

**计算机网络实验报告**

**学院：计算机学院**

**[组员：马天成]**

**[****组员：王宸]**

**[组员：罗帅]**

**2022年5月20日**

**[滑动窗口实验]**

**北京邮电大学《计算机网络》课程实验报告**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **实验**  **名称** | 数据链路层滑动窗口协议的设计与实现 | | **学 院** | 计算机 | **指导教师** | 蒋砚军 |
| **班 级** | **班内序号** | **学 号** | | **学生姓名** | **成绩** | |
| **2020211305** |  | **2020211376** | | **马天成** |  | |
| **2020211305** |  | **2020211383** | | **王宸** |  | |
| **2020211305** |  | **2020211409** | | **罗帅** |  | |
| **实**  **验**  **内**  **容** | 本次实验选用的滑动窗口协议为选择重传/回退N步协议，利用所学数据链路层原理，自行设计一个滑动窗口协议，在仿真环境下编程实现有噪音信道环境下两站点之间无差错双工通信。信道模型为8000bps全双工卫星信道，信道传播时延270毫秒，信道误码率为10-5，信道提供帧传输服务，网络层分组长度固定为256字节。  本次实验选用的滑动窗口协议为选择重传/回退N步协议，并且未使用/使用了NAK通知机制。 | | | | | |
| **学生实验报告** | （详见“实验报告和源程序”册） | | | | | |
| **课**  **程**  **设**  **计**  **成**  **绩**  **评**  **定** | **评语**:  **成绩**:  指导教师签名：  年 月 日 | | | | | |

注：评语要体现每个学生的工作情况，可以加页。

目录

[一.实验内容和实验环境描述 4](#_Toc103956512)

[二.协议设计&软件设计 4](#_Toc103956513)

[1.协议设计 4](#_Toc103956514)

[2.软件设计（GBN和Select不同的地方会标出） 4](#_Toc103956515)

[「1」数据结构 4](#_Toc103956516)

[「2」模块结构 6](#_Toc103956517)

[「3」程序总体机构和算法流程图 8](#_Toc103956518)

[「4」补充伪代码描述选择重传代码 9](#_Toc103956519)

[「5」简单的参数说明（以选择重传为例，为参数选择做铺垫） 9](#_Toc103956520)

[三.实验结果分析 10](#_Toc103956521)

[1. 功能性 10](#_Toc103956522)

[2. 健壮性 10](#_Toc103956523)

[3. 协议参数的选取 11](#_Toc103956524)

[滑动窗口 11](#_Toc103956525)

[Data\_Timer 11](#_Toc103956526)

[Ack\_Timer 11](#_Toc103956527)

[4. 理论分析 11](#_Toc103956528)

[Go\_Back\_N信道利用率 11](#_Toc103956529)

[Select\_Repeat信道利用率 11](#_Toc103956530)

[5. 实验结果 12](#_Toc103956531)

[Go\_Back\_N 12](#_Toc103956532)

[Select\_Repeat 12](#_Toc103956533)

[总表 13](#_Toc103956534)

[6. 实验结果分析 13](#_Toc103956535)

[7. 存在的问题 14](#_Toc103956536)

[四.研究和探索的问题 14](#_Toc103956537)

[1. CRC校验能力 14](#_Toc103956538)

[2. CRC 校验和的计算方法 14](#_Toc103956539)

[3. 程序设计方面的问题 15](#_Toc103956540)

[4. 软件测试方面的问题 15](#_Toc103956541)

[5. 对等协议实体之间的流量控制 16](#_Toc103956542)

[6. 与标准协议的对比 16](#_Toc103956543)

[五.实验总结和心得体会 16](#_Toc103956544)

[附录：源代码（只展示datalink.c） 18](#_Toc103956545)

[Go\_Back\_N（不携带Ack） 18](#_Toc103956546)

[Go\_Back\_N++（携带Ack） 20](#_Toc103956547)

[Select\_Repeat（有Nak） 22](#_Toc103956548)

# 一.实验内容和实验环境描述

设计一个滑动窗口协议，在仿真环境下编程实现有噪音信道环境下

两站点之间无差错双工通信，并且需要充分利用传输信道的带宽。

1.信道模型

信道模型为8000bps全双工卫星信道，信道传播时延270毫秒，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。

2.设备环境

（1）Windows环境PC机；

（2）Microsoft Visual Studio集成化开发环境。

3.编程环境

由实验指导教师提供协议软件设计的基本程序库，利用仿真环境下所提供的物理层服务和定时器机制为网络层提供服务。可执行程序中包括物理层，数据链路层和网络层三部分内容，其中，物理层和网络层程序由程序库提供。

# 二.协议设计&软件设计

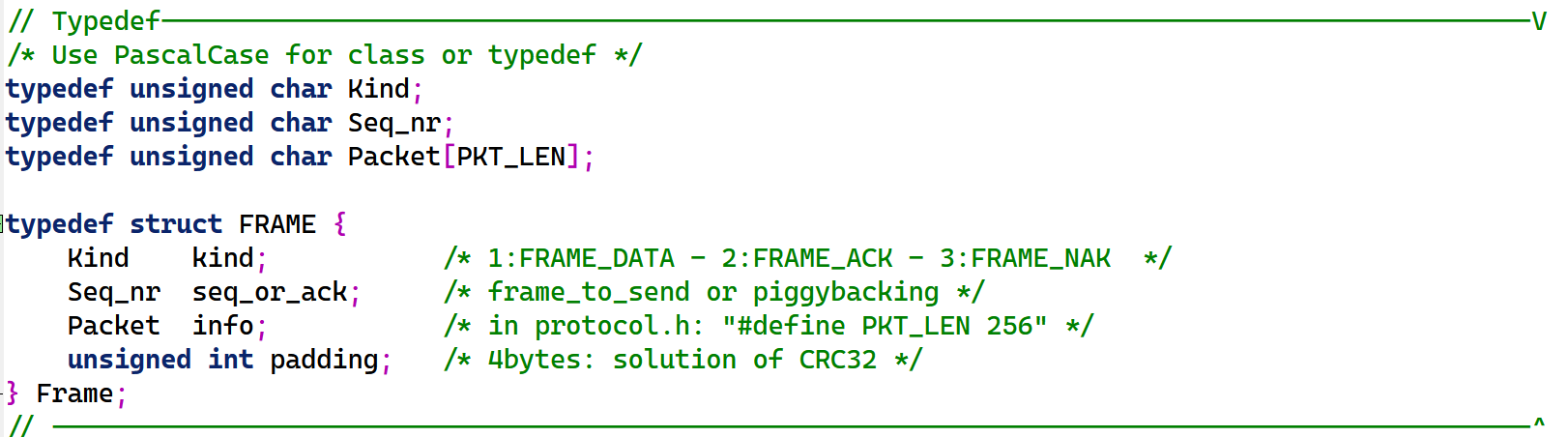
## 1.协议设计

* 网络层：利用数据链路层提供的“可靠的分组传输”服务，在站点 A 与站点 B 之间交换长度固定为256 字节的数据分组。网络层把产生的分组交付数据链路层，并接受数据链路层提交来的数据分组。
* 数据链路层：通过get\_packet( )函数从网络层接收要发送的数据包，将之组装成帧，向物理层发送，启动计时器；进行适当的流量控制；数据帧经信道传送给接收方；接收方数据链路层终止定时器（或启动 ack 定时器，ack 成帧传送），判断数据是否出错，若正确，则判断是否为预定接受数据，最终将其通过put\_packet( )函数提交给网络层。
* 物理层：为数据链路层提供的服务为 8000bps，270ms 传播延时，10-5 误码率的字节流传输通道。

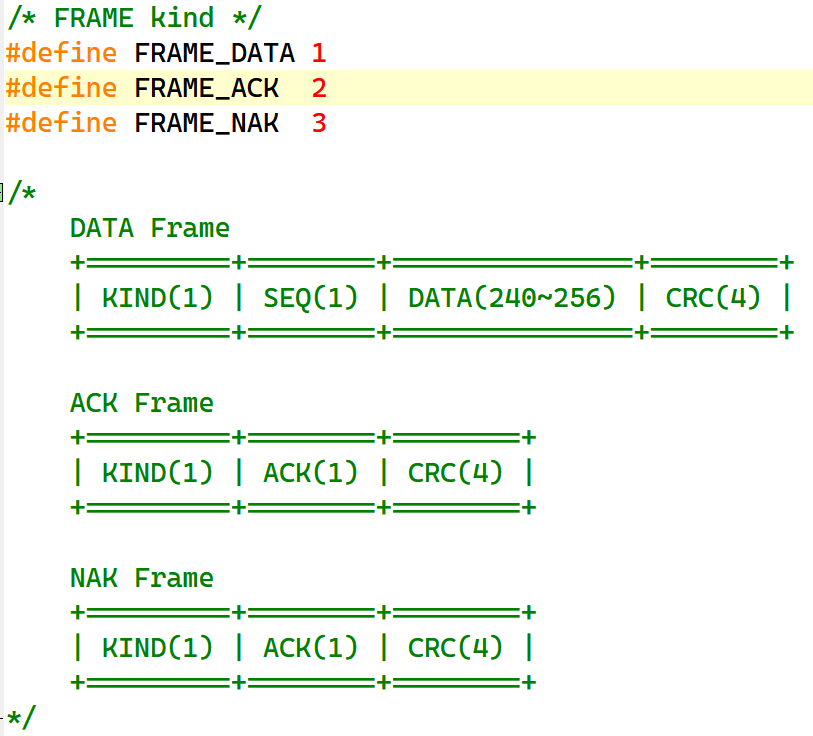
## 2.软件设计（GBN和Select不同的地方会标出）

### 「1」数据结构

（1）变量类型&帧结构定义（全局数据结构）

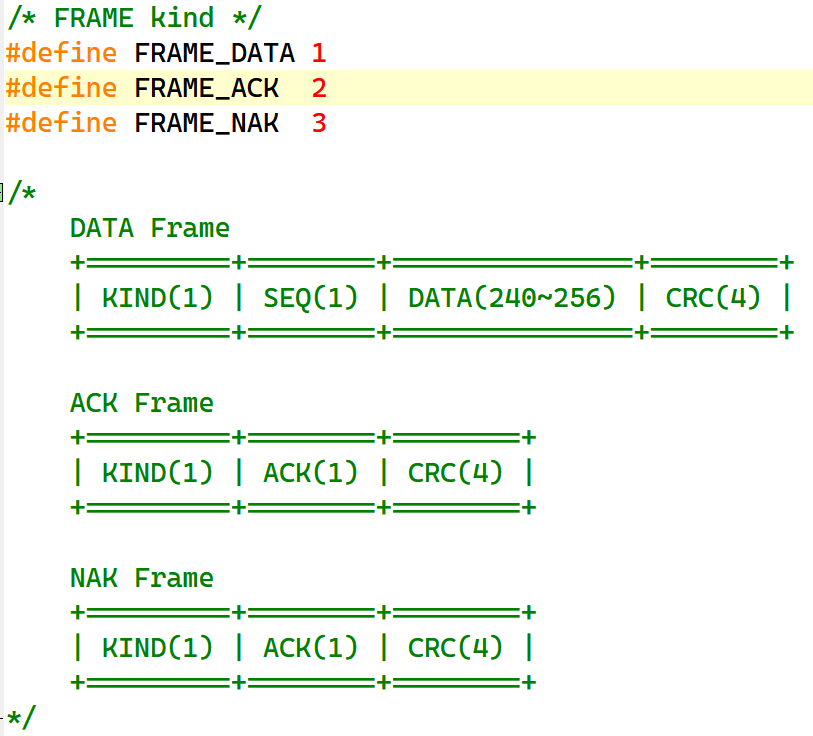


（2）帧类型定义和形式结构

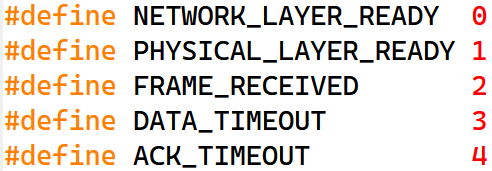


（3）一些宏定义：

帧类型：



（4）事件类型：



（5）参数赋值：(select举例)

#define DATA\_TIMER 3712 //数据定时器

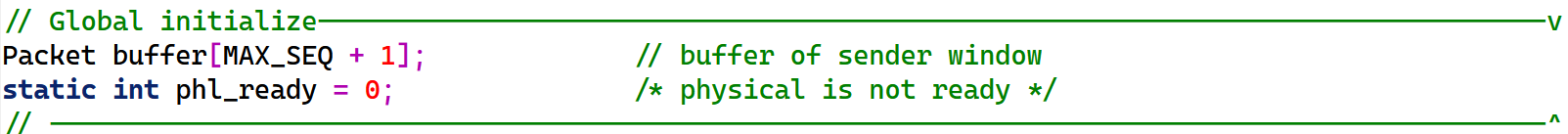
#define ACK\_TIMER 263 //ACK定时器

#define MAX\_SEQ 15 //2\*最大窗口-1

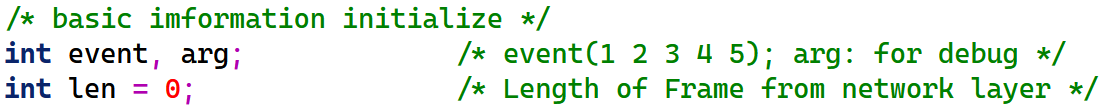
（6）序号增加：（方便窗口滑动）



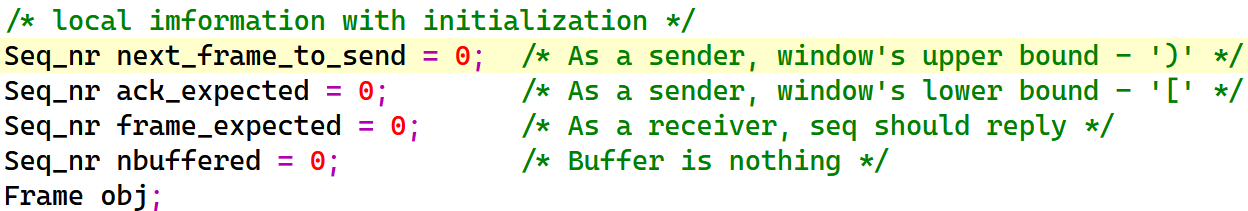
（7）全局变量定义：**// 缓冲区** **//物理层是否准备好**



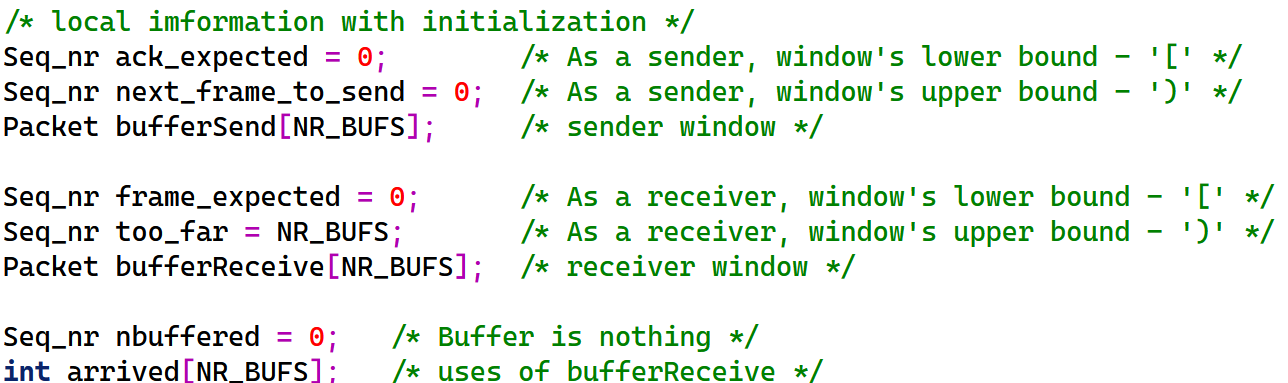
（8）主函数变量定义：



（8-1）Go\_Back\_N 变量：



（8-2）Select变量：

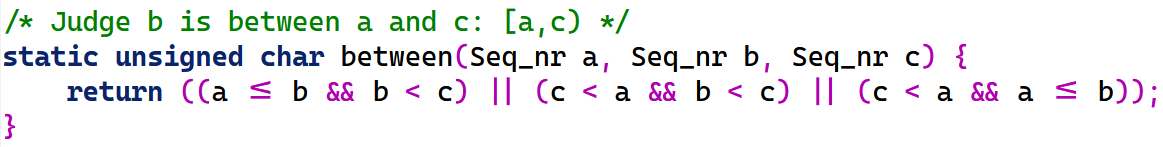


Frame obj;

### 「2」模块结构

（1）函数定义：

（1-1）判断帧是否在窗口内

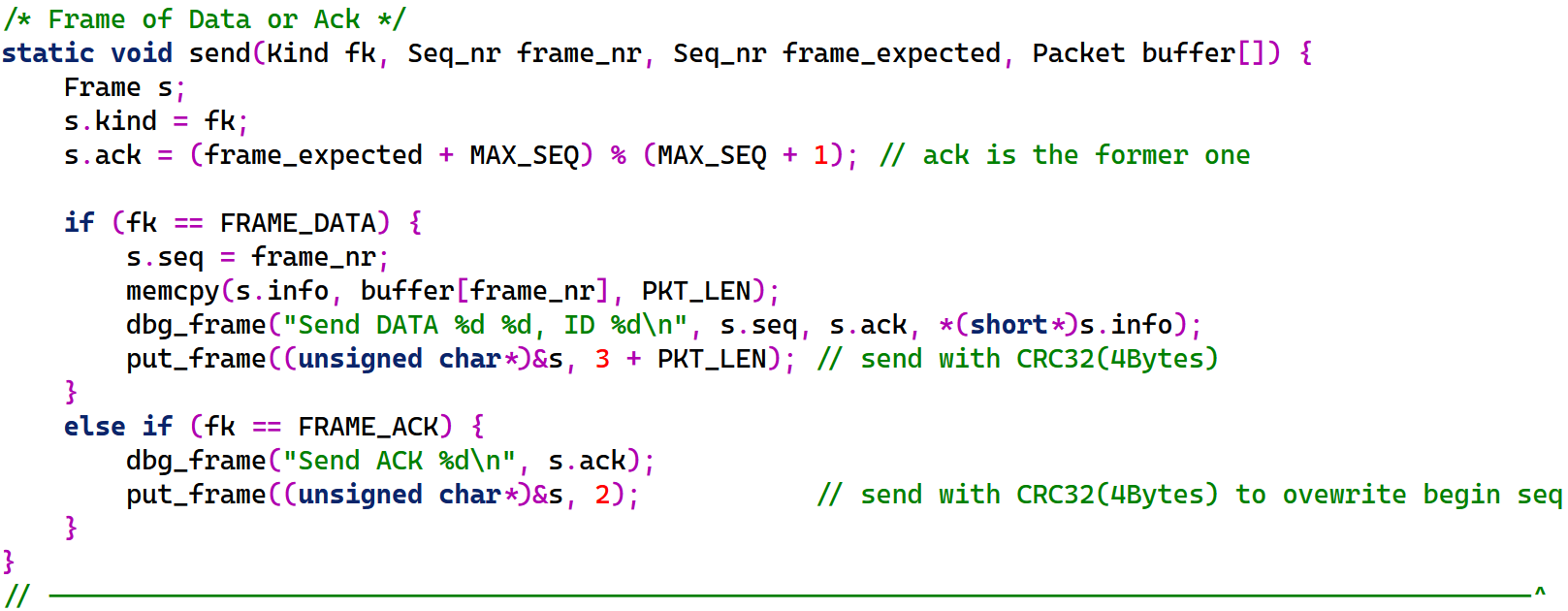


（1-2）计算帧的校验和，添加在帧尾，发送至物理层（划分）

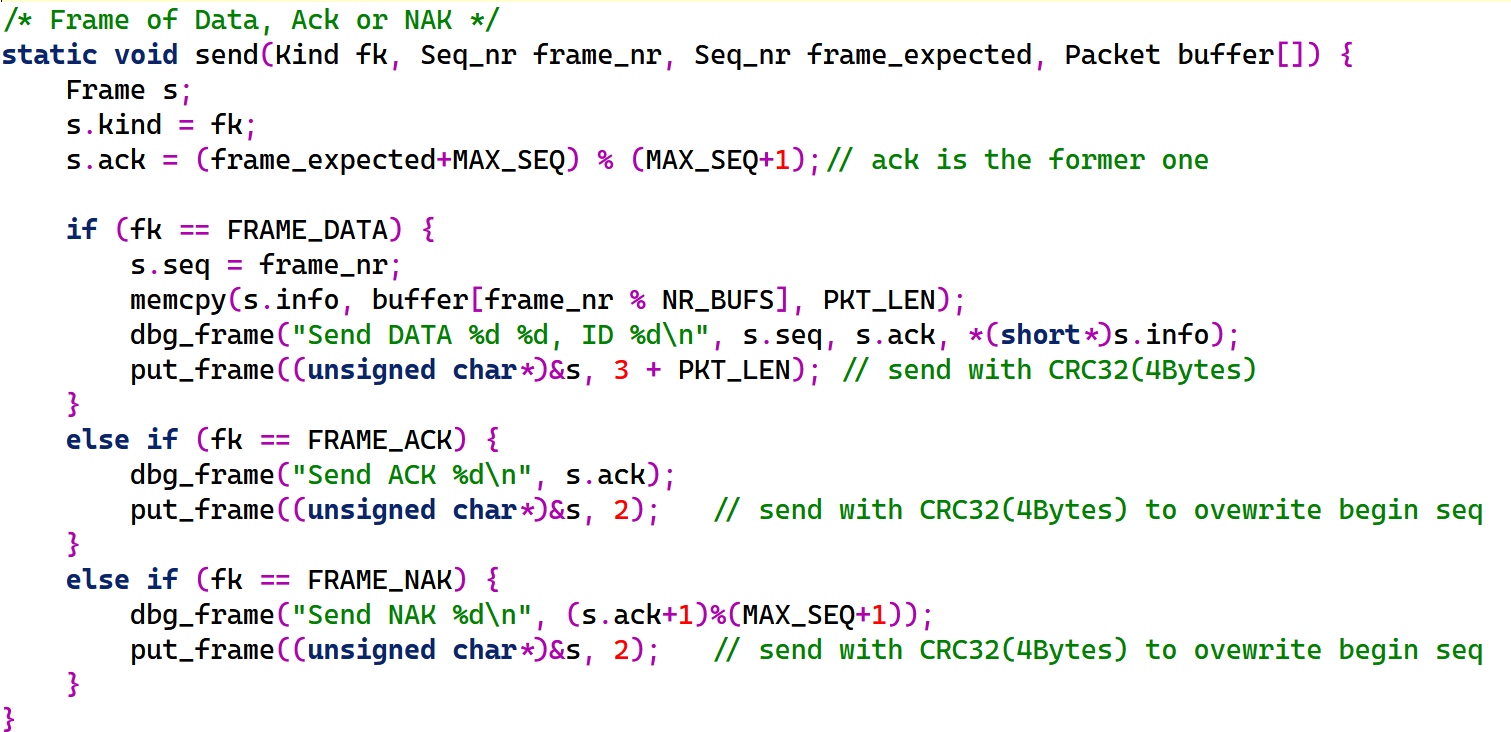
**这里用到了一个CRC校验函数**

**static void create\_crc(unsigned char \*frame, int len , other\_param \*nak\_and\_phl)**

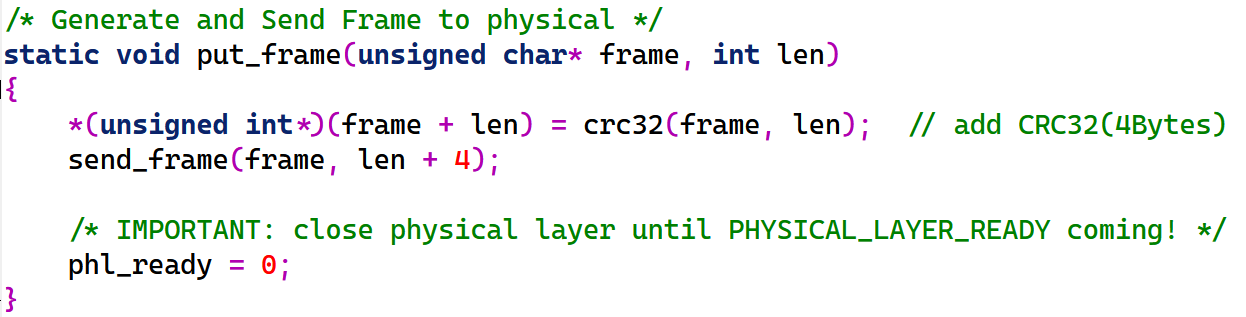
1.**Go\_Back\_N**（主要是没有nak）



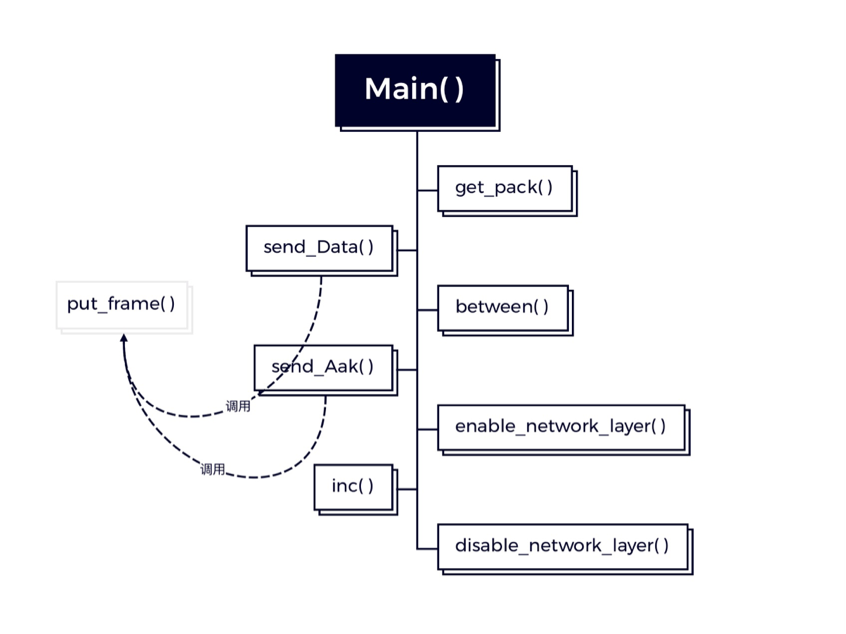
2.**Select**



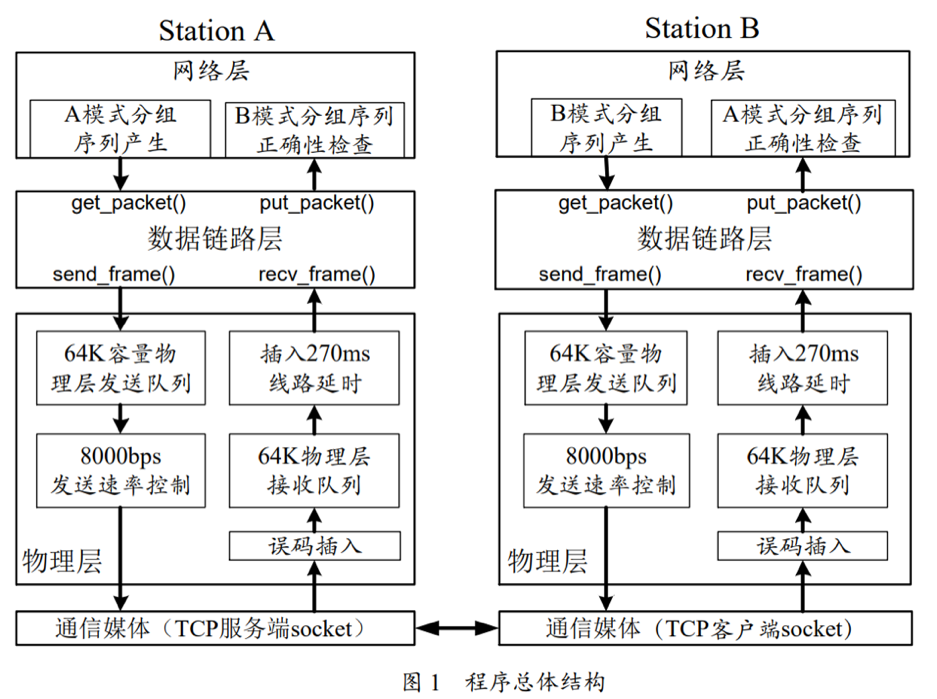
（1-3）向物理层发送数据帧



（2）函数之间的调用关系：



### 「3」程序总体机构和算法流程图



### 「4」补充伪代码描述选择重传代码

**//--------------------------------------------------------------------**

Main()

{

While(1)

{

event = wait\_for\_event(&arg);

Switch(event)  
 {

Case 网络层准备好：

发送一个新的数据包 ; break;

Case 收到一个包：

校验CRC失败发送NAK；

CRC校验成功：

{

如果是数据帧：

判断序号是否在窗口

判断成功：

将包送到网络层

否侧：

发送NAK；

如果是NAK帧

重发数据包；

滑动窗口

}

Case 数据定时器超时：

重发数据包

Case ACK定时器超时：

发送ACK帧；

}

}

}

**//--------------------------------------------------------------------**

### 「5」简单的参数说明（以选择重传为例，为参数选择做铺垫）

* 信道模型参数：8000bps , 270ms延迟，网络层长度固定为256B。

所以一个完整的数据包长度为256+3+4=263B，一个完整的NAK/ACK包长度为7B;

* 窗口大小：现在考虑理想情况下，即无误码，丢包，数据充足情况下：

当发出1号包后，马上发2号包 ， 就要保证在n号包发出后，1号包的ack已经回来。而ACK回来的最短时间为(270ms + 263B \* 8 \* 1000ms / 8000bps) \*2 = 1066ms,说明在1066ms内，发送方的的窗口应该还没有到最大值，即最大窗口为1066/263=4.05，所以窗口选择4合理。不过考虑到信道有误码率等情况，为了避免由于丢包的情况导致1号的ACK回来超过预期时间从而造成的1号窗口的等待，故选择8个窗口。

* 定时器：分为数据定时器和Ack定时器
  + 数据定时器：考虑到信道误码率为10^-5，即单个比特发生错误的概率为10^-5，则一个完整的数据包发生错误的概率为：1-(1-10^-5)^(273\*8) = **0.0216**。

考虑到采用的模型，发送方发出一个数据包，接受方接受到一个无误的数据包，发送一个数据包搭载ACK，假设接收方发出的数据包产生错误，则发送NAK帧，接收方再次发出搭载ACK的数据包，这个过程消耗的时间为270\*4+263\*3+7=1856，而此时发送方没有接收到ACK的概率是0.0216^2=0.0004.

如果该过程在发生一次，则时间为3712，发送方没有接收到ACK的概率是0.0216^4=0.0000002，该概率已经很少，基本不太可能发生，所以可以 保证在3712ms内，发送方一定可以得到ACK，所以数据定时器选择3712ms.

* + ACK定时器：初步设置成一个包发送需要的时间263，没有比较好的方法确定

# 三.实验结果分析

## 1. 功能性

能完成传输功能，且利用率都还不错。搭载ack，不搭载ack，回nak都能很好的运行。约20分钟后达到一个稳定的状态。

## 2. 健壮性

完成的所有代码都能在老师所给的信道上运行1小时以上。（只不过偶尔会因为系统繁忙而sleep一下）

## 3. 协议参数的选取

### 滑动窗口

Go\_Back\_N：上述理论可知，窗口大小大约在4以上。经过测试，7比较好。

Select：根据上述理论得知，窗口选取8。

### Data\_Timer

考虑：在没有传输错误的情况下，窗口用完就立即进行第一个帧的重发，避免耽误时间。

Go\_Back\_N：7\*263，设置为2000ms

Select：270\*4+263\*3+7=1856 1856\*2=3712ms

### Ack\_Timer

思路：配合Data\_Timer使用，重传之前需要之前回，窗口循环前就回。

感觉并没有太好的方法，经过资料的查询基本都是200-300左右的时间设置。

Go\_Back\_N：263ms

Select：263ms

## 4. 理论分析

### Go\_Back\_N信道利用率

同selective一样，理论上每发送48个数据包中第一个包出错，则此时的信道利用率 为：

 x100%

其中N为发送窗口，当N=7时，信道利用率为84.94%。当信道误码率为1e-4时，同样的方法可以得出每发送5个包就有一个包发送错误，此时信道利用率为40.55%

### Select\_Repeat信道利用率

1. ：理想情况

由于每一帧中具有CRC校验和ACK等额外信息，所以在最理想情况下信道最大利用率为：

**256/263=97.19%**

1. ：有误码情况下

已知信道误码率为1e-5，即可认为发送100000个比特就会有一个比特错误，100000个比特对应数据包为48个，即每发送48个数据包，就有1个包会发生错误，导致重传，此时的信道利用率为：

**256\*48/263\*49 = 95.53%**

## 5. 实验结果

### Go\_Back\_N

**1. 信道1**

****

****

**2. 信道2**

****

****

**3. 信道3**

****

****

**4. 信道4**

****

****

**5. 信道5**





### Select\_Repeat

**1. 信道1**





**2. 信道2**





**3. 信道3**





**4. 信道4**





**5. 信道5**





### 总表



## 6. 实验结果分析

比较和老师提供的数据得知：  
在平稳的传输信道上，我们的协议还可以。

但是在洪水的时候就会出现利用率比较低的情况。

* 首先给出书上的结论：

**Go\_Back\_N适合低误码率信道；Select\_Repeat适合高误码率信道。**

* 但是我们发现，在这里select普遍高于gbn：

因为10e-5这个误码率已经很高了。但是这个引起的误差也不是很大。

* 面对第四五组数据，我们发现相差很大：

因为GBN面对大量的数据输入，误码率提高时，帧的运输要求量大，使得超时的时候重传的帧会使得多个帧一起重传，而这就浪费了很多带宽。

* 信道5的利用率太小了。因为误码率太大，帧的错误率基本高达

1 - (1-0.0001)^(273\*8) = **0.1972，五个里面错一个！**

**所以GBN显得非常离谱，select也会显得比较慢。**

* GBN不携带ack和携带ack的区别：

发现携带的普遍利用率高于不携带的（很显然），但是这个差距很不明显。

## 7. 存在的问题

在运行过程中，我们发现信道4的利用率是一直在涨的，直到92-93左右停止，这个收敛速度感觉比较慢，可能ack\_timer变动一下应该会收敛的更快（比如调成300ms）

# 四.研究和探索的问题

## 1. CRC校验能力

CRC校验码的检错能力很强，具有以下检错能力：

* CRC校验码能检查出全部单个错；
* CRC校验码能检 查出全部离散的二位错；
* CRC校验码能检查出全部奇数个错；
* CRC校验码能检查出 全部长度小于或等于K位的突发错；CRC校验码能以[1-（1/2）^( K-1)]的概率检 查出长度为（K+1）位的突发错。

由于本次实验过程的误码信道是一个比较固定的误码率，而在实际生活当中的误码率不是稳定的，可能会因为传输环境的不同，使得他的误码率波动比较大，传输的距离也是影响因素。 对于这种动态的误码率的通信过程，可能需要其他的一些参数来控制基本参数值 （窗口大小，重传时间等等）来完成。

## 2. CRC 校验和的计算方法

在作业题中我就明显感觉到他和异或的极为相似之处了。虽说是除法套减法，但这个减却是异或。所以不由得想到用异或去运算整个CRC校验

答：两者等价。模二减法（异或运算）的性质：

**交换律：a ^ b = b ^ a  
结合律：a ^ b ^ c = a ^ (b ^ c)**

分解计算步骤并运用模二减法的交换律和结合律（先算某部分的CRC码，再与原始数据的某部分作模二减法，该部分为**两位处理**）抽象出数学模型：将计算CRC校验值过程记为函数f()；第一个两位记作sh1；第二个两位记作sh2。则得到：

**f(sh1 << 2 + sh2) = (f(sh1) << 2) ^ f((f(sh1) >> 2) ^ sh2)**

得到按两位查表算法的基本原理，等式右侧的f()函数调用即为查表操作。

## 3. 程序设计方面的问题

**Q1：观察函数名称可知，很显然，这是开发人员用来调试时所需要的反馈信息。**

信道是独立的，带宽，延时都是不确定的。这时候就需要开发人员利用自己的程序去确定需要的timer。而dbg反馈的就是需要的信息。比如经常在稍微减小datatimer之后信道利用率迅速减少，就知道触碰到ack的最短返回时间了。因为在这种情况下，所有的帧都等不到ack，使得所有的帧都至少传2遍。

**Q2：get\_ms()很好实现**

用static确定程序开始的时间；

每次执行完调用system\_time，相减即可得到运行时间。

C语言的time.h当中提供了一些关于时间操作的函数可以实现get\_ms()函数。 可以利用的函数有clock()函数原型为：clock\_t clock()。该函数返回程序开始 执行后占用的处理器时间，如果无法获得占用时间则返回-1。因为我们计时的起 点并不是程序开始之时，而是开始通信之时，所以需要一个静态变量start\_time 来记录通信起始的时间。然后在每次调用get\_ms()后，获取当前的时间 current\_time。然后再返回start\_time-current\_time即可。

**Q3：start\_timer()和 start\_ack\_timer()，它们都是定时器函数.**

* start\_timer()是启动帧计时，超时重发，每帧一个，不刷新。
* start\_ack\_timer()是启动ack定时器，超时单独发ack（不携带），每个协议一个，可刷新。

## 4. 软件测试方面的问题

* **-u**，无误码信道传输，在该模式下，信道利用率主要与测试协议本身的帧格式有关，反应协议的最大吞吐率。
* 无参数，在该模式下，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，慢发100秒”，可以综合考察协议性能，模仿实际环境协议的运行情况；
* **-fu**， 无误码信道，站点A和站点B的分组层都洪水式产生分组，主要考察协议面对大量数据传输请求时的最大输出能力和抗压能力。
* **-f**, 站点A/B的分组层都洪水式产生分组，在一定误码率的情况下进一步考察协议面对大量数据传输请求时的信道利用率和性能；
* **-f -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。通过显著提高误码率，测试协议面对误码率较高的情况时的信道利用率和性能。如果协议没有设置合理的计时器重传时间或者ACK/NAK发送频次，那么程序性能将在误码率高的情况下有显著下降。

**Q1：对于独立的程序进行反复测试，用不同的信道来查看利用率，实用性如何。**

因为不同的信道要求的的timer是不一样。而sender的动作也会极大部分影响到协议的发挥。所以我们的静态协议需要经得起不同情况的摧残，使得其在任何情况下都不会显得那么难堪。

**Q2：设计这么多种测试方案的目的是什么？**

主要分为以下几个：

* 无误码：测试基本运行能力（基本收发的控制）
* 洪水收发：处理大量数据时网络层的停用和ack的及时回复。
* 间断性：测试健壮性，用不平滑的数据来测试协议的利用率，考察datatimer的合理性。
* 增大误码率：测试误码率波动对协议的影响，考察datatimner和ack的健壮性。

**Q3：你的协议软件存在什么问题时，测试会失败？**

当然，在误码率增大时，Go\_Back\_N会显得很蠢，甚至有时候会直接堵塞。

**Q4：你觉得还 存在有哪些问题是这些测试尚未覆盖的？**

我认为没有覆盖到的地方：两个站基本都不发数据。虽然很无聊但是确实是一种情况。

**Q5：你会采用哪些手段来验证你的程序能正确工作？**

我会用及其离谱的方法来测试：

* 两个极限是疯狂发数据和不发数据（需要带宽的波动）
* 平衡和极端是都发和一个根本不发（对等的波动）
* 误码率的波动

在这三个条件之间寻找合适的测试点，进行测试。

**Q6：本次实验所提供的程序库还有哪些不足？**

我认为还缺少一个自身设置信道带宽，延时，误码率的模块（虽然改代码可以，代码看完了整体也不是非常难看）。这样的话更方便我们对这个实验进行更深层次的理解，或许还可以发现新东西。（反正就到处摧残）

## 5. 对等协议实体之间的流量控制

* 流量控制：针对数据链路层，接收方的反馈。
* 拥塞控制：针对网络层，路由的实时状态反馈。

我认为基本做到流量控制，但也只是能跑的程度，还没有很强的健壮性。

改进方案：监控网络层，进行协议的修改。比如在长时间不发的时候修改为立刻回ack。直到发送数据帧时修改回来。

## 6. 与标准协议的对比

这问题太大了。每个协议有每个协议适用的地方，要是两个站都一直发数据并且误码率不高，我觉得我的的GBN能跑过绝大部分数据链路层协议。但是我的协议就健壮性而言太差了，基本只能完成老师的信道，稍微物理层因为某些原因慢了一点，那就影响大了。

# 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？**

本次实验我们一共编写了四个代码

回退N协议：

Go\_Back\_N（无nak，不携带ack）

Go\_Back\_N++（无nak，携带ack）

选择重传协议：

select.c

Select\_Repeat（在上述协议上改进）

上机调试在18h+，上手5h，编写代码和查看各种参数设置花费较多时间。

由于是线上进行实验，交流起来没有面对面高效，在加上我第一次接触这种实验，刚开始没有准确理解各库函数的作用和参数说明，从而导致花费大量时间反复观看实验说明文档，初步完成实验花费4天时间，然后在对参数的选取，代码的优化上又花费3天的时间，总共花费1周时间完成测试和调试。

**(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。**

在第一次运行函数时，我的VS报了如下错误：



经过查询资料，最终顺利解决

**(3)编程语言方面遇到了哪些问题？包括C语言使用和操控能力上的问题。**

基本没有，就是一开始上手的时候发现函数看不懂，那些参数缺省和类似于源码的代码比较难看。

其中主要体现在protocol.c中的函数不会用，无法理解他的意思，只能根据老师的代码去一步一步推导和尝试。在这个方面我们花费了大量的时间。

**(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。**

* 设计错误：只会简单的信息应答，没有整体架构，导致很简单的低级错误。比如：不做处理的帧判断条件写错了，或写成了与
* 类型转换错误：对于帧格式，错误地进行了二维数组的空间申请（只需要一维的buffer），导致直接超出内存。
* 协议死锁：data\_timer设置过短，不适应窗口大小。
* 不能正确工作：转移条件写错；物理层不能及时关闭；buffer溢出，或者重传的时候给的字段序号不对直接爆炸；滑动窗口的参数修改不正确（有地方错了经常是跑几个就不跑了）

**(5) 开发库方面遇到了哪些问题？**

基本没有，书上的代码思路已经比较清晰了。

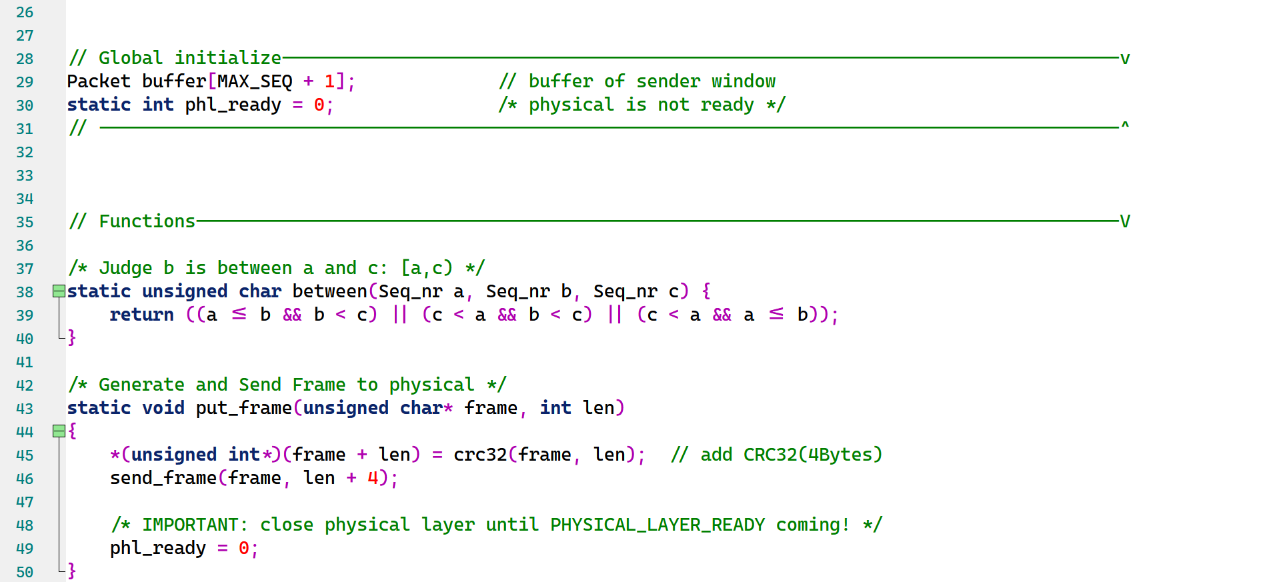
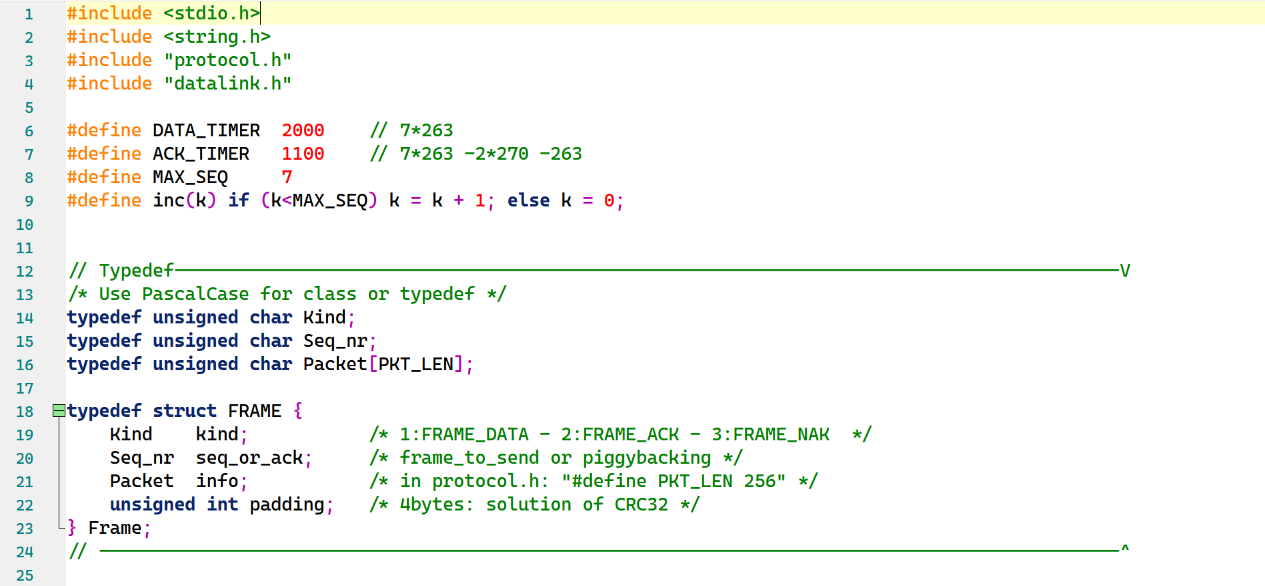
**(6) 心得体会**

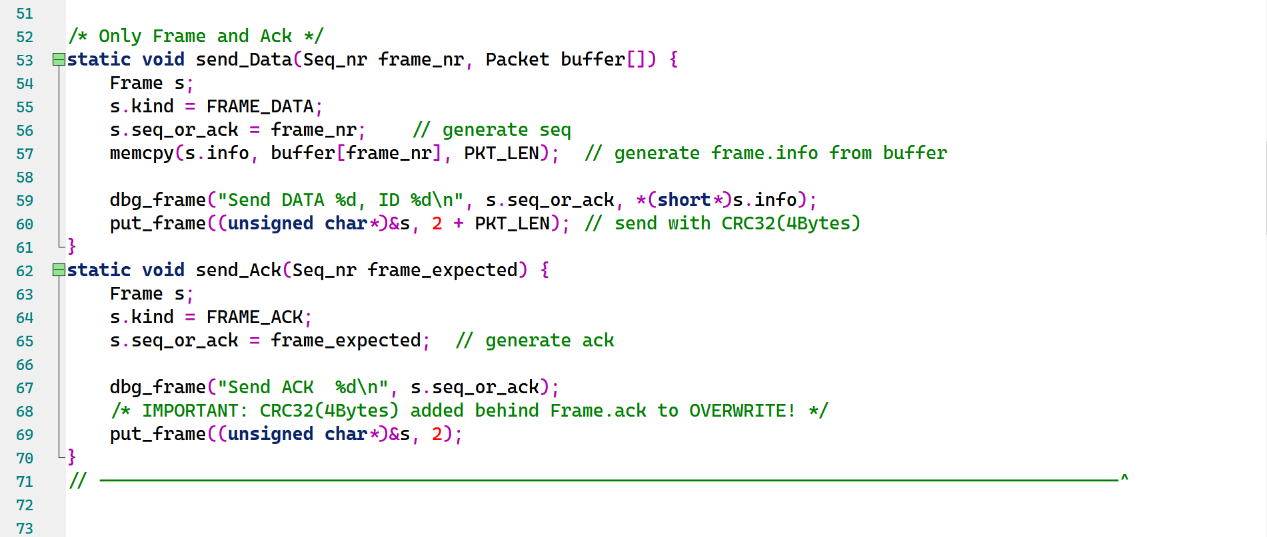
通过这次实验，首先，对完整的数据链路层通信过程有了很深刻的了解，特别是将在课上不是很懂的协议过程，在实验中，慢慢地分析，一步一步地弄明白了。

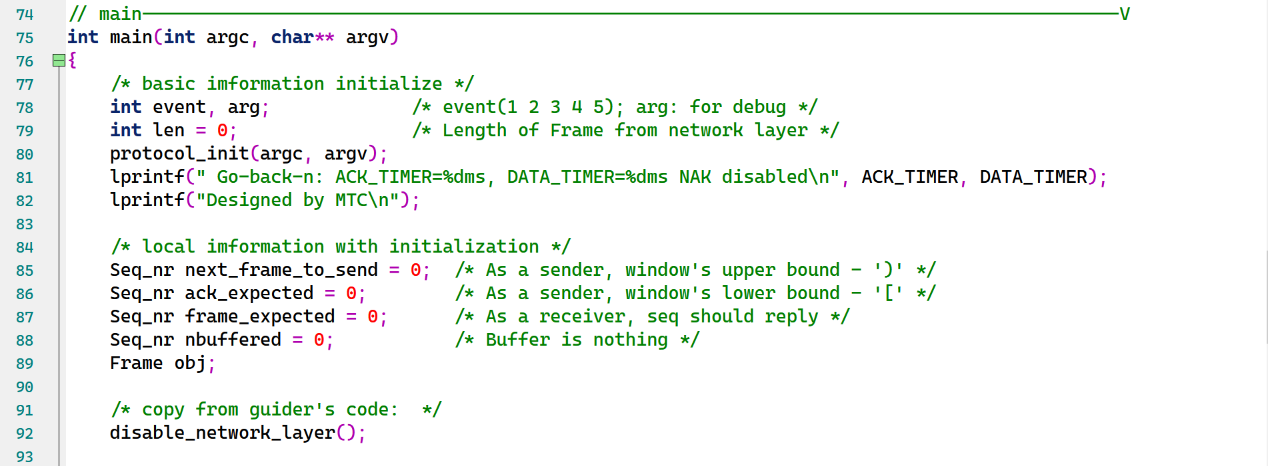
* **对帧格式的深刻理解**：数据帧是正常的，ACK，NAK，以及帧的覆盖问题，都弄得比较透彻。
* **协议对于网络层和物理层的衔接功能实现**：之前只知道拿和放，但是对于时间上的理解并不透彻。这次试验让我们对数据链路层接口的使用有了深层次的理解。
* **对工作代码也有了深刻理解**：虽然在课上大致讲了协议的代码，但概念一直很模糊，通过实验，算是对代码有了详细的分析，不管是对各个变量，还是函数中的各个参数，都能够清楚知道它所代表的含义，所起的作用。包括对通信命令的使用。其核心就是：状态的应答和转换。
* **锻炼了编代码的能力与加强了对小组合作的认识**，在小组内部，分工明确，代码接口清晰，为以后打下基础。

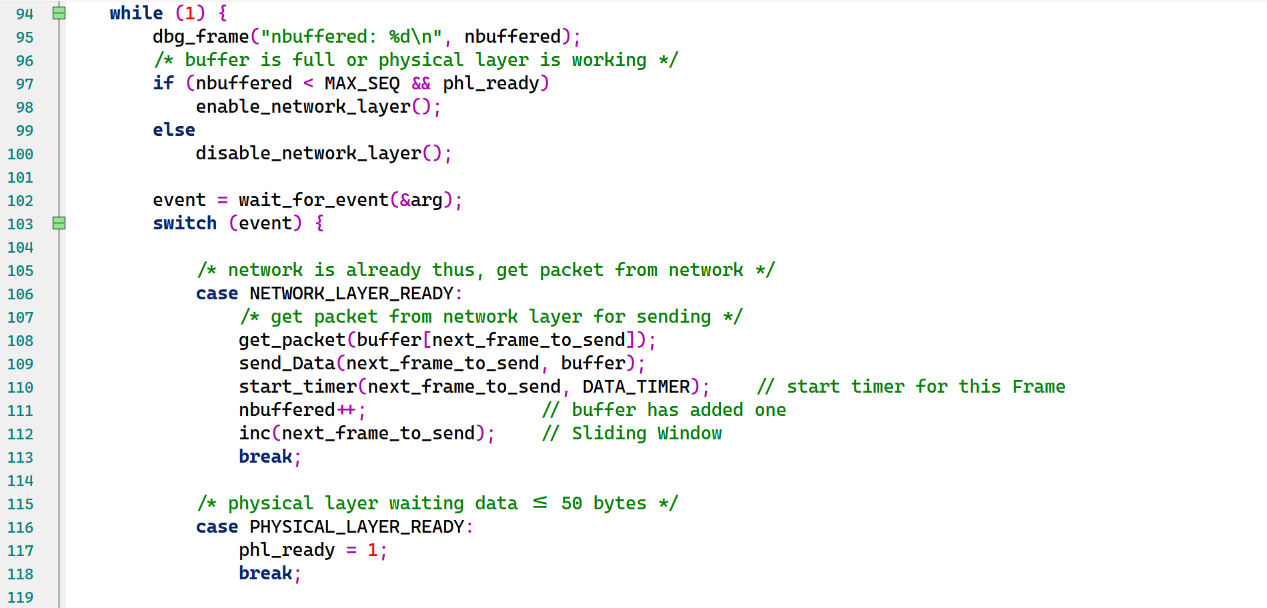
# 附录：源代码（只展示datalink.c）

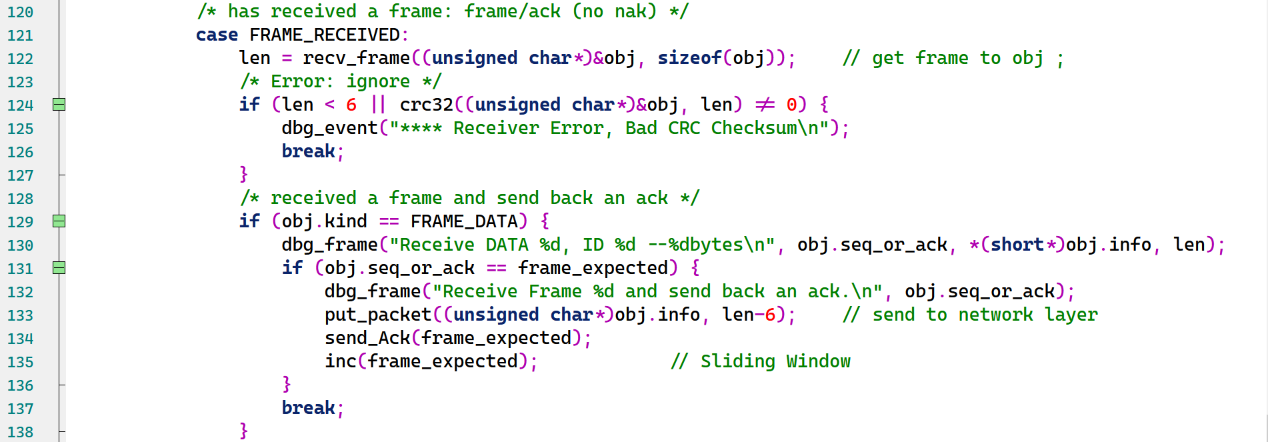
## Go\_Back\_N（不携带Ack）

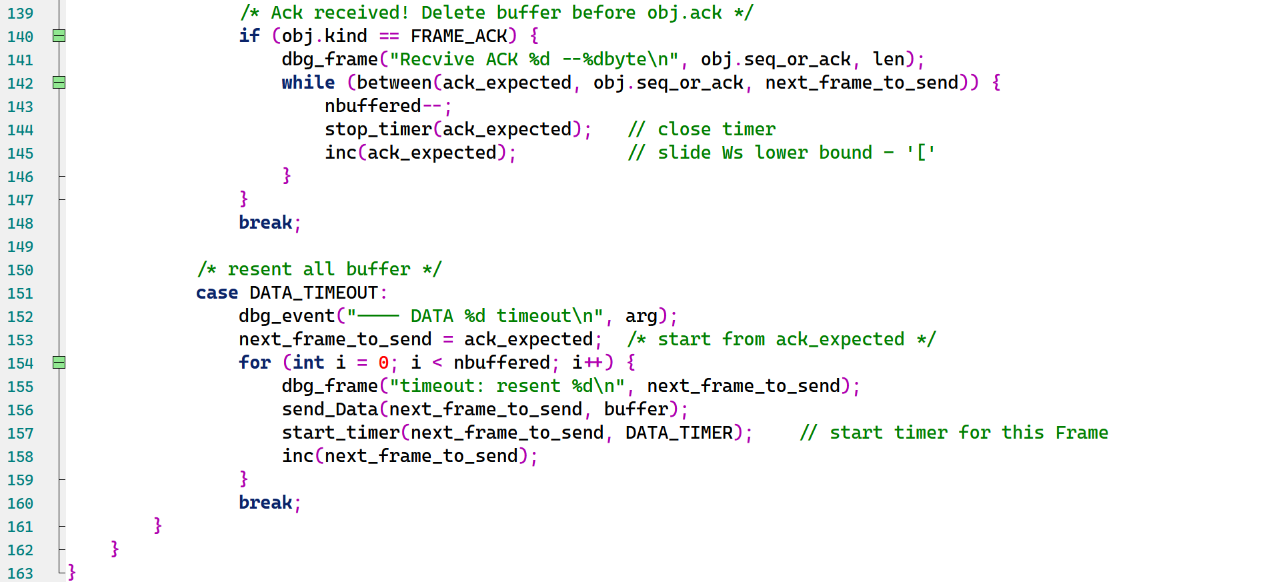




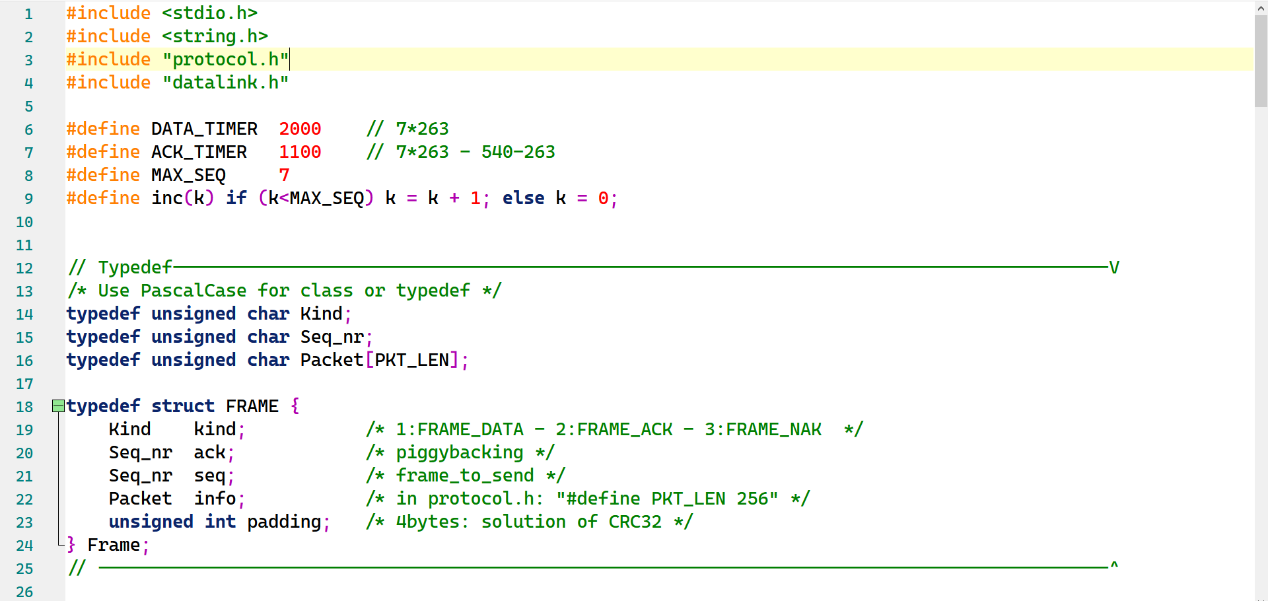


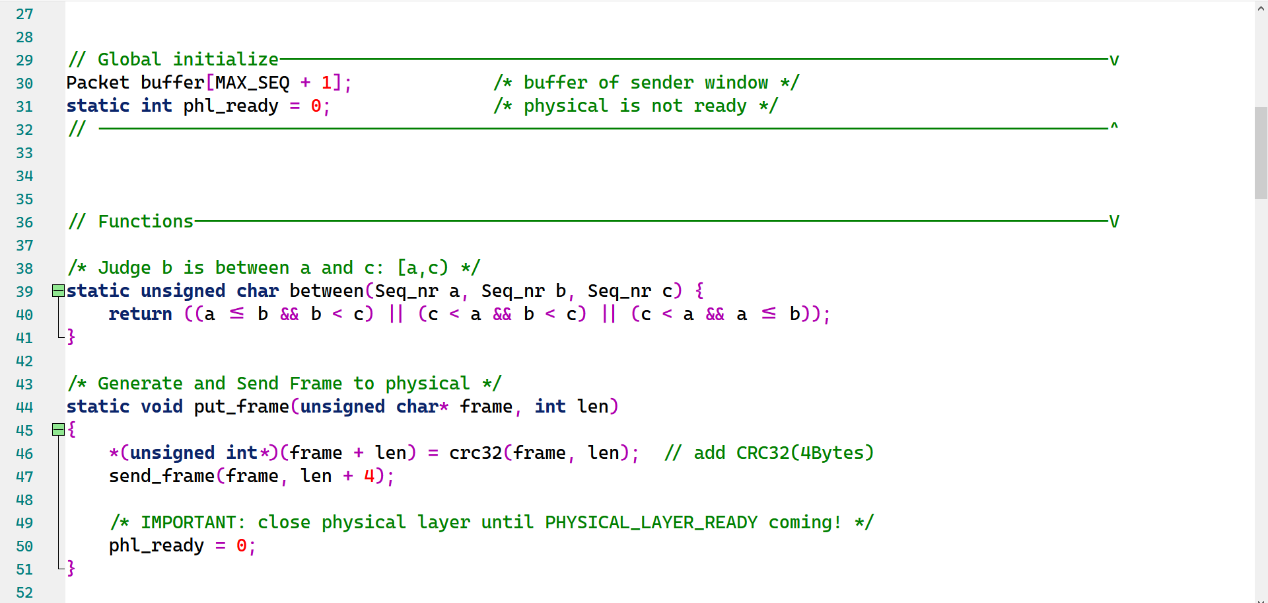


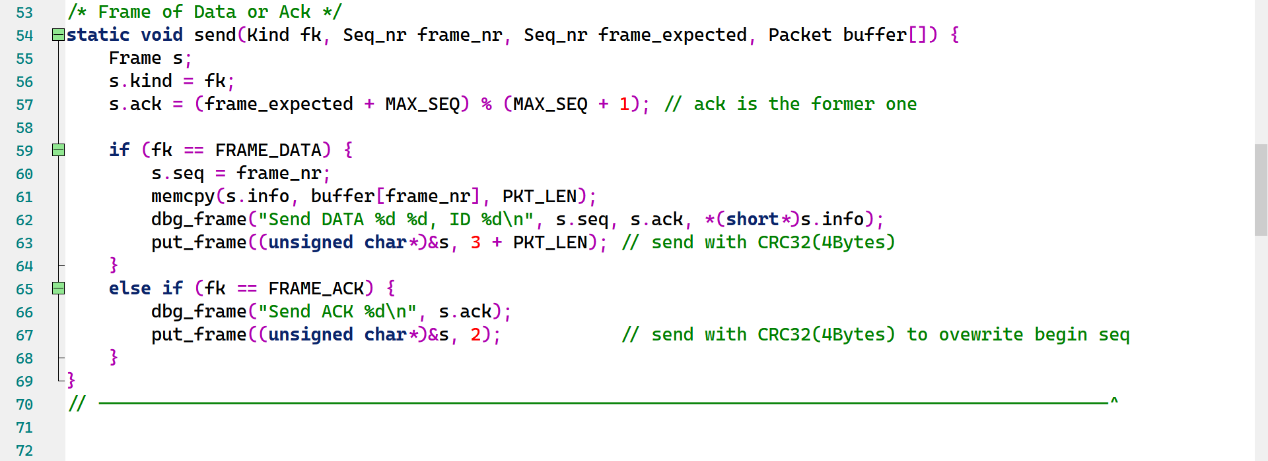


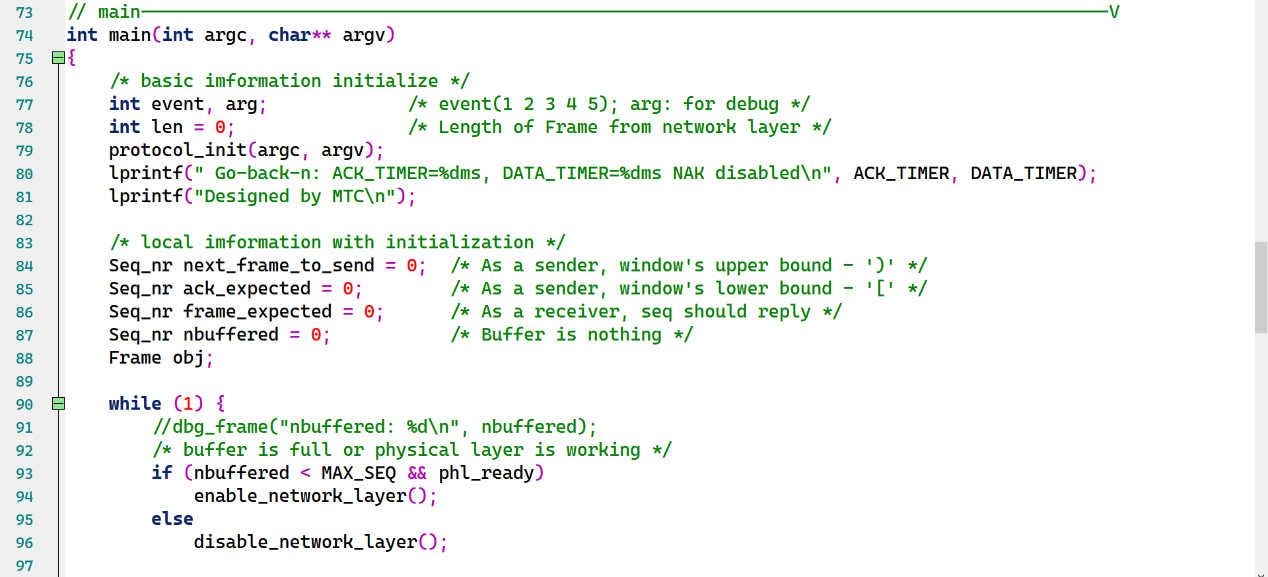


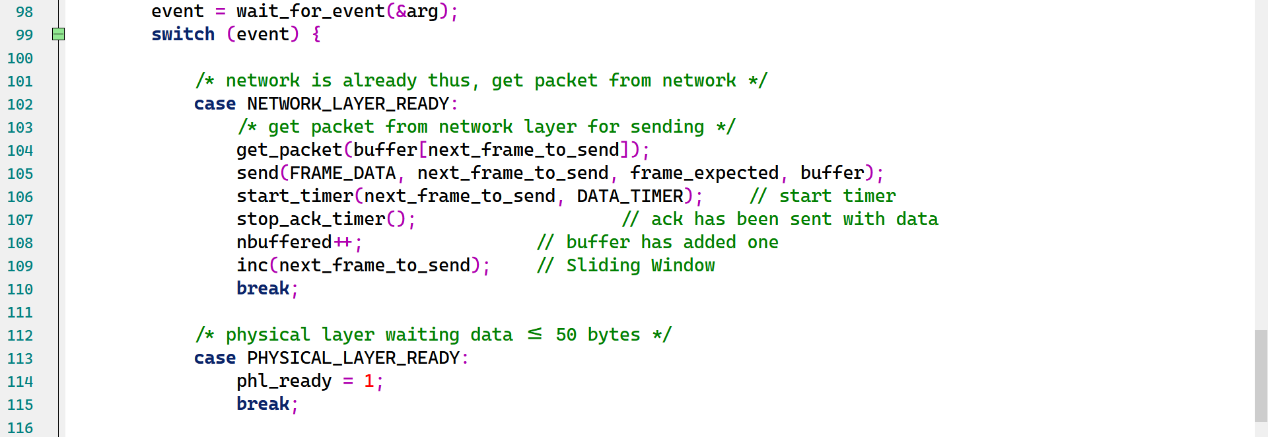
## Go\_Back\_N++（携带Ack）



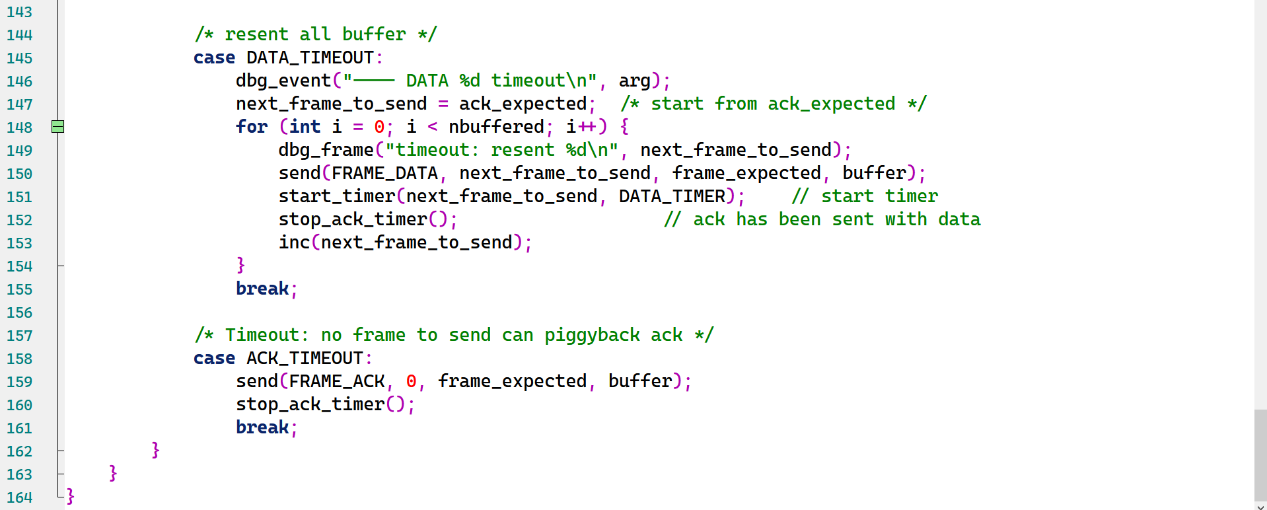












## Select\_Repeat（有Nak）



