数据链路层滑动窗口协议的设计与实现

一、 实验内容和实验环境描述

本次实验的任务：

利用所学数据链路层原理，自己设计一个滑动窗口协议，在仿真的环境下编程实现有噪音信道环境实现两站点之间无差错双工通信。通过该实验，进一步巩固和深刻理解数据链路层误码检测的 CRC 校验技术，以及滑动窗口的工作机理。

本次实验的内容：

信道模型为 8000bps 全双工卫星信道。信道传播时延 270ms，信道误码率为 10-5，信道提供字节流传输服务，网络层分组长度固定为 256 字节。

滑动窗口机制的两个主要目标：

(1)实现有噪音信道环境下的无差错传输；

(2)充分利用传输信道的带宽。

在程序能够稳定运行并成功实现第一个目标之后，运行程序并检查在信道没有误码和存在误码两种情况下的信道利用率。 为实现第二个目标，提高滑动窗口协议信道利用率，需要根据信道实际情况合理地为协议配置工作参数，包括滑动窗口的大小和重传定时器时限以及 ACK 搭载定时器的时限

本次实验的实验环境：

Windows 环境 PC 机，Microsoft Visual Studio 2017+集成化开发环境。

二、 软件设计*

给出程序的数据结构，模块之间的调用关系和功能，程序流程。

(1) 数据结构：数据结构是整个程序的要点之一，程序维护者充分了解数据结构就可以对主要算法和 处理流程有个基本的理解。描述程序中自定义结构体中各成员的用途，定义的全局变量和主函数中的变量的变量名和变量所起的作用。

数据帧

KIND(1)	SEQ(1)	ACK(1)	DATA(240~256)	CRC(4)

确认帧

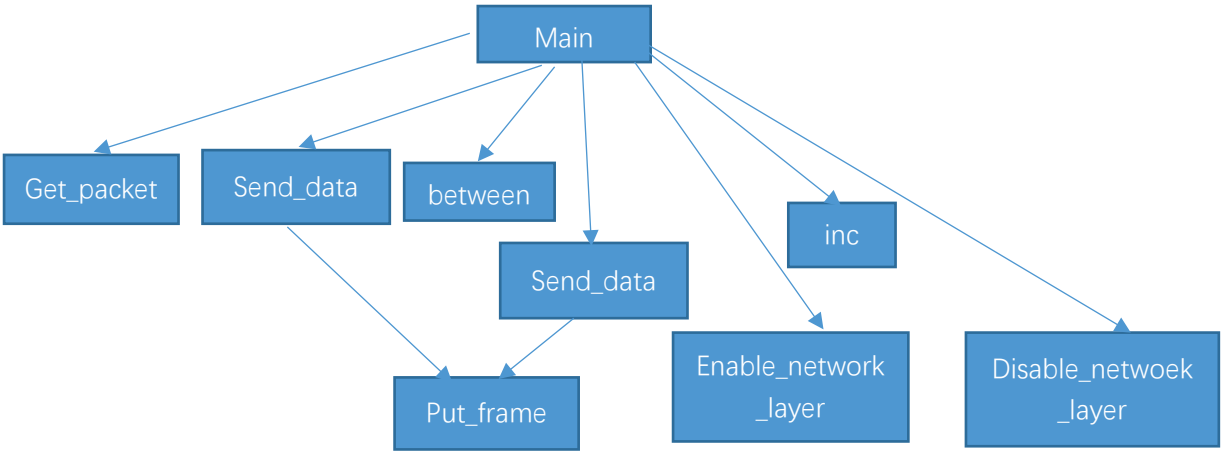
KIND(4)	ACK(1)	CRC(4)

否定确认帧

```
+========+======+=======+
|KIND(1) |ACK(1)|CRC(4) |
+========+======+=======+
```

KIND:表示帧的类别
ACK：ACK 序列号
SEQ：帧序列号
CRC：校验和

(2) 模块结构：给出程序中所设计的子程序所完成的功能，子程序每个参数的意义。给出子程序之间 的程序调用关系图。



#define inc(k)if(k<MAX_SEQ)k++;else k=0

作用：使一个字节在 0~MAX_SEQ 的范围内循环自增。
参数：a,b,c,字节类型。

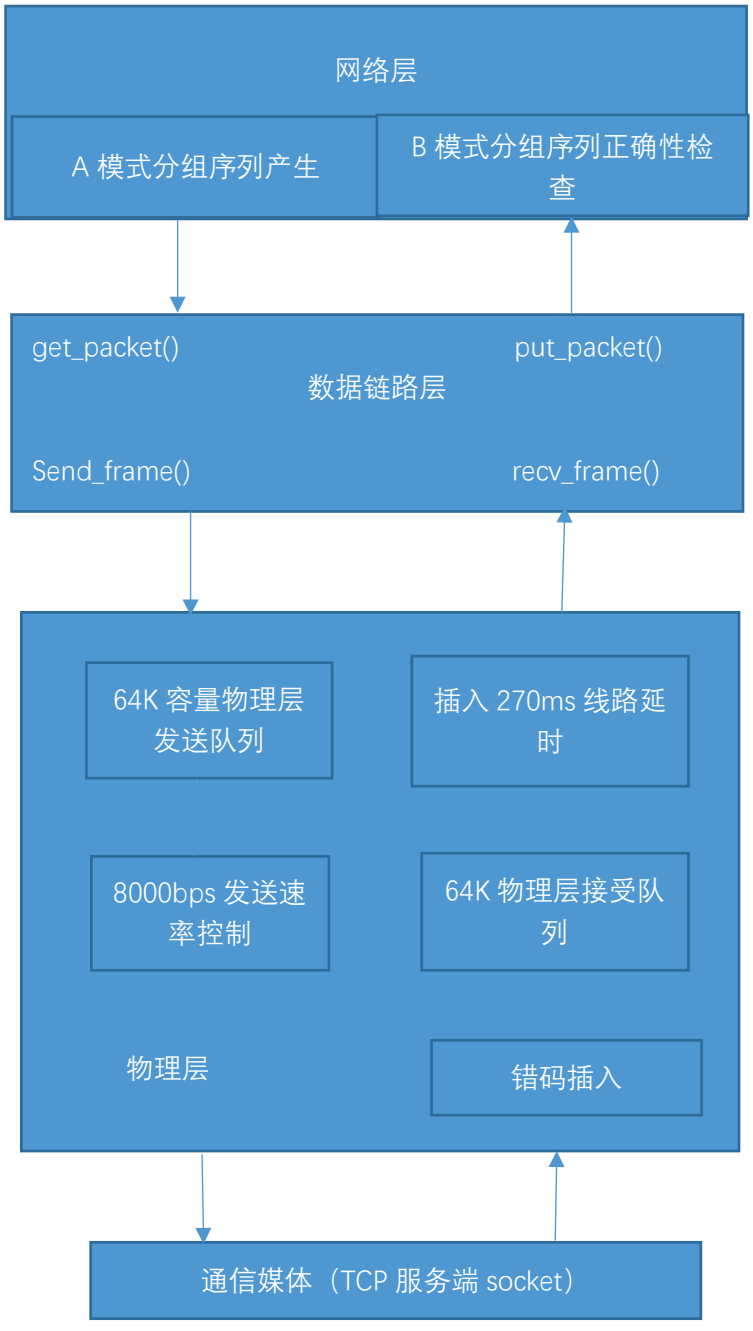
static int between(unsigened char a,unsigned char b,unsigned char c)
作用：判断当前帧是否落在发送/接收窗口内。
参数：a,b,c，均为字节类型，其中两个分别为窗口的上、下界，一个为帧的编号。其中，发送窗口的上界和下界分别为 next_to_send 和 ack_expected，接收窗口的上界和下界分别为 too_far 和 frame_expected，均定义在 main 函数中。

static void put_frame(unsigned char *frame, int len)
作用：为一个帧做 CRC 校验，填充至帧的尾部并将其递交给网络层发送。
参数：frame，字节数组，由除 padding 域之外的帧内容转换而来；len，整型，为帧的当前长度。

Static void send_data(unsigned char type, unsigned char frame_nr)
Type 分成三种情况；
Switch(type)
Case FRAME_DATA 发送一个帧；
Case FRAME_ACK 发送 ACK 确认信号；
Case FRAME_NAK 发送 NAK 信号。
作用：构造一个帧，并将其发送。
参数：type，字节类型，为帧的内容；frame_nr，字节类型，为帧的编号；

Bool Bad_Package(const int len)
作用：确认是否坏帧；
参数：len，const int 型，帧的长度；
int main(int argc,char **argv)
作用：主程式，包含选择重传协议的算法流程。
参数：argc，整型，表示命令行参数的个数；argv，二维字符数组，表示参数内容。

(3) 算法流程：画出流程图，描述算法的主要流程。



以上是 A 到 B 的算法流程，B 到 A 的也是如此，只需要把 AB 互换即可。

程序开始，首先是网络层的准备，得到包，发送数据，
然后是物理层的准备，没有准备好就 break，准备好了则发送帧；
其次是物理层的准备，得到数据后发送帧，物理层准备好，break；
然后是数据接收，计数校验和 csc，找到帧的末尾，csc 等于 0 时，S_seq=frame 帧的序列号，封装进包 packet，发送 ack 确认信号，In_len=1；数据超时时，重设 ack num，next=ack expected，物理层准备好时，发送帧，get_num=nbuffered,break；ack 超时时，重新创建 ack 帧，发送这个帧，break；nbuffered<max_seq 时，enable network，反之，disable network。

三、 实验结果分析

(1) 所完成的 Go-Back-N 协议和 SR 协议均实现了有误码信道环境中无差错传输的功能，对于错误的桢、丢失的帧 能都检测并重新传输（这里不考虑 CRC 校验和中， 一个不正确的帧被当作有效帧接受的概率是 2^{1-r}）

(2) 协议健壮性较强，能够长时间可靠的运行（测试时长一晚上，程序正常运行，并未出现异常）

(3) 协议参数的选取：

SR 协议：

滑动窗口的大小：16

重传定时器的时限：3000ms

ACK 搭载定时器的时限：1000ms

我们物理层提供的是字节流传输服务，使用字节填充技术成帧，分组长度为 256 字节。

为了避免在有出错帧接收方要求重传时产生二义性，我们定义窗口大小为 2^n-1，并且双方的窗口大小均为((MAX_SEQ+1)/2)，这样的大小足够使用又不会有过于富余的空间浪费。滑动窗口的大小直接涉及到信道利用率和数据拥塞问题，若太大，数据发送过快将产生拥塞导致数据丢失，出错率增加，若太小则信道利用率降低，通过实验测试合适的窗口大小为 16。

重传定时器时限涉及到重传的响应时间，太小会导致频繁重传，太大则重传等待时间太久，经过我们的试验测试，选取重传定时器时限定为 3000 毫秒，ACK 搭载定时器的时限为 1000 毫秒最合适。

(4) 理论分析：根据所设计的滑动窗口工作机制(Go-Back-N 或者选择

重传)，推导出在无差错信道环 境下分组层能获得的最大信道利用率；推导出在有误码条件下重传操作及时发生等理想情况下分组层能 获得的最大信道利用率。给出理论推导过程。理论推导的目的是得到信道利用率的极限数据。为了简化 有误码条件下的最大利用率推导过程，可以对问题模型进行简化，比如：假定超时重传的数据帧的回馈 ACK 帧可以 100%正确传输，但是简化问题分析的这些假设必须不会对整个结论产生较大的误差。

假设(SR)： 一个数据帧中有效数据为 N byte

一个数据帧长度为 L byte

数据发送速率 R=8000bps

发送时延 T_f

实际发送一个帧平均时间 T_{af}

窗口大小 W

一个数据帧出错的概率为 $P_f = 1 - (1 - 10^{-5})^{8L}$

一个数据帧重传概率为 $P = 2P_f - P_f^2$

一个窗口出错概率为 $\frac{P}{W}$

$T_{af} = T_f + T_f(1 - \frac{P}{W})\sum_{i=0}^{\infty} i(\frac{P}{W})^i = T_f \frac{W}{w-P}$

线路利用率：$\frac{N}{L}\frac{T_f}{T_{af}} = \frac{N}{L}\frac{w-P}{W}$

无误码信道利用率=$\frac{N}{L} = \frac{256}{263} = 97.338\%$

误码率为10^{-5} 的信道利用率(W=8)= $\frac{N}{L}\frac{w-P}{W} = 96.85\%$

误码率为10^{-5} 的信道利用率(W=16)= $\frac{N}{L}\frac{w-P}{W} = 97.09\%$

由此可见当 MAX_SEQ 取值为 15 和 31 时效率相差不大，但 31 时更好

(5) 实验结果分析：你的程序运行实际达到了什么样的效率，比对理论推导给出的结论，有没有差距？ 给出原因。有没有改进的办法？如果没有时间把这些方法付诸编程实施，介绍你的方案。

由实验结果可以看出，窗口大小是 16 时信道利用率最高，即 MAX_SEQ 取值 31，与理论最大值最接近。与理论窗口最佳值一致。

在洪泛模式下，信道利用率普遍要比不在泛洪模式下的要高，而且信道利用率与理论值接近。而在其他情况下负载有时较轻，所以信道利用率相比理论值较低

(6) 存在的问题：在“表 3 性能测试记录表”中给出了几种测试方案，

在测试中你的程序有没有失败，或者，虽未失败，但表现出来的性能仍有差距，你的程序中还存在哪些问题?

实验顺利进行，每次测试得到的数据都比较接近，但和理论相比信道利用率较低，无法达到理论的信道利用率这与信道的负载有关。

实验结果：
窗口大小 16，DATA_TIMER 3000,ACK_TIMER 1000

性能测试记录表

序号	命令选项	说明	运行时间(分钟）	Selective 算法线路利用率(%)		
				A	B	
1	--utopia	无误码信道数据传输	10	51.93	96.97	
2	无	站点 A 分组层平缓方式发出数据，站点 B 周期性交替“发送 100 秒，停发 100 秒”	10	52.85	94.44	
3	--flood --utopia	无误码信道，站点 A 和站点 B 的分组层都洪水式产生分组	10	96.96	96.97	
4	--flood	站点 A/B 的分组层都洪水式产生分组	10	95.35	95.10	
5	--flood --ber=1e-4	站点 A/B 的分组层都洪水式产生分组，线路误码率设为 10^{-4}	10	60.38	59.85	

运行时间要求超过 10 分钟。

四、 研究和探索的问题

前面列出的“可研究和探索的问题”，有哪些问题你有了答案或者自己的见解？给出你的结论，并详 细阐述你的理由或见解。

(1)CRC 校验能力

本次实验采用的 CRC 校验方案为 CRC-32，与 IEEE 802.3 以太网校验和生成多项式相同。生成多项式为：

x32+x26+x23+x22+x16+x12+x11+x10+x8+x7+x5+x4+x2+x1+1

除了性能优良外，该多项式还能检测到长度小于等于 32 的所有突发

错误，以及影响到奇数位的全部突发错误

带 r 个校验位的多项式编码可以检测到所有长度小于等于 r 的突发错误。
如果突发错误长度为 r+1，则当且仅当突发错误与 G（x）一致时，错误多项式除以 G（x）的余数才是 0.一个不正确的帧被当作有效帧接受的概率是 2^{1-r}
同样可以证明，当一个长度大于 r+1 位的突发错误发生时，或者几个短突发错误发生时，一个坏帧被当作有效帧通过检测的概率为2^{-r}，这里假设所有位模式都是等概率的。

(2)校验和的生成有硬件产生和软件计算两种方式。
硬件：
硬件产生通过移位寄存器电路实现。教科书中 给出了计算 CRC 校验和的方法：根据 CRC 多项式得到一个 33 比特的除数；将待校验的 n 字节帧理解为 8n 比特的二进制数，在后面追加 32 个 0 构成被除数；然后进行“模 2”除法，得到 32 比特余数作为校 验和。
软件：
由于一般通用的 CPU 未提供按位“模 2”除法的指令，需要通过软件模仿这种除法。目前，通过软件计算的方式求校验和，一般用查表并叠加的方式，将逐比特进行的模 2 除法改造为更高效率的逐字 节进行模 2 除法，程序库中直接给出了实现校验和的函数调用 crc32()。源程序参考了 TCP/IP 协议簇中的 PPP 协议相关文本 RFC1662。
函数 crc32()返回一个 32 比特整数。设指针 p 的定义为 char *p 并且 p 指向一个缓冲区，缓冲区内有 243 字节数据，为这 243 字节数据生成 CRC-32 校验和，并且把这 32 比特校验和附在 243 字节之后， 执行下面的语句：
```
*(unsigned int *)(p + 243) = crc32(p, 243);
```
注意：p 所指缓冲区必须至少有 247 字节有效空间，以防内存访问越界

五、 实验总结和心得体会

(1) 完成本次实验的实际上机调试时间是多少？
本次实验编写程序及调试共用 8 小时左右，主要是开始是线路利用率较低，然后写了 GBN 和 SR 两个协议

(2) 编程工具方面遇到了哪些问题？包括 Windows 环境和 VC 软件的安装问题。
我们用的是 VS 之前已经多次使用，没有遇到什么问题

(3) 编程语言方面遇到了哪些问题？包括 C 语言使用和对 C 语言操控

能力上的问题。
C 语言使用较为熟练，开始时对数据结构定义时不够完善，一些参数未定义初始值，经过检查之后修正

(4) 协议方面遇到了哪些问题？包括协议机制的设计错误，发现协议死锁，或者不能正确工作，协议 参数的调整等问题。
开始时协议运行一段时间就会自动崩溃，无法长时间可靠的工作，经过核对,发现一些参数写错了,经过调整之后协议可以正常运行，但是效率不理想，于是调整了窗口大小，超时计数器时长等参数，协议效率得到改善

(5) 开发库方面遇到了哪些问题？包括库程序中的 BUG，库函数文档不够清楚导致误解,库函数在 所提供的功能结构上的缺憾导致编程效率低下。这些问题或建议影响不同模块之间功能界限的划分。
没有遇到明确问题

(6) 总结本次实验，你在 C 语言方面，协议软件方面，理论学习方面，软件工程方面等哪些方面上有所提高
通过本次实验，我们对数据链路层的协议工作原理有了更加深刻的理解，对窗口大小，重传定时器的时限，ACK 搭载定时器的时限，NAK 对协议效率的作用等有了进一步认识。同时对 CRC 校验和的校验能力和校验原理有了更加透彻的理解

六、 源程序文件

源文件(.c)见附件。