

第三章 数据链路层

滑窗协议



如何提高信道利用率？

- 全双工
- 捎带确认
- 批量发送数据：滑窗技术（协议4~协议6）
 - 协议4： $n=1$ ——引出滑动窗口的基本概念
 - 协议5： 回退 n 帧（Go Back n ）
 - 协议6： 选择重传（Select Repeat）



如何提高信道利用率？

□ 两个窗口

➤ 发送窗口

- 对应着 已经发送但还未被确认的帧 的序号

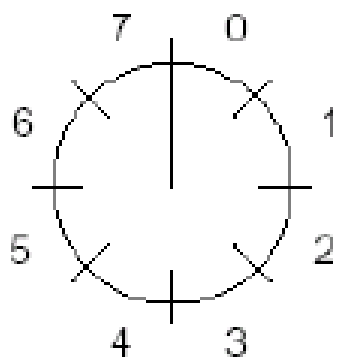
➤ 接收窗口

- 对应着 期望接收 的帧的序号

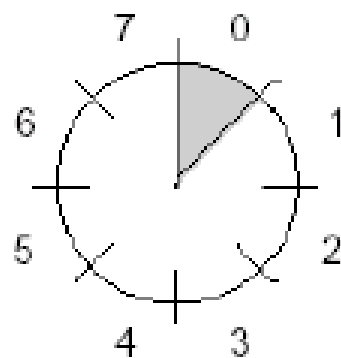


窗口大小为1，有3位序列号的滑动窗口

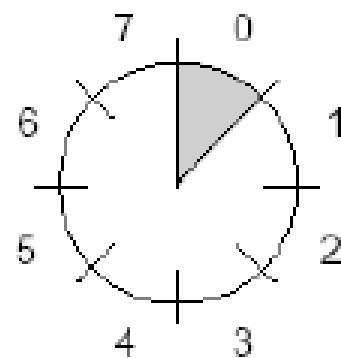
Sender



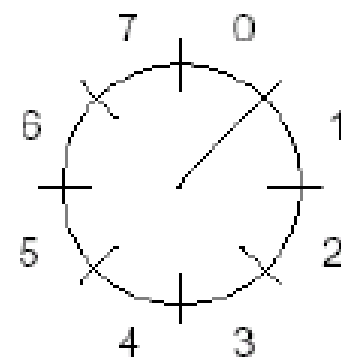
初始时



发送了第一帧

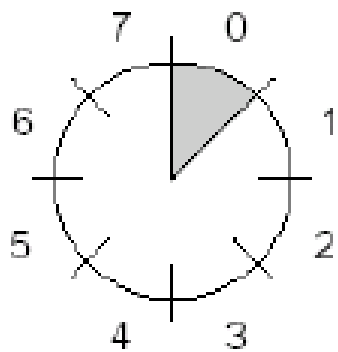


接收了第一帧

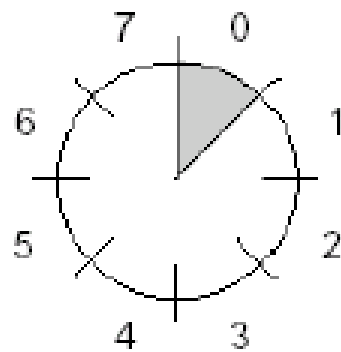


收到第一个确认帧

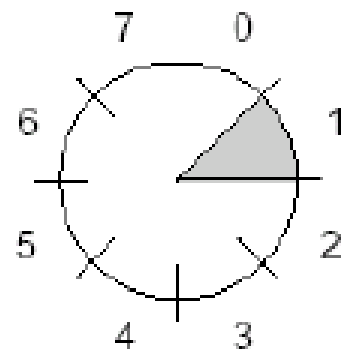
Receiver



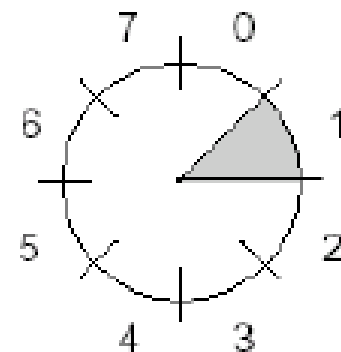
(a)



(b)



(c)



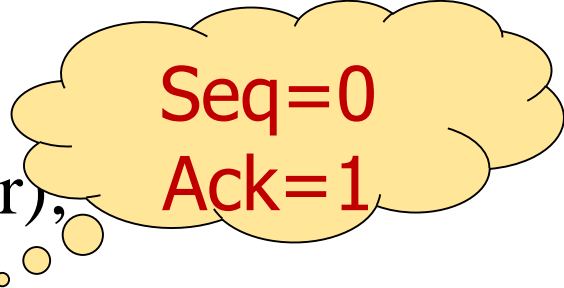
(d)



窗口滑动的条件

- 接收方收到帧后，首先核对是否为预期帧号(`frame_expected`)，如果是的，则接收并`frame_expected+1`，即**移动接收窗口**。
- 发送端收到应答帧，核对响应帧号`next_frame_to_send`，核对无误后，从网络层取新的帧，并执行`next_frame_to_send+1`，即**移动发送窗口**。如核对帧号不正确，则不移动窗口

```
void protocol4(void)
{
    seq_nr next_frame_to_send;
    seq_nr frame_expected;
    frame r, s;
    packet buffer;
    event_type event;
    next_frame_to_send = 0;
    frame_expected = 0;
    from_network_layer(&buffer),
    s.info = buffer;
    s.seq = next_frame_to_send;
    s.ack = 1 - frame_expected;
    to_physical_layer(&s);
    start_timer(s.seq);
}
```



Seq=0
Ack=1

```
while (true) {  
    wait_for_event(&event);  
    if (event == frame_arrival) {  
        from_physical_layer(&r);  
        if (r.seq == frame_expected) {  
            to_network_layer(&r.info);  
            inc(frame_expected);  
        }  
        if (r.ack == next_frame_to_send) {  
            stop_timer(r.ack);  
            from_network_layer(&buffer);  
            inc(next_frame_to_send);  
        }  
    }  
    s.info = buffer;  
    s.seq = next_frame_to_send;  
    s.ack = 1 - frame_expected;  
    to_physical_layer(&s);  
    start_timer(s.seq);  
}  
}
```

移动
接收窗口


收到对方的
捎带确认

移动
发送窗口

捎带确认


w=1滑动窗口概念

A的发送窗口



	A7	窗口外的 发送帧
	A6	
1	A5	待确认的 发送帧
0	A4	
1	A3	被确认的 发送帧
0	A2	
1	A1	
0	A0	
Seq	Data	

B的接收窗口



Seq	Data	提交网络层的 正确接收帧
0	A0	
1	A1	
0	A2	
1	A3	待提交的 接收帧
0	A4	
1	A5	窗口外的 接收帧
	A6	



滑动窗口的基本概念

- 每个待发送帧被赋予一个序列号seq
 - seq的取值范围是 $0 \sim 2^n - 1$ (n位字段)
- 建立缓冲区
 - 发送窗口：缓存已发送、待确认的帧
 - 顺序接收来自网络层的分组，成帧，赋予序列号
 - 最多保存W个已经发送、等待确认的帧
 - 窗口达到最大值W时强制关闭网络层
 - 接收窗口：缓存期待接收的帧（序号）
 - 对进入窗口的帧顺序提交网络层，产生确认
 - 落在窗口外的帧被丢弃



协议4的滑动窗口基本工作原理

□ 窗口设置

- 滑动窗口最大值: $\text{MAX_SEQ} = 1$
- 通信双方初始值: $\text{seq} = 0, \text{ack} = 1$ (期待接收 $\text{seq} = 0$)

□ 窗口滑动机制

- A首先发送数据帧 ($\text{seq} = 0, \text{ack} = 1, A0$)
- B收到A0, 发送捎带确认帧 ($\text{seq} = 0, \text{ack} = 0, B0$)
- A收到对A0的确认, 滑动窗口, 发送帧 ($\text{seq} = 1, \text{ack} = 0, A1$)

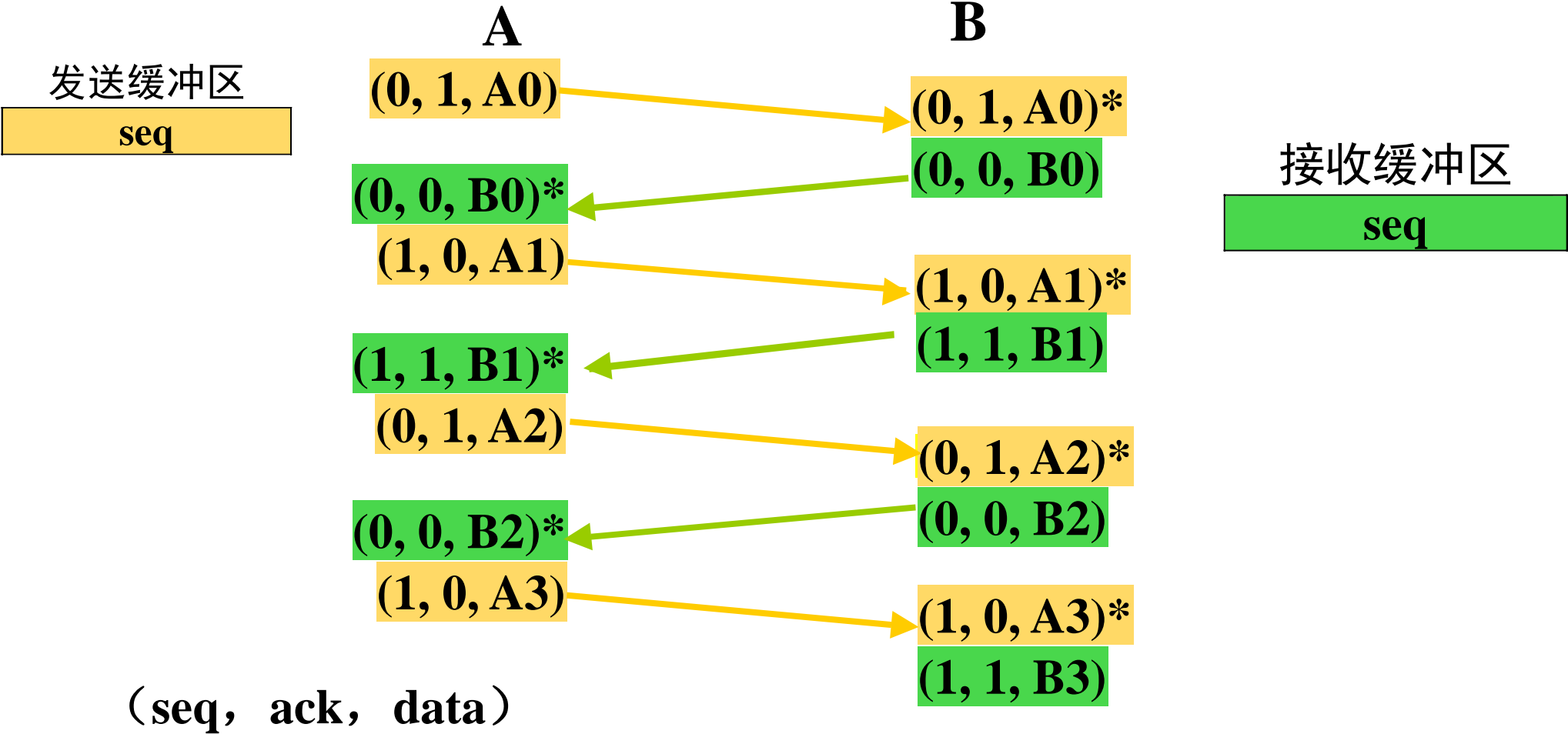


协议4的滑动窗口基本工作原理

□ 特点

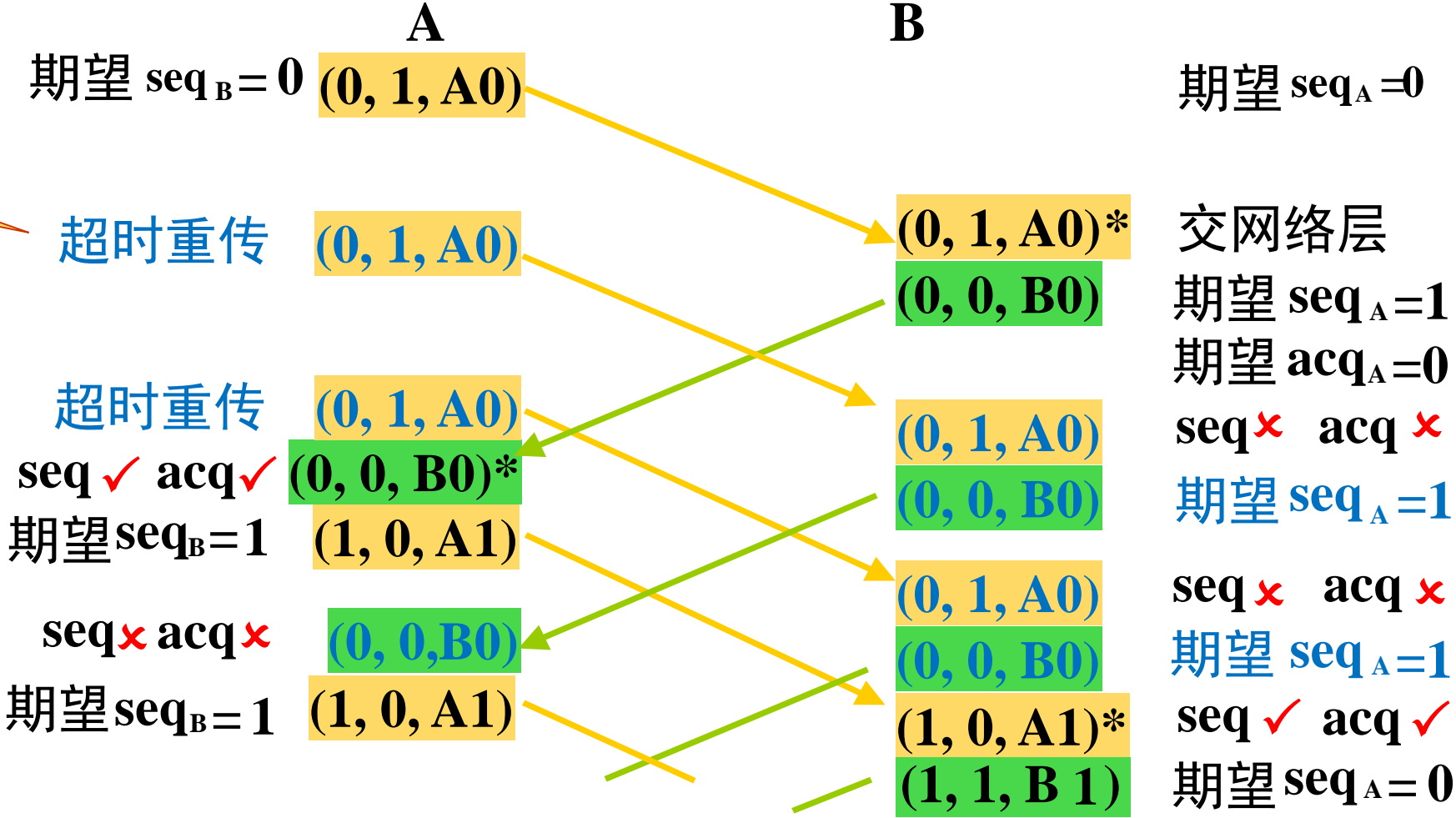
- 序列号seq和确认值ack“0”“1”交替
- 滑动窗口长度 $W=1$ ，收到确认才移动窗口
- 保证按顺序将接收到的正确帧只一次上交网络层

正常情况下发送窗口滑动机制

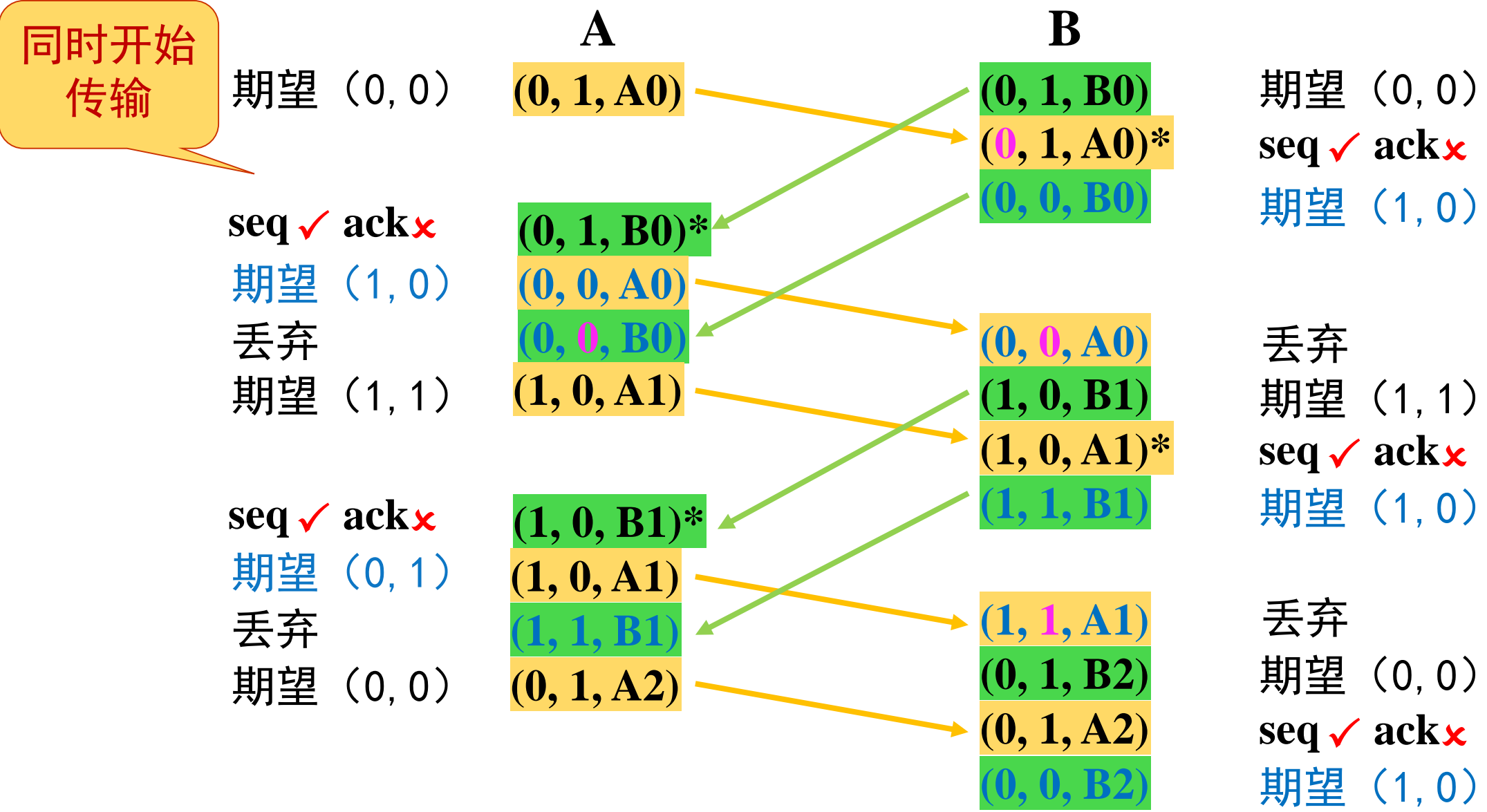


异常情况一：对重复帧的差错控制

定时器
设置短了



异常情况二：同步开始发送过程的差错控制





协议4的信道利用率怎样呢？

- 在协议4中假设：以下时间是可以忽略的：
 - 接收方处理到达帧的时间
- 事实上，在低速信道上，来回时间（RTT: the round-trip time）可能非常大，发送方在这段时间处于blocked状态



协议4的信道利用率

□ 如果：

➤ 信道传输速率是： b bps

➤ 每帧的大小是： k bits

➤ 来回时间是： R sec

则信道的利用率是：

$$\text{Line Utilization Rate} = k/(k + bR)$$

例

□ 已知：

- 信道容量 $b = 50 \text{ kbps}$
- 传输延迟 $R = 500 \text{ ms}$ （双程）
- 数据帧的长度 $k = 1000 \text{ bit}$
- 设接收方收到数据帧后马上回送确认短帧，没有延时

□ 求：信道利用率

解

$$\text{Or: } k/k+bR=1000/1000+50\text{kbps}*500\text{ms}=3.85\%$$

- 在源端发送数据帧过程需要的时间

$$T_f = k/b = 20 \text{ ms}$$

- 从发送完毕到确认帧返回需要的时间（双程延迟）

$$R = 500 \text{ ms}$$

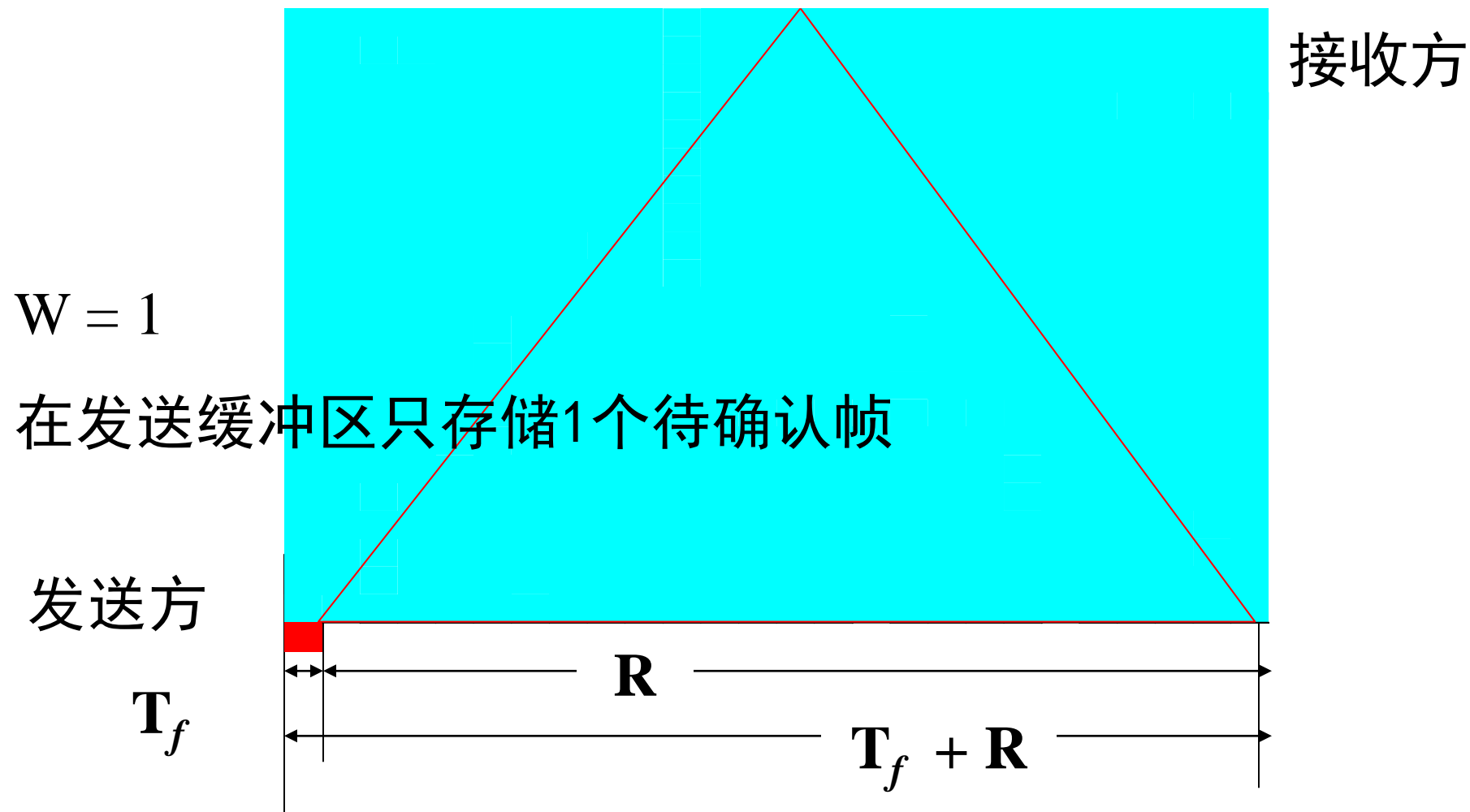
- 从开始发送到确认返回总共需要的时间

$$(T_f + R) = 20 + 500 = 520 \text{ ms}$$

- 线路的利用率

$$T_f / (T_f + R) = 20 / 520 = 3.85 \%$$

信道利用率不足4%





提高信道利用率的方法

- 增加滑动窗口最大长度W

$$\text{信道利用率} = W * T_f / (T_f + R)$$

$$= W * k / (k + bR)$$

- 理想情况下，使例题信道利用率达到100%，则滑动窗口最大长度为：

$$W = (T_f + R) / T_f = 520 / 20 = 26$$



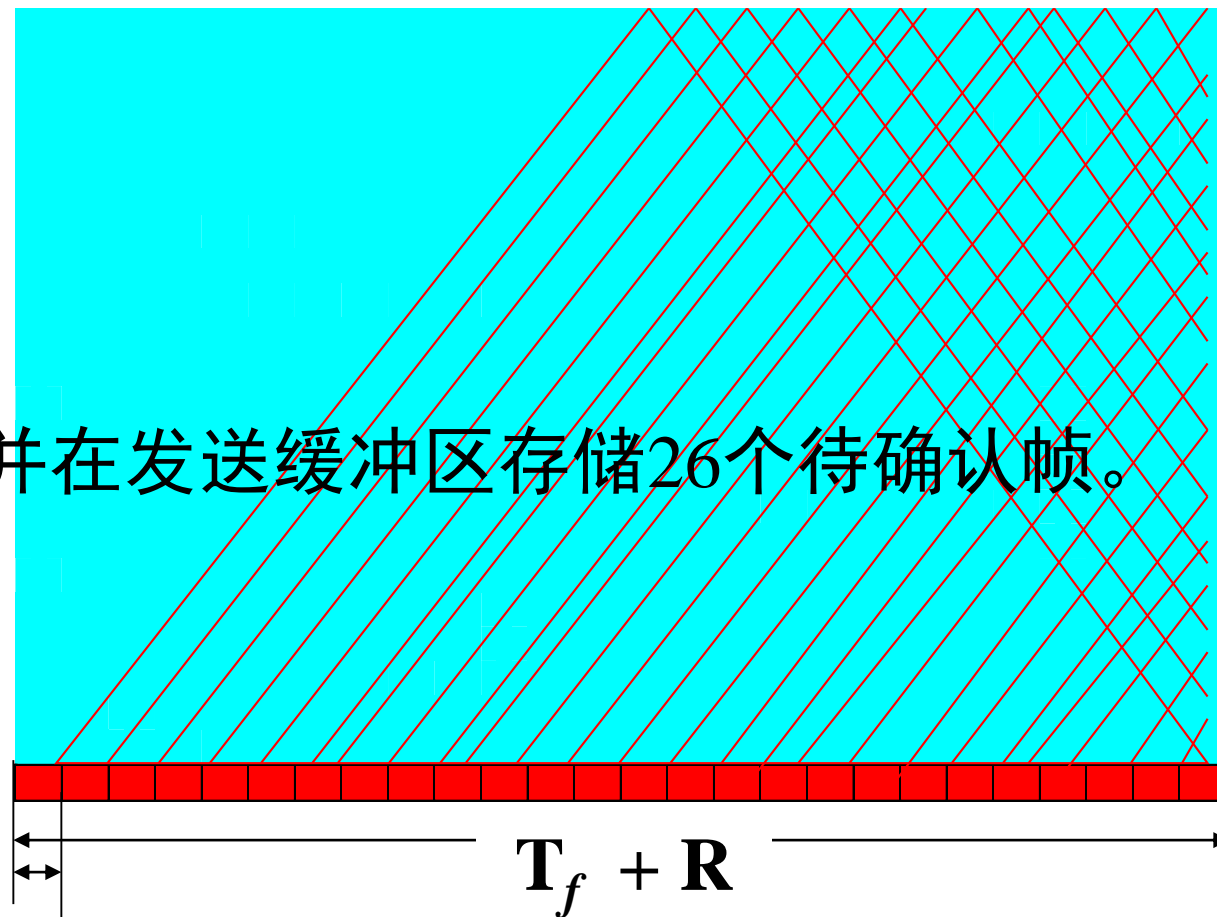
使信道利用率达到100%

$$W = 26$$

连续发送、并在发送缓冲区存储26个待确认帧。

发送方

T_f



$T_f + R$

管道化 (pipelining)

接收方



怎样找到一个合适的w值？

- 信道上的容量：一帧从发送方传输到接收期间可容纳的帧数量
- 带宽-延迟积：BD
- 窗口值： $w=2*BD+1$
- 上述的例子：
 - $BD=50\text{kbps} * 0.250 = 12.5\text{kb}$
 - $W=2*12.5\text{kb}+1=26\text{kb}=26\text{帧}$

实际上：

$$w \leq 2*BD+1$$

例（2014考研题）

主机甲和主机乙之间使用后退N帧协议（GBN）传输数据，甲的发送窗口尺寸为1000，数据帧长为1000字节，信道为100Mbps，乙每收到一个数据帧立即利用一个短帧（忽略其传输延迟）进行确认。若甲乙之间的单向传播延迟是50ms，则甲可以达到的最大平均传输速率约为：

- A. 10Mbps B. 20Mbps
- C. 80Mbps D. 100Mbps

解

设可达到的最大传输率为 x ，于是

➤ $1000f * 1000Bpf * 8 = xbps * 2 * 50ms / 1000ms$

➤ $80000000 = 8000 + 2 * 0.05x$

➤ $x = 80Mbps$

➤ 或者： $w = 1251$ ，现在只有 1000，于是

➤ $X = 100M * (1000 / 1251) = 80M$

用管道化技术发送帧面临的新问题

□ 出错情况

- 连续发送 W 个数据帧，其中有一帧出错，但其后续帧被成功发送

□ 接收方的接收策略选择

- 丢弃错帧及后续帧，其后续帧因不是期望接收帧也被丢弃
- 丢弃错帧，缓存后续正确接收帧

用管道化技术发送帧面临的新问题

□ 对应的发送方的重传策略选择

- **缓存**在发送窗口中的出错帧以及其后续帧全部重发——
协议5
- **只重发出错帧**——协议6



小结

- 滑窗技术可以批量收发数据，提高了信道利用率。
- 发送窗口对应着已经发送但还未被确认的帧。
 - 滑动条件：收到了帧的确认。
- 接收窗口对应着：期待接收的帧。
 - 滑动条件：收到了期待接收的帧。
- 窗口数的确认跟带宽延迟积正相关。

思考题

- 为什么提出滑窗技术？
- 什么是发送窗口？
- 发送窗口什么时候滑动？
- 什么是接收窗口？
- 接收窗口什么时候滑动？
- 什么是带宽延迟积？
- 窗口值怎么确定？跟哪些因素有关？

谢谢观看

致谢

本课程课件中的部分素材来自于：（1）清华大学出版社出版的翻译教材《计算机网络》（原著作者：Andrew S. Tanenbaum, David J. Wetherall）；（2）思科网络技术学院教程；（3）网络上搜到的其他资料。在此，对清华大学出版社、思科网络技术学院、人民邮电出版社、以及其它提供本课程引用资料的个人表示衷心的感谢！

对于本课程引用的素材，仅用于课程学习，如有任何问题，请与我们联系！