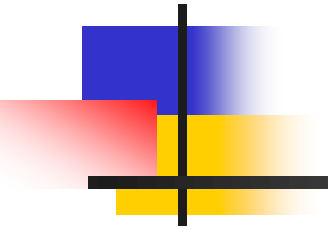


第3部分 数据链路层
——ARQ

刘志敏
liuzm@pku.edu.cn

数据链路层的功能

- 相邻结点间的数据传输存在哪些问题？

(1) 链路管理：建立链路、拆除链路

(2) **同步**：组帧、解帧

(3) 流量控制

(4) **差错控制**

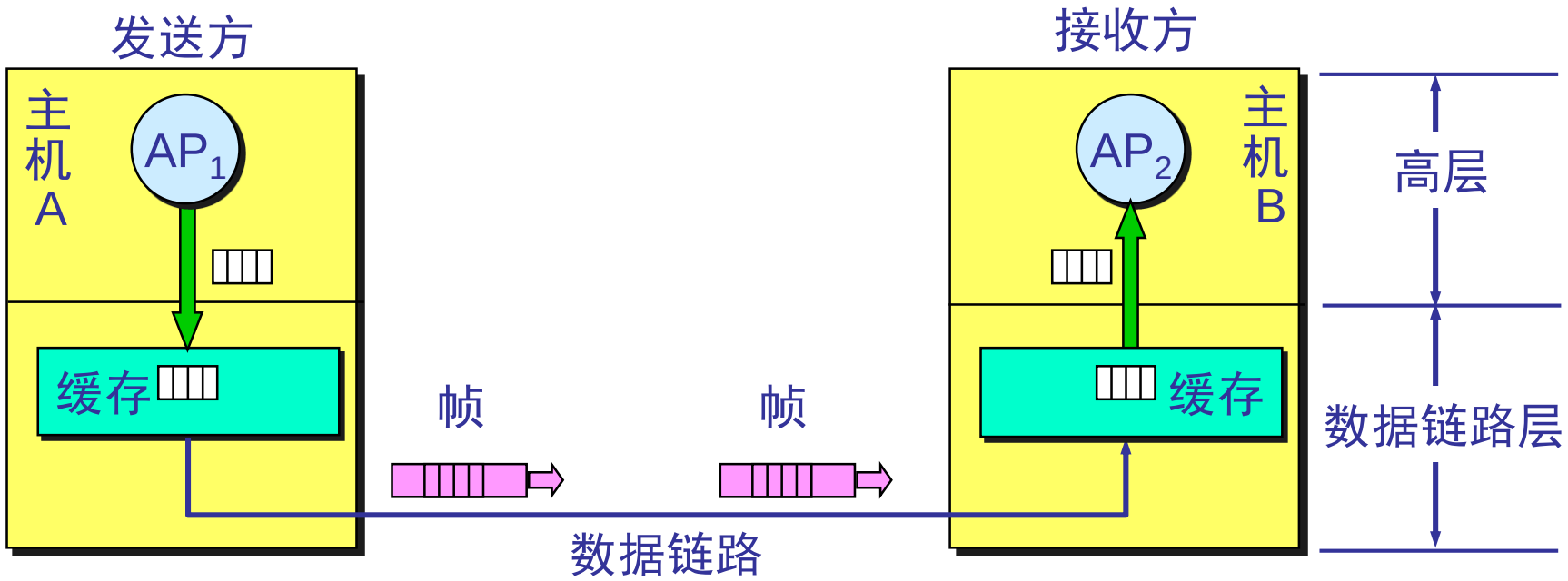
(5) 区分数据信息和控制信息

(6) **透明传输**

(7) 寻址

数据链路层协议

- 链路层实现相邻结点之间以帧为单位的通信
- 帧的检错及纠错：在帧尾加帧校验
- 定义一系列的函数，如 wait_for_event(&event)；event，表示发生的事件
- 发送方：发送帧给另一主机
- 接收方：收到数据后计算帧校验，设置event；若有错，则event=chksum_err；否则event=frame_arrival



数据链路层协议程序

数据类型定义*protocol.h*

```
#define MAX_PKT 1024                                  /* determines packet size in bytes */

typedef enum {false, true} boolean;                   /* boolean type */
typedef unsigned int seq_nr;                          /* sequence or ack numbers */
typedef struct {unsigned char data[MAX_PKT];} packet; /* packet definition */
typedef enum {data, ack, nak} frame_kind;             /* frame_kind definition */


typedef struct {                                      /* frames are transported in this layer */
  frame_kind kind;                                    /* what kind of frame is it? */
  seq_nr seq;                                         /* sequence number */
  seq_nr ack;                                         /* acknowledgement number */
  packet info;                                        /* the network layer packet */
} frame;
```

数据链路层协议程序

函数定义 protocol.h

```
/* Wait for an event to happen; return its type in event. */
void wait_for_event(event_type *event);

/* Fetch a packet from the network layer for transmission on the channel. */
void from_network_layer(packet *p);

/* Deliver information from an inbound frame to the network layer. */
void to_network_layer(packet *p);

/* Go get an inbound frame from the physical layer and copy it to r. */
void from_physical_layer(frame *r);

/* Pass the frame to the physical layer for transmission. */
void to_physical_layer(frame *s);

/* Start the clock running and enable the timeout event. */
void start_timer(seq_nr k);

/* Stop the clock and disable the timeout event. */
void stop_timer(seq_nr k);
```

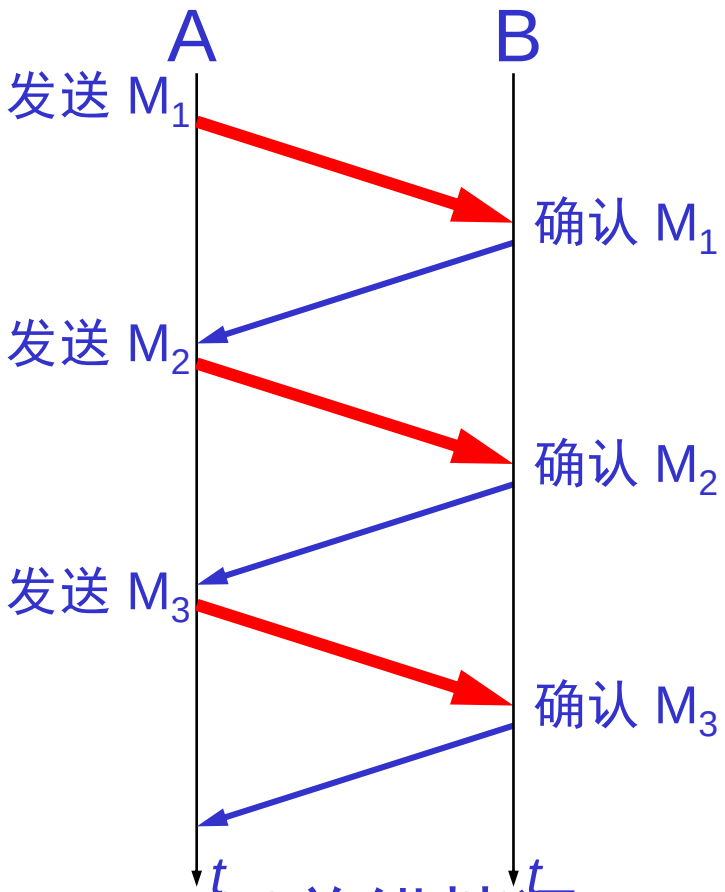
数据链路层协议程序V0.1

```
typedef enum {frame, arrive} event type;
#include "protocol.h"
void sender1(void)
{
   frame s;
   packet buffer;
   while(true){
     from_network_layer(&buffer);
     s.info = buffer;
     to_physical_layer(&s);
   }
}
```

```
#include "protocol.h"
void receiver1(void)
{
  frame r;
  packet buffer;
  event type event;
  while(true){
     wait_for_event(&event);
     physical_layer(&r)
     to_network_layer(&buffer);
   }
}
```

- 这对程序运行之后有何问题？
- 若发端发得快收端来不及接收或未准备好，将丢失数据。
- 发端应根据收端的情况控制其发送速率——流量控制！

停等协议的原理



(a) 无差错情况

收端收到帧后发送确认帧；发端收到确认帧后才发送新的帧

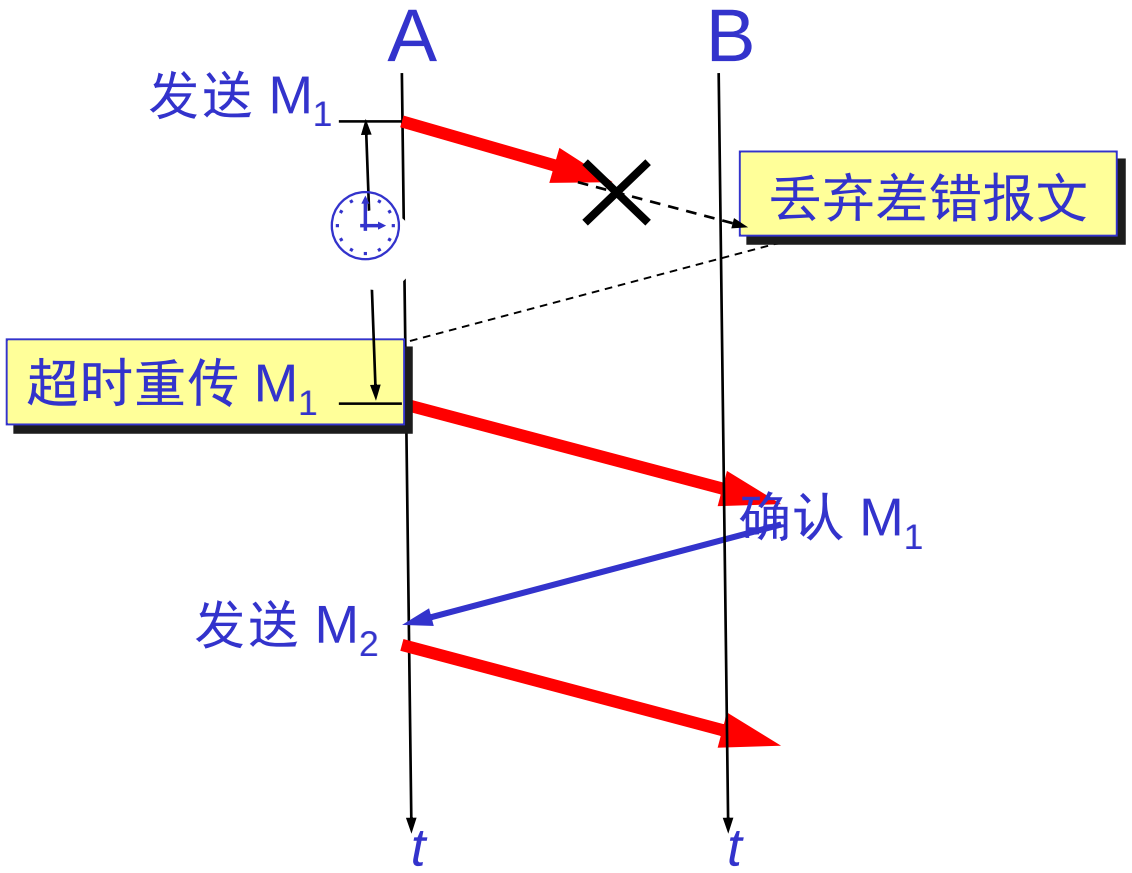
数据链路层协议程序V0.2

```
typedef enum {frame, arrive} event type;
#include "protocol.h"
void sender2(void)
{
  frame s;
  packet buffer;
  while(true){
     from_network_layer(&buffer);
     s.info = buffer;
     to_physical_layer(&s);
     wait_for_event(&event);
   }
}
```

```
#include "protocol.h"
void receiver2(void)
{ frame r,s;
  packet buffer;
  event type event;
  while(true){
     wait_for_event(&event);
     physical_layer(&r)
     to_network_layer(&buffer);
     to_physical_layer(&s);
   }
}
```

- 这一对程序运行之后还有何问题？
- 发送帧可能会错，需要接收端确认；
- 若发端没有收到确认，原因是什么？怎么办?

停等协议的原理



(b) 有差错情况，超时重传

收端用帧校验检测并丢弃错帧，发端增加超时定时器

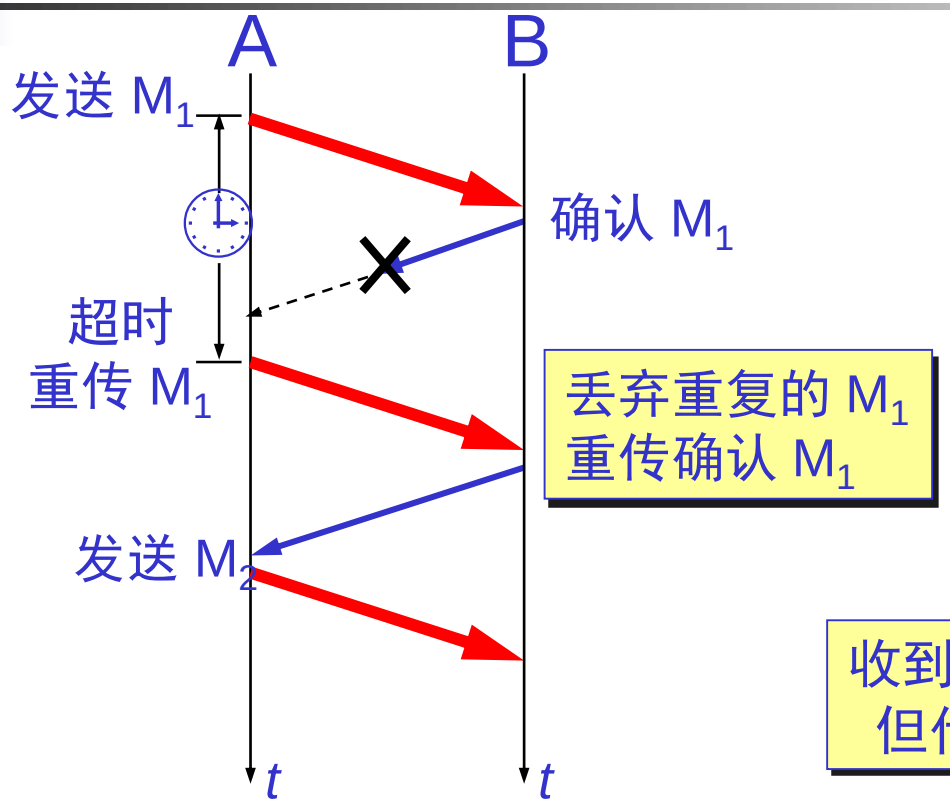
当超时定时器到但还未收到确认，则重传已发送的帧

数据链路层协议程序V0.3

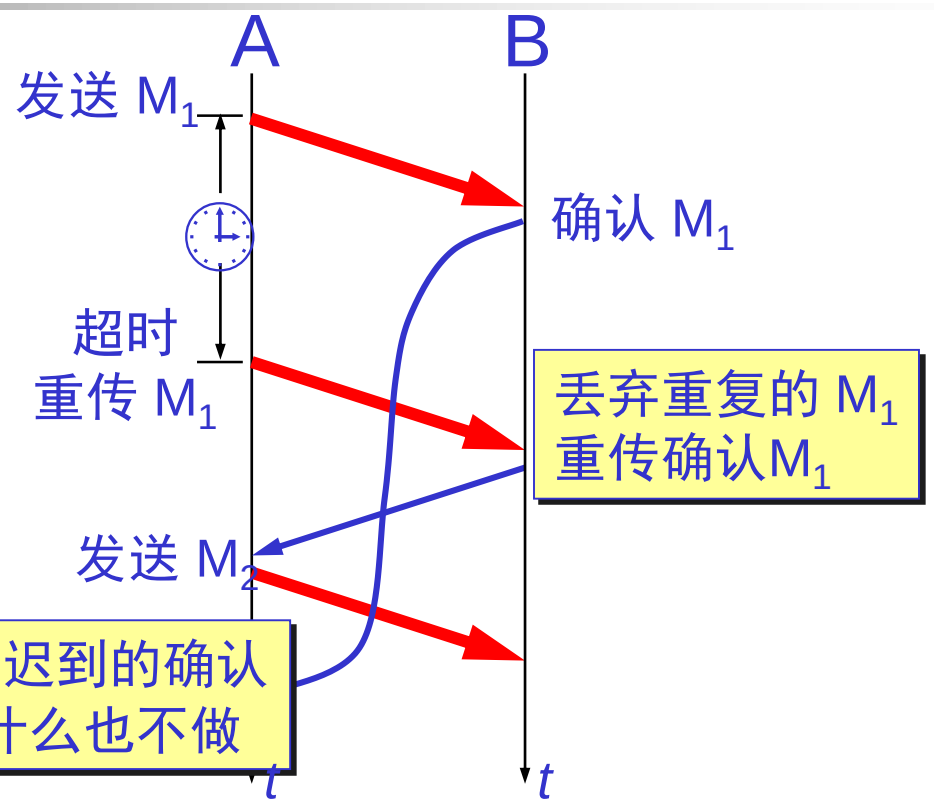
```
typedef enum {frame_arrive} event type;
#define MAX_SEQ1
#include "protocol.h"
void sender3(void)
{ seq_nr next_frame_to_send=0;
  frame s;
  packet buffer;
  from_network_layer(&buffer);
  while(true){
     s.info = buffer;
     s.seq = next_frame_to_send;
     to_physical_layer(&s);
     start_timer(s.seq);
     wait_for_event(&event);
     if(event== arrive)
         from_physical_layer(&s);
     if(s.ack==next_frame_to_send){
        stop_timer(s.ack):
        from_network_layer(&buffer);
        inc(next_frame_to_send);     }
    }
}
```

```
#include "protocol.h"
void receiver3(void)
{ seq_nr frame_expected=0;
  frame r,s;
  packet buffer;
  event type event;
  while(true){
     wait_for_event(&event);
     if(event == frame_arrive){
        from_physical_layer(&r);
        if(r.seq==frame_expected){
            to_network_layer(&r.info);
            inc(frame_expected);        }
        s.ack=1-frame_expected;
        s.info = buffer;
        to_physical_layer(&s);    }
}
```

停等协议的原理



(c) 确认帧丢失的情况



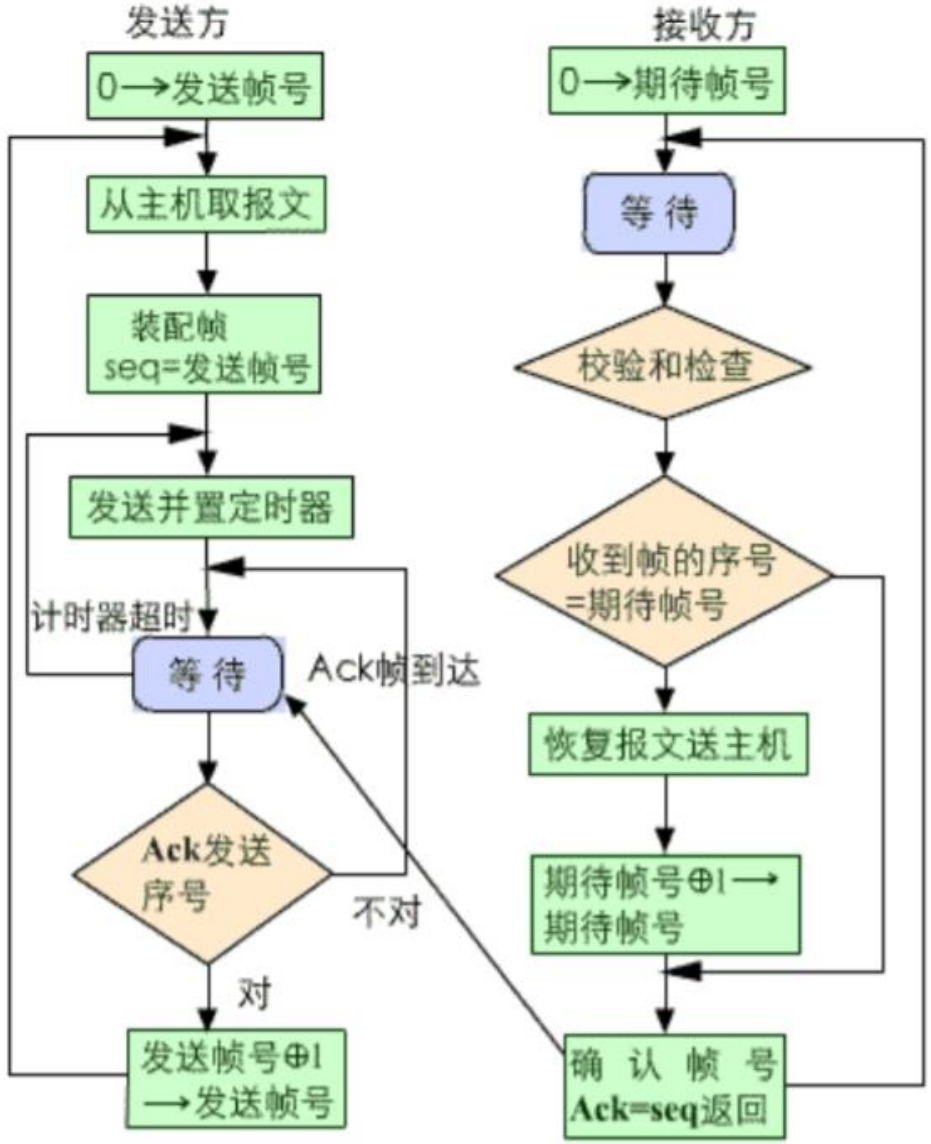
(d) 确认迟到的情况

避免重复帧：数据帧带序号，每发一个新数据帧序号加1；（序号占1比特；对序号异或⊕）

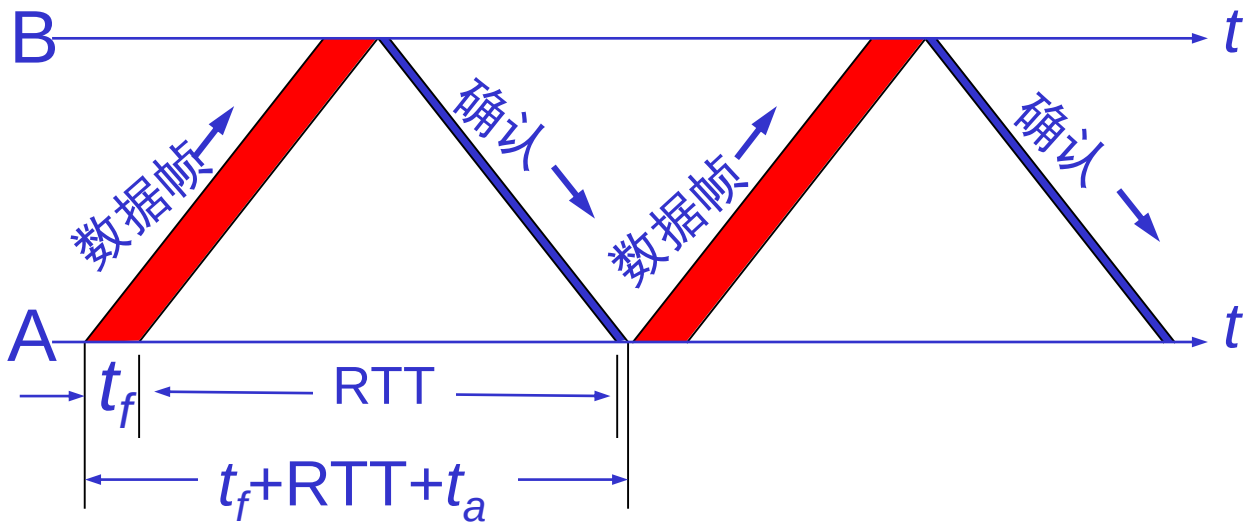
确认帧也要带序号

实现停等协议的要点

- 发送方：
  - 发送数据帧后缓存
  - 启动超时定时器
  - 收到确认帧后清除缓存，发送新数据帧
  - 当定时器到，重传缓存帧
- 接收方：
  - 收到错误的帧，丢弃
  - 收到正确的数据帧时，发送确认；若是新数据帧则保存；若为重复帧则丢弃
- 发送端自动对错帧重传，称为ARQ (Automatic Repeat reQuest)



信道利用率



- 信道利用率$U = \dfrac{t_f}{t_f + RTT + t_a}$
- 停等协议的优点是简单，但在长时延信道上，信道利用率较低

例题：

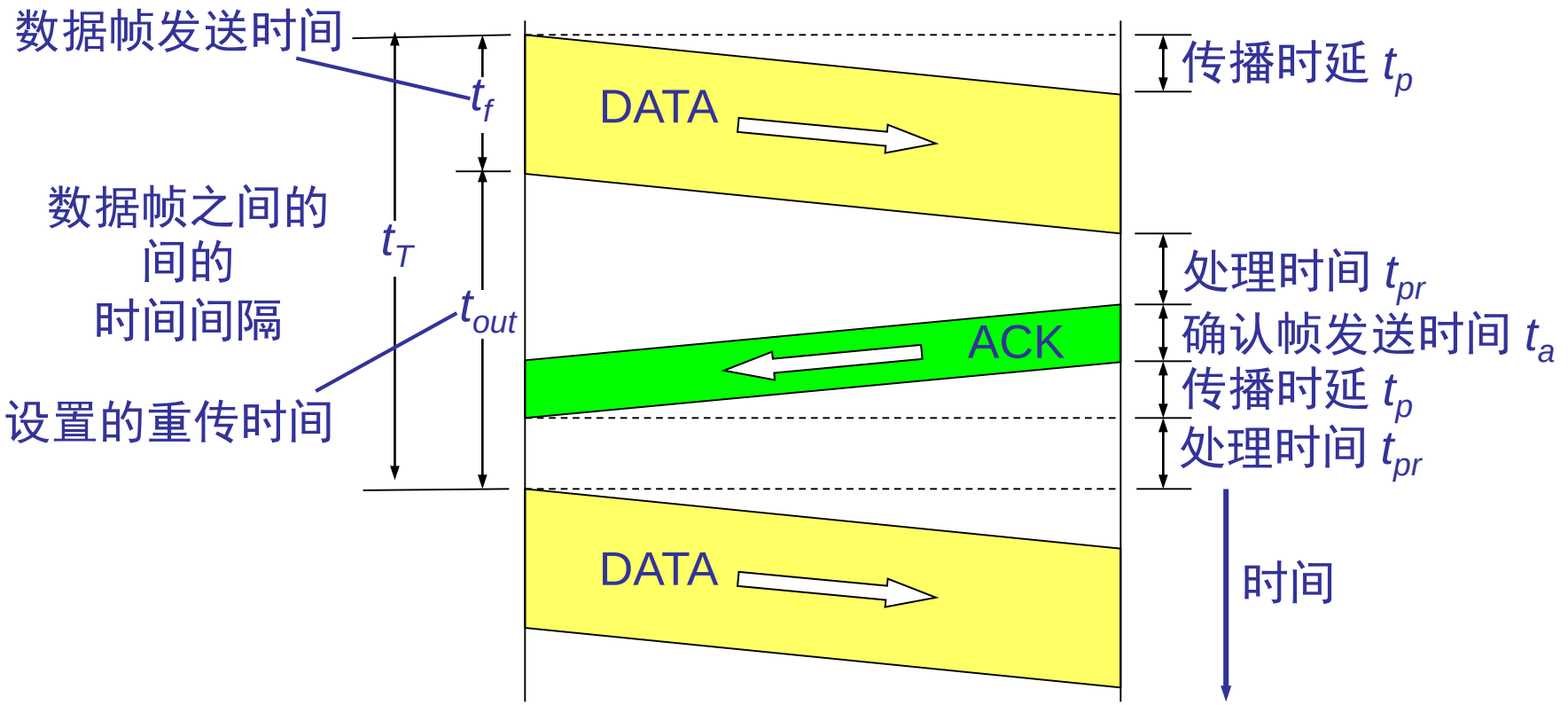
- 在往返时延为40ms链路速率为2Mbps的链路上，发送的帧长为1KB，则链路吞吐量为1000×8/0.04=200Kbps
- 信道利用率为200Kbps/ 2Mbps=10%
- 定义：延迟带宽积=链路速率*延迟时间
- 则延迟带宽积=2Mbps×40ms=80Kb=10KB

- 如何提高链路利用率？
- 如何分析停等协议的性能？

停等协议的定量分析

- 设数据帧长 l_f (bit)，发送速率 R (bit/s)，则数据帧的发送时间t_f

$$t_f = l_f / R \quad \text{(s)} \qquad (1)$$



计算重传时间及信道利用率

- 数据帧发送后若在重传时间后未收到确认，则重传。
- 设重传时间为

$$t_{out} = t_p + t_{pr} + t_a + t_p + t_{pr} \qquad (2)$$

- 因处理时间 t_{pr} 和确认帧的发送时间 t_a 远小于传播时延 t_p，予以忽略，因此，$t_{out} = 2t_p$ (3)
- 定义$\alpha=\frac{t_p}{t_f}$，表示传播时间与帧发送时间之比
- 信道利用率$U=\frac{t_f}{t_f+2t_p}=1/(1+2\alpha)$

例：计算停等协议信道利用率

- 例：在速率为2Mbps传播延迟为20ms的信道上采用停等协议发送数据，为了使信道利用率至少为50%，问(1)帧长为多少?(2)若信道误码率为10^{-6}，求误帧率。

$$u = \frac{1}{1 + 2t_p R / l_f} \geq 50\%$$

$$l_f \geq 2t_p R$$

$$= 2 \times 2000K \times 0.02$$

$$= 80Kb = 10KB$$

- 误帧率=1-(1-p)^L= 80K×10^{-6}=8%，与帧长成正比
- 在发生误帧时数据帧需要重传

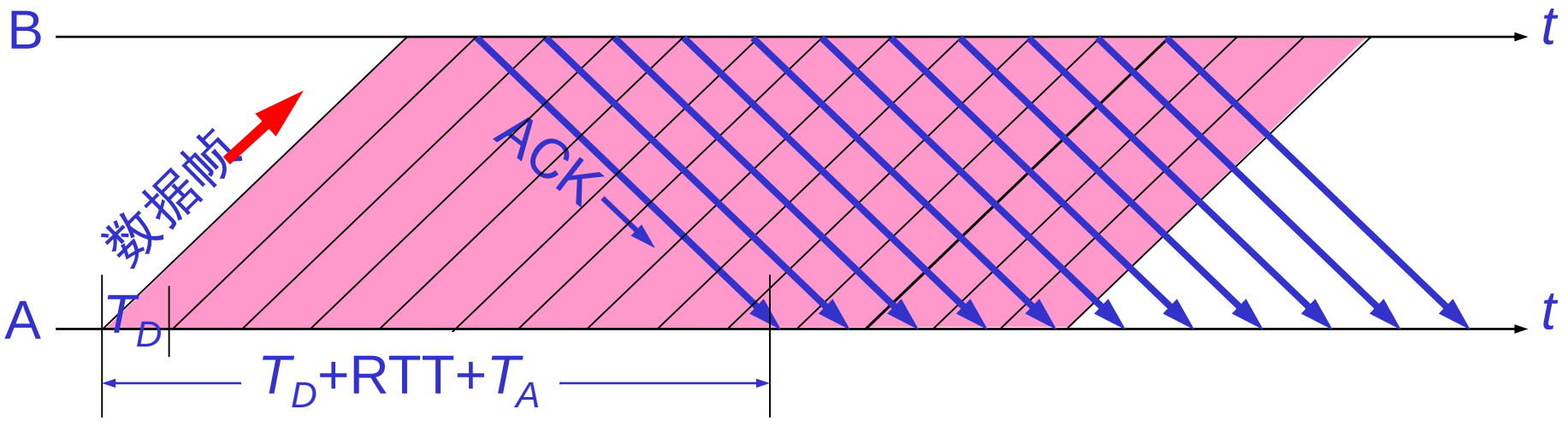
在有误码的信道上采用停等协议，通过增大数据帧帧长的方法不能有效提高链路利用率！

讨论

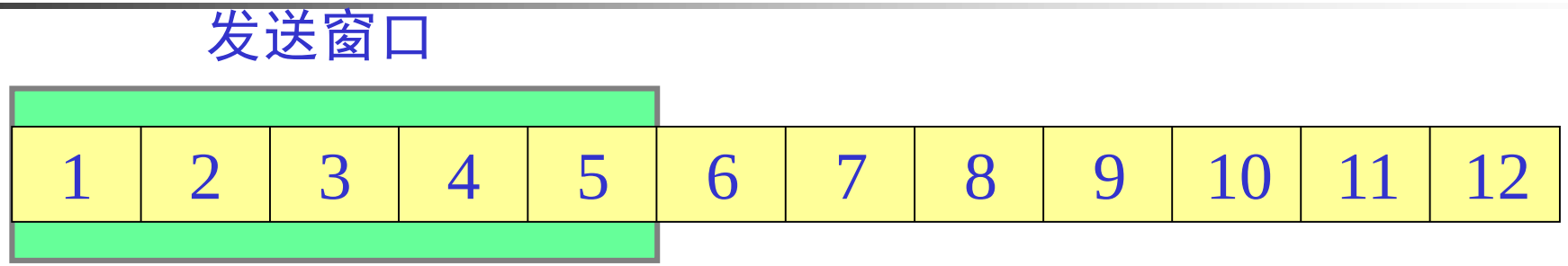
- 停等协议的优缺点：
  - 优点：简单
  - 缺点：信道利用率不高，即信道传输时间未被数据帧充满
- 如何克服缺点？
  - 连续 ARQ
  - 选择重传ARQ
  - 多通道（或多进程）的停等协议（在LTE系统中采用）

连续发送多个帧

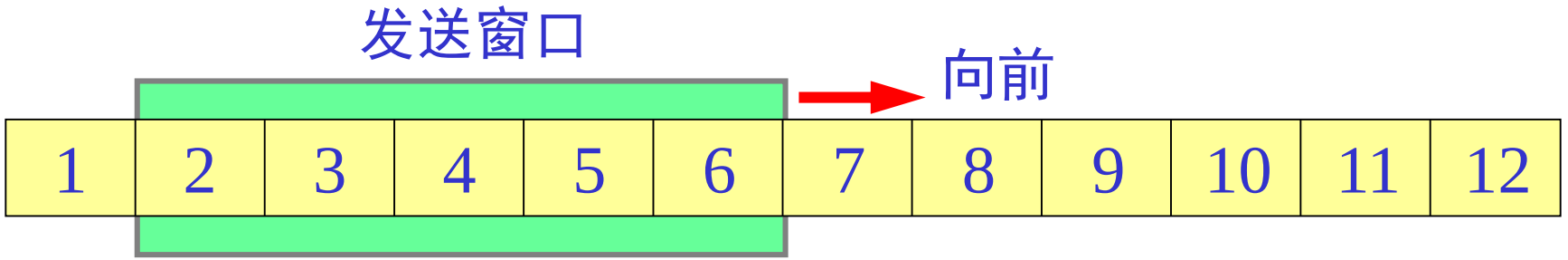
- 发送方连续发送多个帧，无需每发完一个帧就停下来等待确认
- 因信道上有多帧在传送，提高了信道利用率
- 若延迟为t_p，带宽为帧率$\frac{1}{t_f}$（即单位时间内的发送的帧数），则延迟带宽积为$\alpha=\frac{t_p}{t_f}$
- 若可连续发送的帧数为w，则利用率$\leq \frac{w}{1+2\alpha}$



连续 ARQ 协议



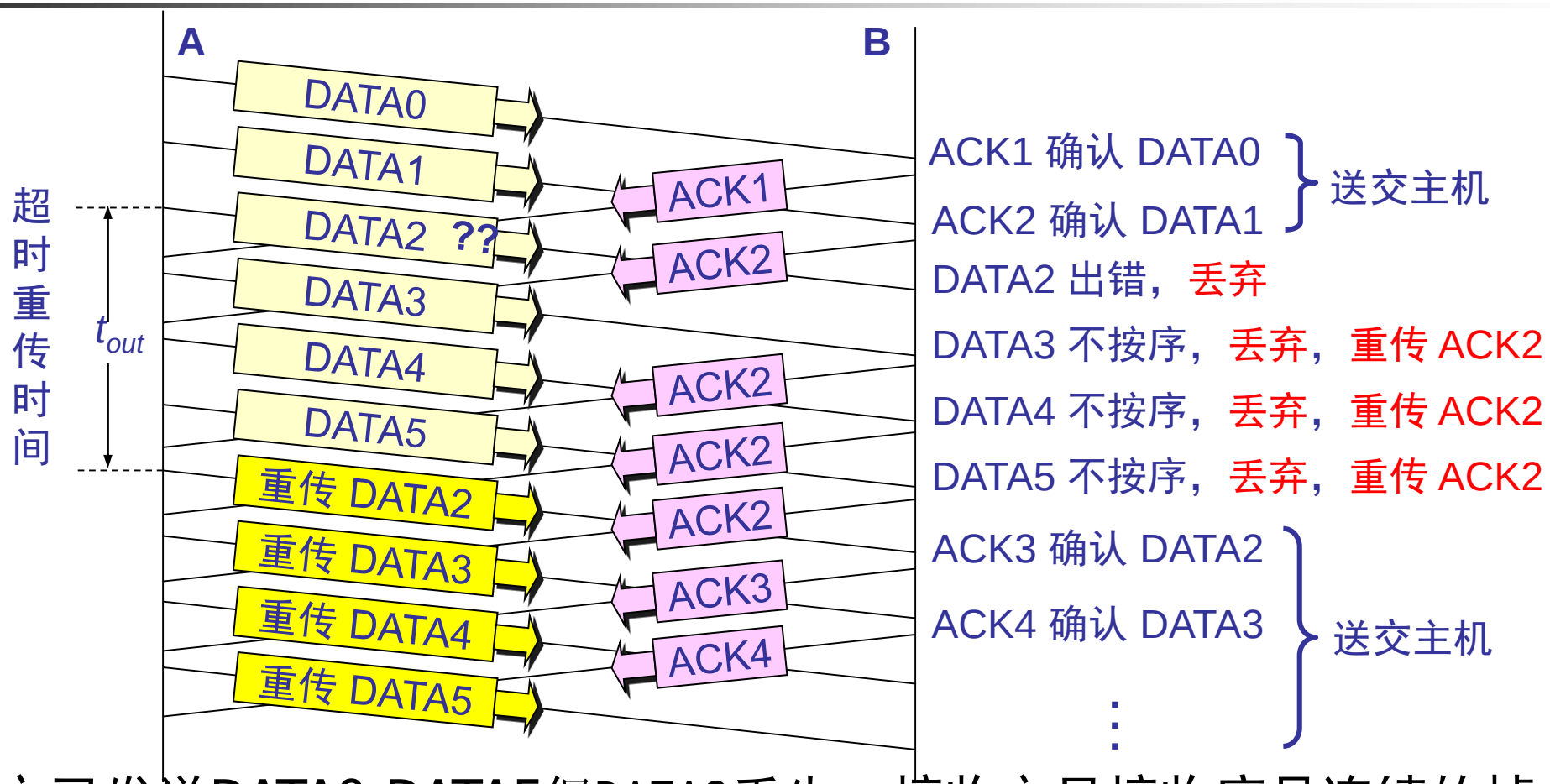
(a) 发送方维持发送窗口（例如 5）



(b) 收到一个确认后发送窗口向前滑动

- 发送方：只允许发送在发送窗口内的W个帧；每发送一个帧启动一个定时器，等待确认
- 接收方：仅接收序号连续的帧，如收到0号帧则响应ACK1，收到1号帧则响应ACK2；ACKi表示期望收到的下一个帧的序号
- 发送方：收到ACKi则取消定时器，调整发送窗口为[i, i+W-1]；若第j个帧定时器到，则调整发送窗口为[j, j+W-1]

连续 ARQ 协议



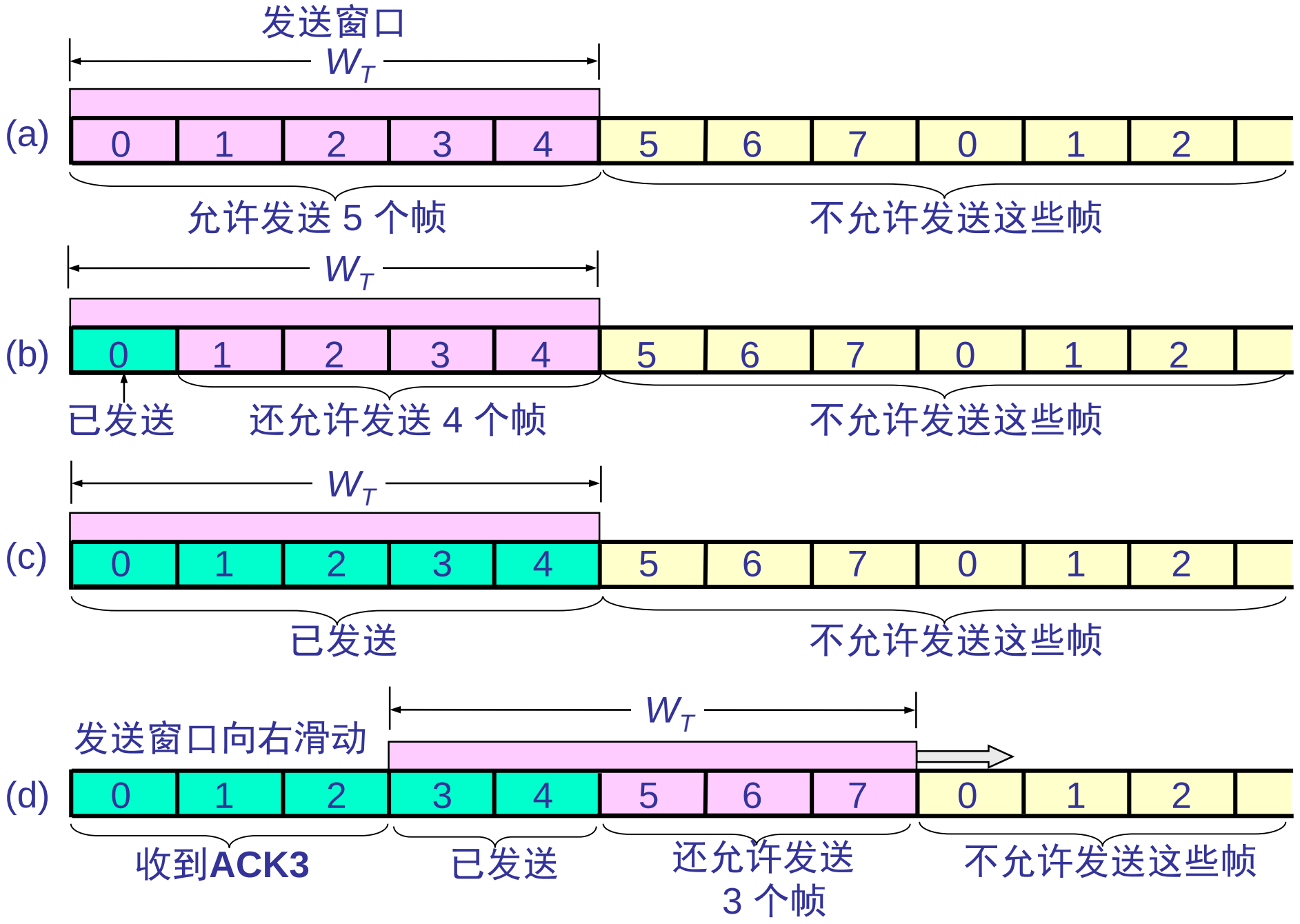
若发送方已发送DATA0-DATA5但DATA2丢失。接收方只接收序号连续的帧，响应ACK1确认DATA0，响应ACK2确认DATA1，之后一直响应ACK2。发送方定时器到时从DATA2帧开始重传发送窗口内的所有帧，称协议为Go-back-N（回退 N），表示回退重传已发送的N个帧。可见，在质量不好的链路上，不适于采用连续ARQ。

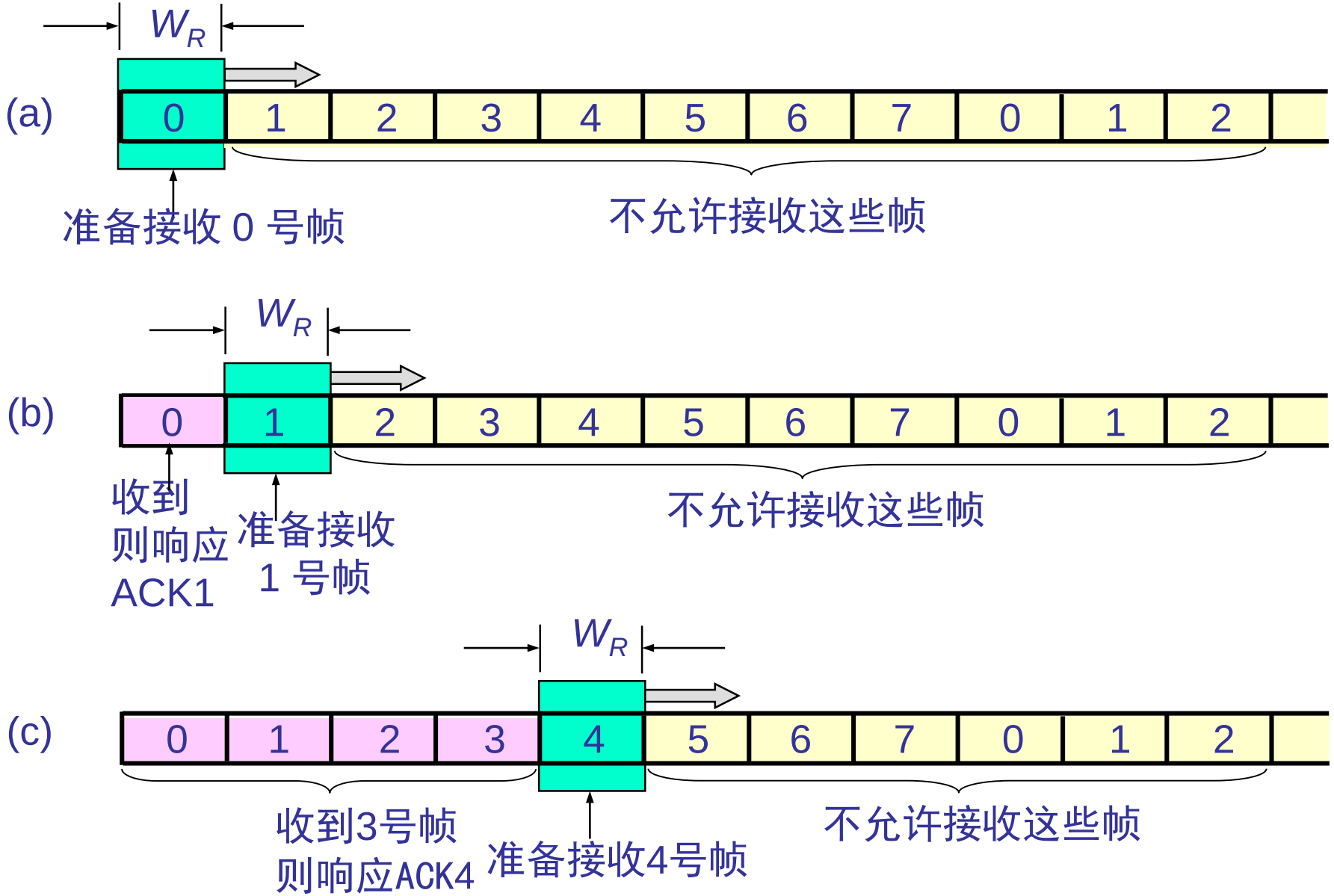
连续 ARQ 的工作原理

- 发送端：设发送窗口为w
  - 发送在发送窗口内[0, w-1]的数据帧，每发送一个数据帧都设置超时计时器
  - 如果收到确认帧i，则取消第（i-1）数据帧的定时器，调整发送窗口为[i, i+w-1]，并发送窗口内的新帧[w, i+w-1]
  - 当第j数据帧超时定时器到，则重传[j, j+w-1]数据帧
- 接收端：按序接收数据帧并响应确认帧。
- 接收端只接收序号连续的帧，因此Go-back-N 又称为连续ARQ 。

滑动窗口的概念

- 发送端和接收端分别设置发送窗口和接收窗口
- 发送窗口：表示发送端最多可发送的数据帧数量
- 接收窗口：接收端只接收帧序号落入接收窗口的数据帧
- 收发两端的窗口不断向前滑动，该协议又称为滑动窗口协议


发送窗口
W_T
(a)
0 1 2 3 4 5 6 7 0 1 2
允许发送 5 个帧
不允许发送这些帧
W_T
(b)
0 1 2 3 4 5 6 7 0 1 2
已发送
还允许发送 4 个帧
不允许发送这些帧
W_T
(c)
0 1 2 3 4 5 6 7 0 1 2
已发送
不允许发送这些帧
W_T
发送窗口向右滑动
(d)
0 1 2 3 4 5 6 7 0 1 2
收到ACK3
已发送
还允许发送
3 个帧
不允许发送这些帧


W_R
(a)
0 1 2 3 4 5 6 7 0 1 2
准备接收 0 号帧
不允许接收这些帧
W_R
(b)
0 1 2 3 4 5 6 7 0 1 2
收到
则响应
ACK1
准备接收
1 号帧
不允许接收这些帧
W_R
(c)
0 1 2 3 4 5 6 7 0 1 2
收到3号帧
则响应ACK4
准备接收4号帧
不允许接收这些帧

有关滑动窗口的大小

- 发送窗口的大小，控制信道上未被确认的数据量，用于流量控制，一般根据接收端处理能力及信道延迟来设置
- 若W_T=1，则为停止等待协议
- 在连续ARQ中，接收窗口$W_R = 1$
- 发送窗口的最大值：当帧序号用 n 个比特编号，若接收窗口为1，则发送窗口为

$$W_T \leq 2^n - 1$$

- 时，连续ARQ才能正确运行。　　（试证明）

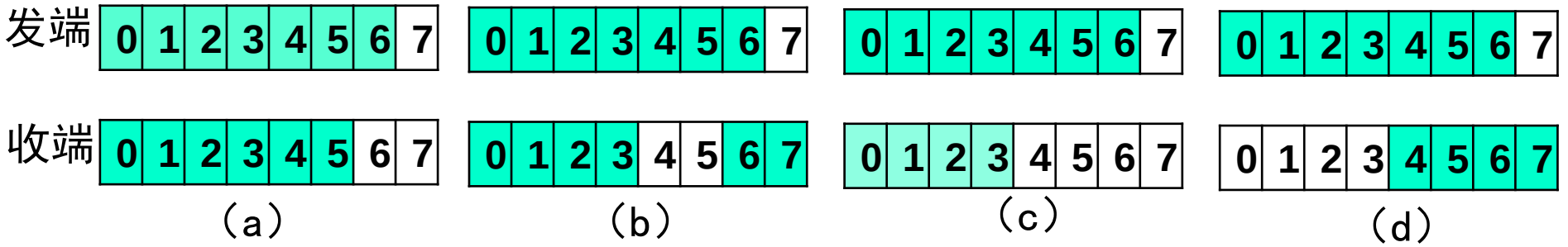
选择重传 ARQ

- 增大接收窗口，接收端可以接收序号不连续但仍在接收窗口中的数据帧。等到收到所缺序号的数据帧后再送交主机
- 选择重传 ARQ 避免重传那些正确到达接收端的数据帧。但代价是接收端需要缓存多帧数据
- 对于选择重传 ARQ 协议，若序号用n比特编码，则接收窗口的最大值为

$$W_R \leq 2^n/2$$

- 为何接收窗口的最大值为$2^n/2$?

选择重传ARQ：接收窗口



- 设帧序号用3位编码，发送窗口为7，接收窗口为6

(a)发送0~6帧后收端确认0~5，但确认帧全部丢失，发端窗口为0~6，发端重发0~6
(b)收端发送了确认帧后，接收窗口调整为6~3；

- 问题：收端在接收窗口内的帧都应是新帧，但是6帧为新帧而0~3帧为旧帧，在接收窗口中新旧序号重叠，收端无法区分；
- 解决：避免接收窗口中新旧序号重叠！接收窗口为4（$=2^n/2$）

(c) 发送0-6帧，收端确认0~3，但确认帧全部丢失，发端重发0~6帧

(d) 接收端发送了确认帧后窗口为4~7，接收了4~6帧拒收了0~3帧；确认4~6帧后，接收窗口调整为7~2；发送窗口调整为7~5，继续……

累计确认与捎带确认

- 发送数据帧是有效传输，而确认帧则是协议开销；如何降低开销？
- 尽可能降低发送确认帧的数量，采用累计确认和捎带确认
- 累计确认：收端不是每收到一个数据帧就发送一个确认帧，若发送确认序号为i，则表示确认(i-1)及其之前的所有数据帧且期望收到的下个帧的序号为i
- 捎带确认：当接收端向发送端发送确认序号时，若有需要发送的数据帧，那么将确认信息和数据合并在一个帧中发送

问题

- 设置发送窗口需要考虑哪些因素？
- 设置超时定时器的长度与哪些因素有关？
- 实际系统的链路速率、传输时延、缓存能力差异很大，如何选择发送窗口以及超时定时器？
  - TCP给出了最佳的解决方案，详见传输层

- 链路层标准协议：HDLC，PPP

小结

- 数据链路层，基于比特流组成帧流
- 组帧方法：帧格式及其功能（帧头、类型、数据、校验）
- 差错编码：检错及纠错
- 差错控制、流量控制
- 典型的链路层控制协议及信道利用率
  - 停等
  - 回退N
  - 选择重传

练习题

- 名词解释：捎带确认，滑动窗口协议
- 试证明连续ARQ协议，发送窗口的极大值为2^n-1，其中，n为帧编号所用比特数。
- 采用滑动窗口协议在3000Km的T1链路上传输64B的帧，若信号传播速度为6μs/km，问序号应为多少位。（参考答案：7位）
- 利用地球同步轨道的卫星，在1Mbps的信道上发送1000b的帧。信号传播时间为270ms，采用捎带确认，假设帧头很短，序号占3位，信道上传输无错帧，分别采用（1）停等协议（2）滑动窗口协议，求信道利用率。（参考答案：全双工信道上0.18%，1.29%，在半双工信道上0.18*2%，1.29*2%，）

作业1：编程实现滑动窗口协议

- 编程实现：
  - 1）1比特滑动窗口协议（停等协议）
  - 2）回退N滑动窗口协议
  - 3）选择重传协议
- 利用Netriver平台编程并测试，以验证协议的正确性
- 提交：源程序，实验报告，
- 截止时间：4月11日（周日）

第3部分： 数据链路层

已附加文件: CH3-2-数据链路层 - ARQ.pdf (1.625 MB)
CH3-1-数据链路层--成帧及检错.pdf (1.68 M

教学目标： 掌握主要的差错控制技术，以及链路协议的分析方法
主要内容：

1）数据链路层的功能，差错控制技术，帧校验方法
2）停等协议
3）滑动窗口协议
4）HDLC协议及PPP协议

作业：

编程实现滑动窗口协议 **4月11日** 之前完成

参考资料：

1）NetRiver网络实验系统实验指导书（清晰版）.pdf

2）introduction to netriver.mp4

（制作人：陈天宇 助教）

停等协议测试函数序框架

```
#define WINDOW_SIZE_STOP_WAIT 1
#define WINDOW_SIZE_BACK_N_FRAME 4
typedef enum{data,ack,nak} frame_kind;
typedef struct frame_head
{       frame_kind kind;
        unsigned int seq;
        unsigned int ack;
        unsigned char data[100];
};
typedef struct frame
{       frame_head head;
        unsigned int size;
 };
```

停等协议测试函数框架

```
int stud_slide_window_stop_and_wait(char *pBuffer,
                                    int bufferSize, UINT8 messageType)
{       frame*buffer;
        if(messageType==MSG_TYPE_SEND)
        {……}          //发送，添加到发送队列，若无需停等则启动发送定时
        else if(messageType==MSG_TYPE_RECEIVE)
        {……  }        //接收，确认队首已收到，结束定时器
        else if(messageType==MSG_TYPE_TIMEOUT) //超时重发
        {       ……   } //超时重发
        else
            return -1;
        return 0;
}
```

注意问题：

- 1.网络字节序与主机字节序可能不同，需要转换；即在发送数据之前调用htonl，接收网络数据后调用ntohl转为主机序，之后再处理
  - 大端法：最高有效字节在最前面的方式，例如0x11223344，其存储方式为0x00:11,0x01:22,0x02:33,0x03:44
  - 小端法：最低有效字节在最前面的方式，正好和大端法相反
- 2. 回退 N滑动窗口协议在超时情况下， 本应重发与超时帧 seq值相同的帧及其后续帧；然而实验中需要将滑动窗口中的所有帧重发
- 3.在回退N滑动窗口协议与选择性重传协议中， 均采用累计确认。

通知： NetRiver实验系统介绍

- 3月25日（周四）18:40-19:30,理科1号楼1235机房西大厅
- NetRiver系统组成：Web服务器+协议测试服务器
- C++编程实现协议，参见NetRiver网络实验系统实验指导书.pdf（网站）