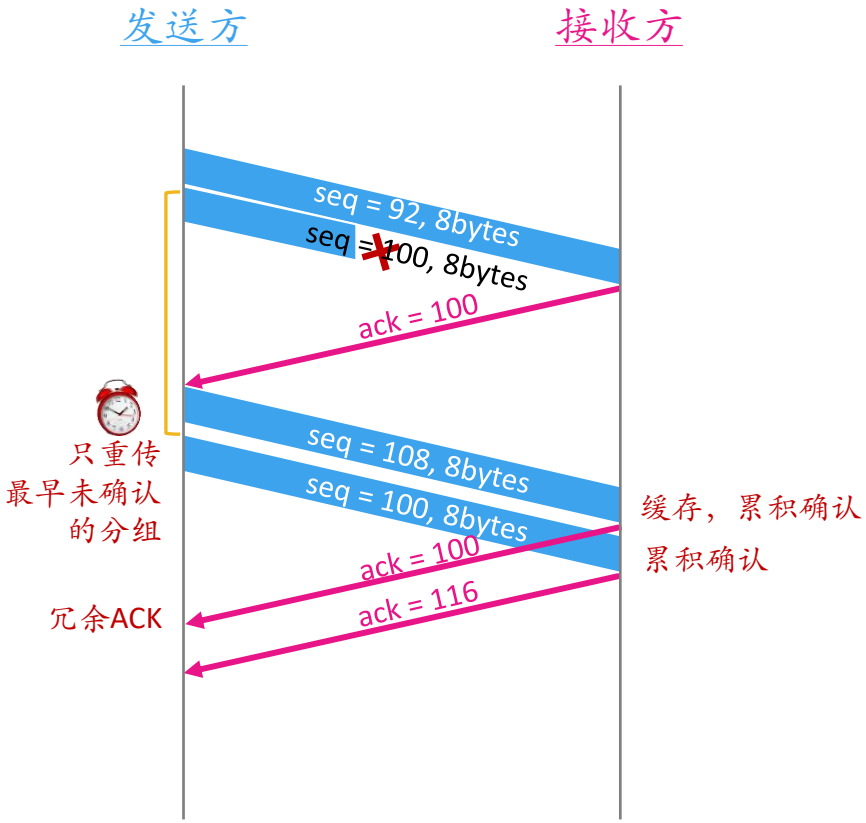
TCP可靠传输

- 数据丢失
  - 假设还有数据发送



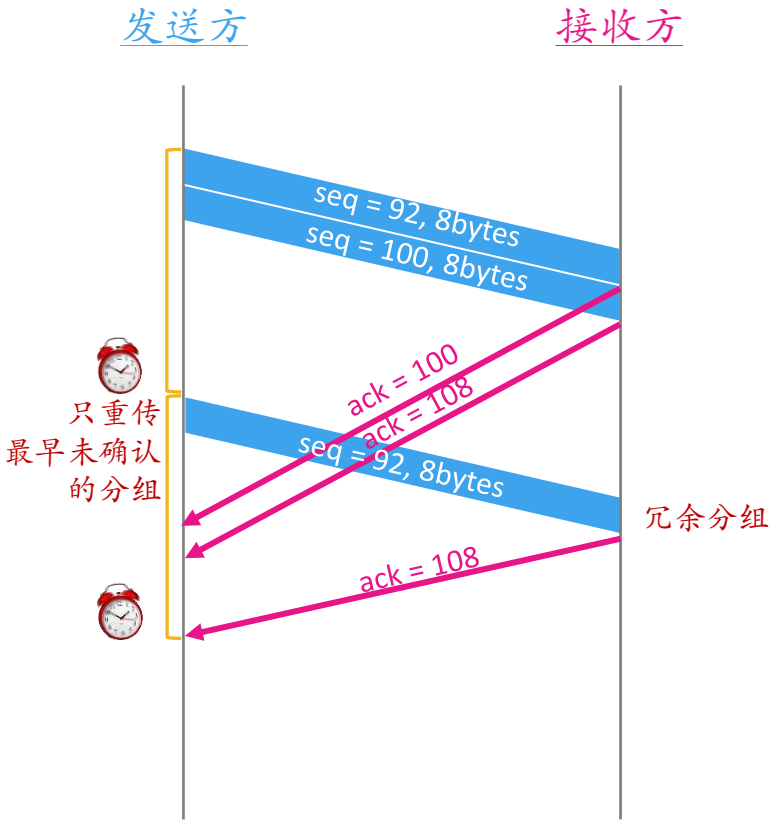
TCP可靠传输

- 数据丢失
  - 假设还有数据发送

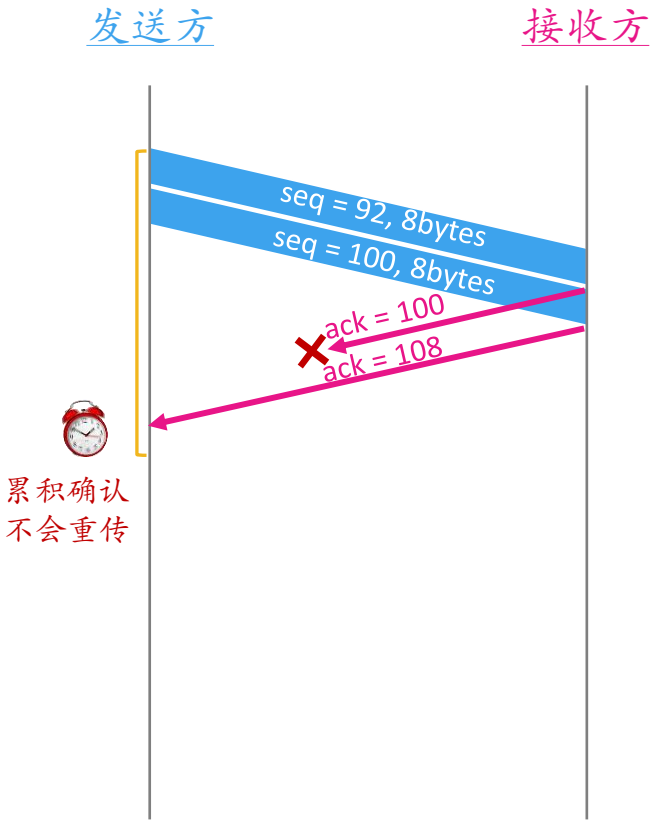


TCP可靠传输

- 过早重传

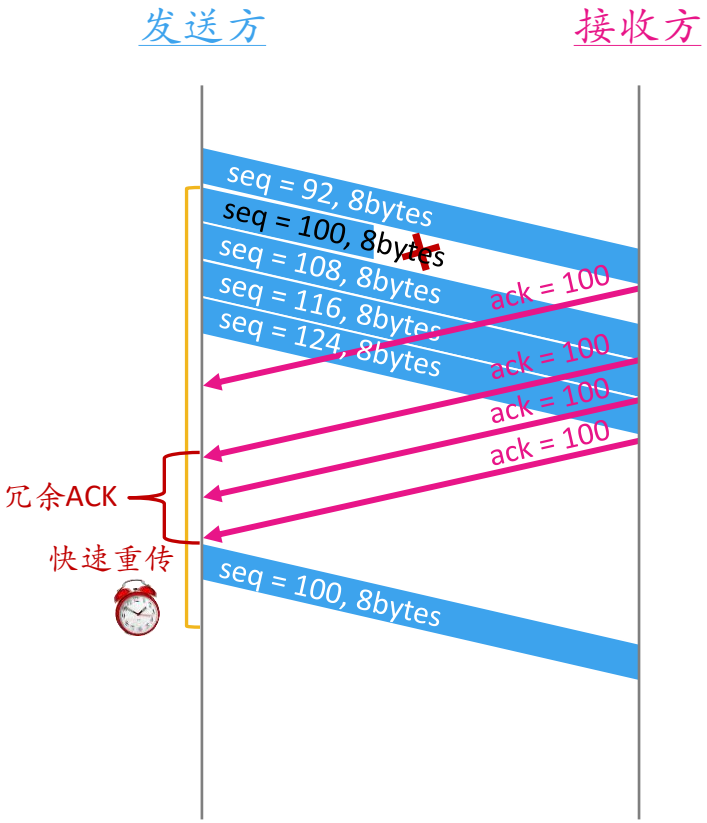


- 累积确认



TCP可靠传输

- 快速重传
  - 如果出现三个冗余ACK，即使定时器还没过期，提早重传

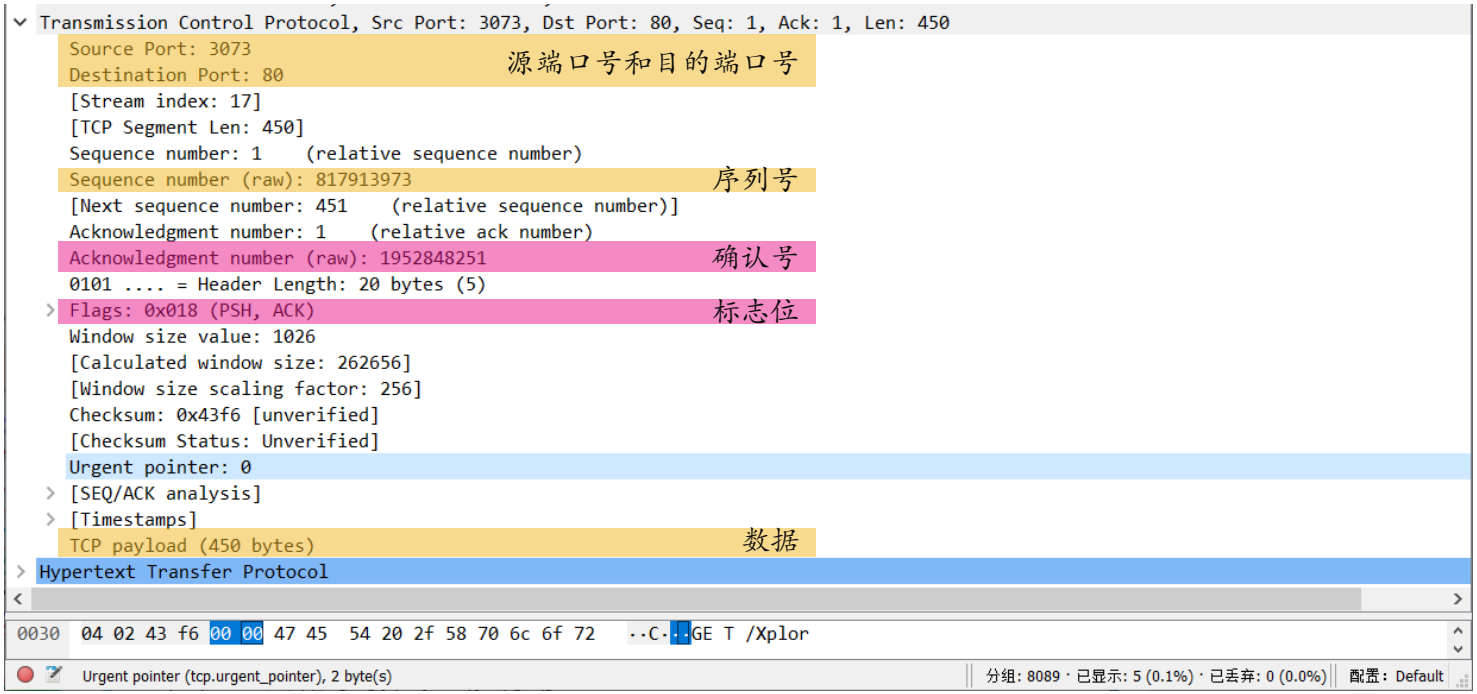


TCP可靠传输

- 利用了校验和、确认、序号、定时器、流水线等技术
- 序号：基于字节流，报文段数据首字节的字节流编号
- 确认号：已确认接收数据的最大序号加一，含义是：期望从对方收到的下一个字节的序号
- 确认机制：累积确认（类似回退N步）
- 重传：只重传丢失或损坏的分组（类似选择重传）
- 定时器：针对窗口中最早未确认的分组
- 回退N步和选择重传的混合体

TCP可靠传输

- TCP提供全双工通信
  - TCP连接的双方可以同时通信，当应用层数据由A流向B时，应用层数据也可以由B流向A
  - 因为两个方向上都有数据传输，所以一个方向的ACK顺带在（piggyback）另一个方向的数据分组中



第三章知识点汇总

- 理解TCP可靠传输的原理
- 理解各种可靠传输技术在TCP中的应用
- 理解TCP快速重传的原理

The best does not come alone.
It comes with the company of the all.

最好的东西不是独来的，
它伴了所有的东西同来。

——*Tagore*