**滑动窗口协议实验报告**

**学号 202034071005 姓名 苏建誉**

**目录**

[1. 实验目的及要求 2](#_Toc59371125)

[2．实验环境 2](#_Toc59371126)

[3. 实验内容与步骤 2](#_Toc59371127)

[3.1 帧的结构体格式 2](#_Toc59371128)

[3.2 滑动窗口的结构体格式 3](#_Toc59371129)

[3.3 发送者的结构体格式 4](#_Toc59371130)

[3.4 接收者的结构体格式 4](#_Toc59371131)

[3.5 crc16函数实现 4](#_Toc59371132)

[3.6 接收者：handle\_incoming\_msgs函数实现 5](#_Toc59371133)

[3.7 发送者：handle\_incoming\_acks函数实现 6](#_Toc59371134)

[3.8 发送者：handle\_input\_cmds函数实现 7](#_Toc59371135)

[3.9 发送者：handle\_timedout\_frames函数实现 8](#_Toc59371136)

[4. 实验结果与数据处理 9](#_Toc59371137)

[4.1 样例1-样例4 9](#_Toc59371138)

[4.2 新增样例5：256个发送者与256个接收者短帧通信 9](#_Toc59371139)

[4.3 新增样例6：256个发送者与256个接收者长帧通信 10](#_Toc59371140)

[4.4 数据处理 11](#_Toc59371141)

[5. 分析与讨论 13](#_Toc59371142)

[5.1 逻辑分析 13](#_Toc59371143)

[5.2 讨论 14](#_Toc59371144)

# 1. 实验目的及要求

1. 目的：通过代码实现数据链路层的滑动窗口协议SWP，熟悉掌握滑动窗口协议，提高编程的水平。

2. 要求：实现数据链路层的滑动窗口协议**SWP**，在信道有噪声的情况下，发送者和接收者之间实现可靠的通信，实现**多对多**的通信，实现**crc16**循环冗余检验的功能，自定义**帧格式**，实现**选择重传协议**，实现**超时重发功能**，实现**分组切片功能**。

# 2．实验环境

1. 操作系统：Ubuntu18.04

2. 编译器：C++/C linux ×86

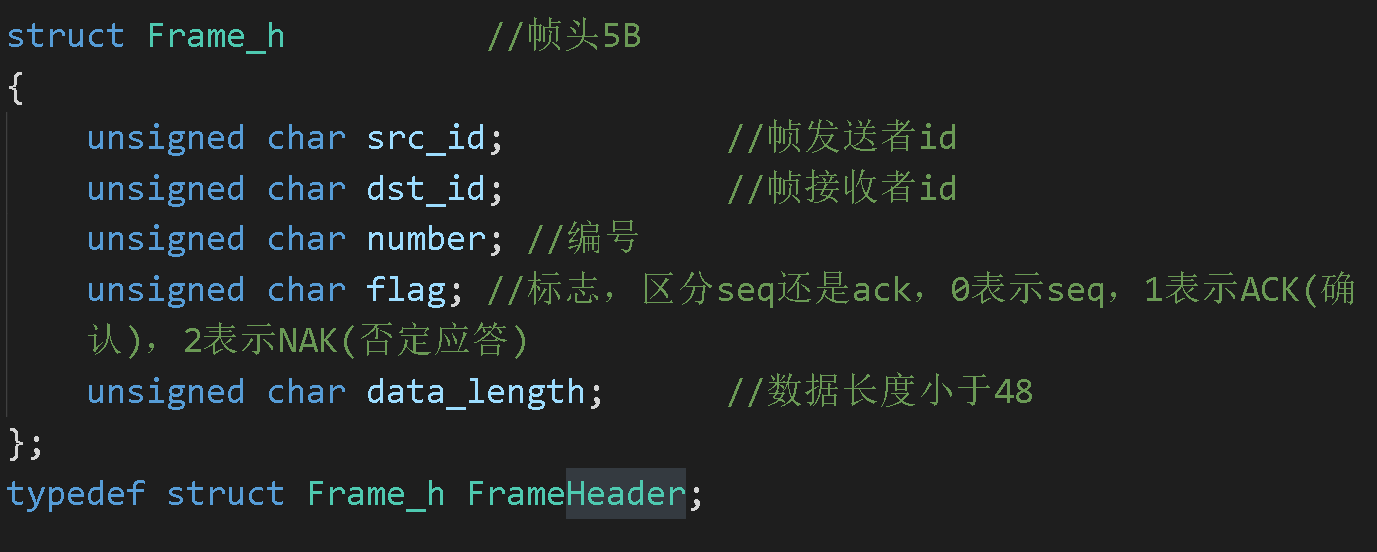
3. 远程连接工具：Remote-WSL

4. 代码编辑器：Visual Studio Code

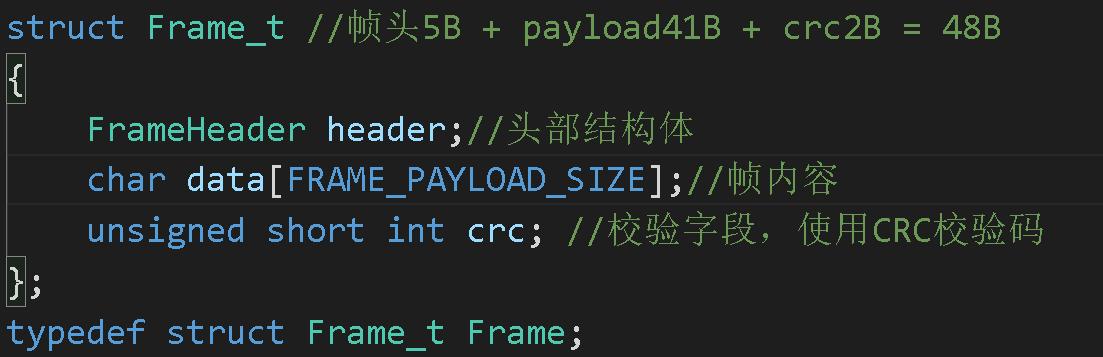
# 3. 实验内容与步骤

## 3.1 帧的结构体格式

3.1.1帧头：



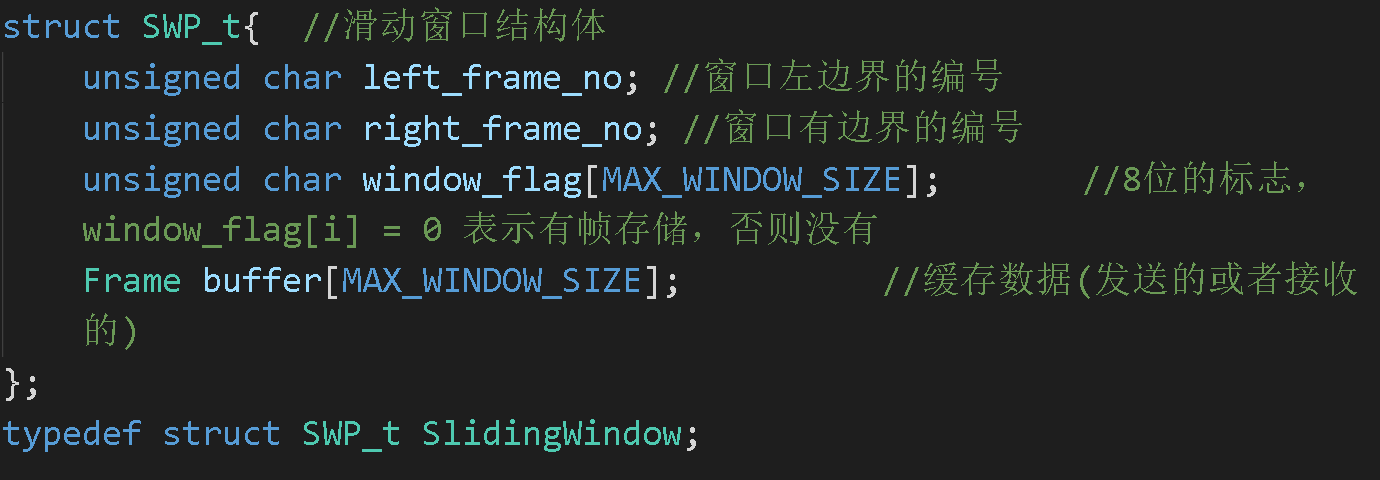
3.1.2 帧



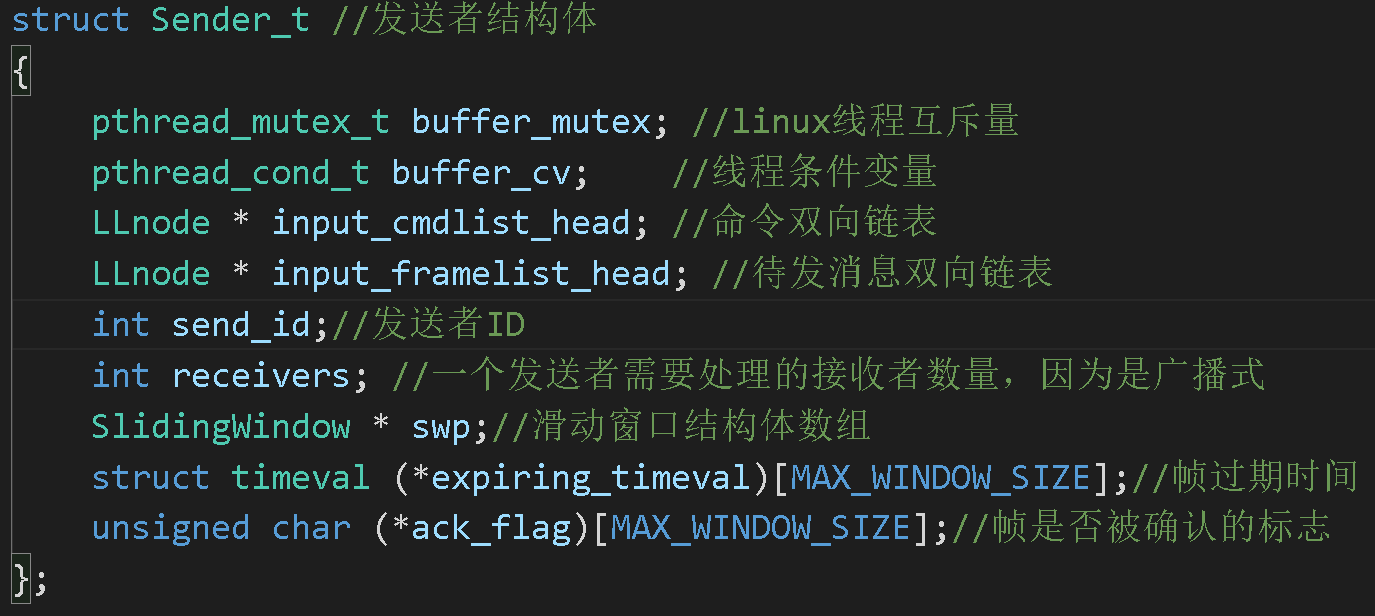
3.1.3 帧可视化：



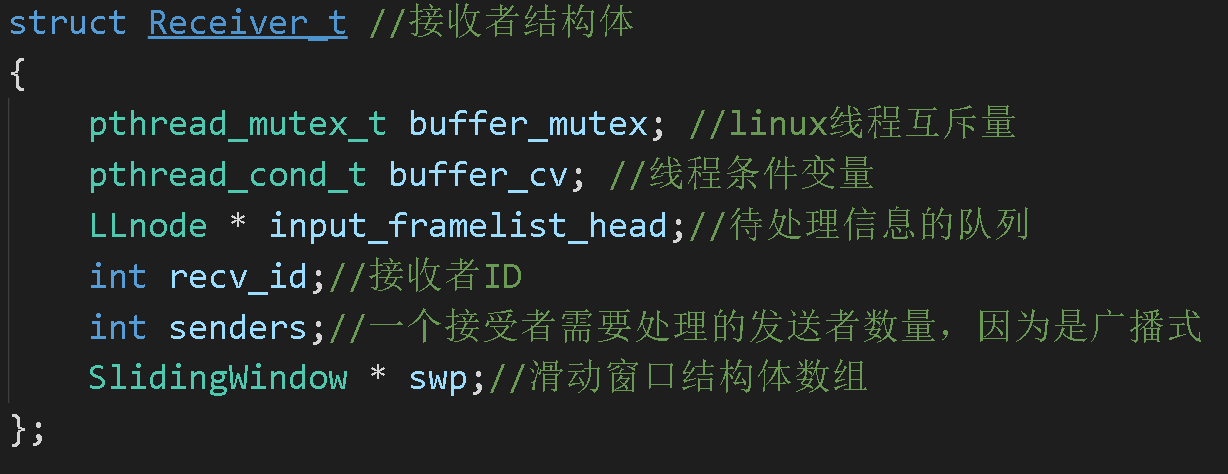
## 3.2 滑动窗口的结构体格式



## 3.3 发送者的结构体格式



## 3.4 接收者的结构体格式



## 3.5 crc16函数实现

3.5.1 步骤：

1）初始化crc = 0，获取字符串当前字节ptr

2）取crc高8位da

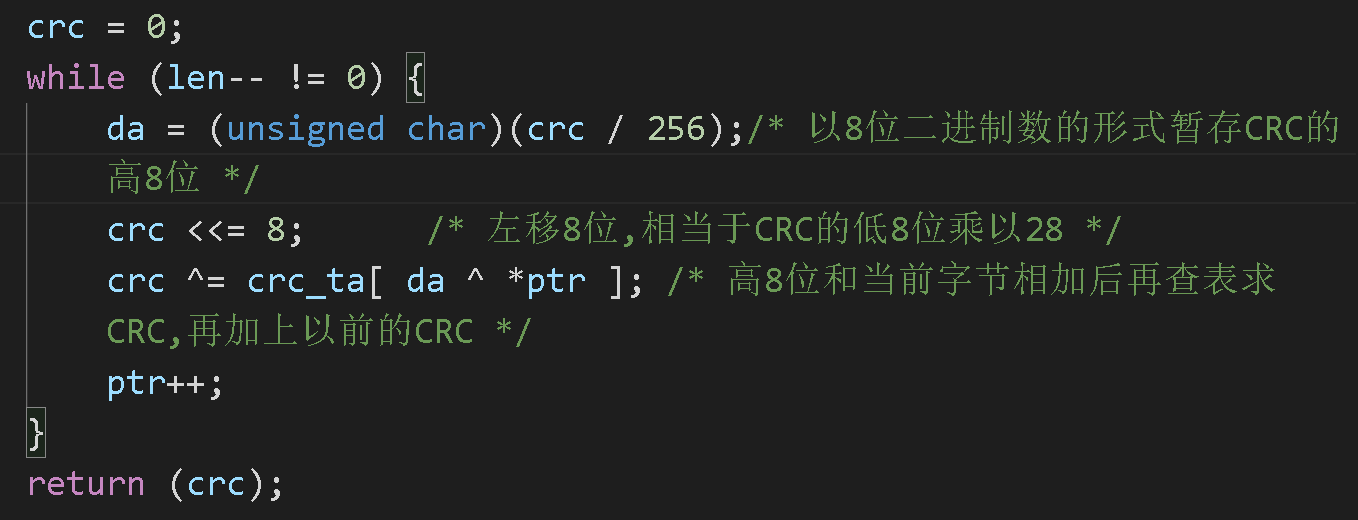
3）crc左移8位

4）da与ptr做异或操作，然后查表得到16位的tmp

5）crc与tmp做异或操作

6）若ptr为‘\0’,返回crc；否则，则&ptr++后返回到第一步

3.5.2 代码截图



## 3.6 接收者：handle\_incoming\_msgs函数实现

3.6.1步骤：

1）将接收者缓冲区的队头节点组成帧

2）利用crc16函数检测帧，如果帧损坏&&发送方ID和接收方ID合法&&帧是属于该接收方的，则向发送者发送NAK帧，跳到步骤7

3）如果帧不是该接收者的，跳到步骤7

4）如果帧落入接收者的窗口内，设置window\_flag标志位，把帧缓存并且向发送者发送ACK帧

5）如果帧不在接收者的窗口内，直接向发送者发送ACK帧

6）如果接收方窗口一部分帧按序到达，则滑动窗口，滑动window\_flag标志位，滑动缓存帧。

7）进入下一次循环

3.6.2 代码截图：



## 3.7 发送者：handle\_incoming\_acks函数实现

3.7.1 步骤

1）将发送者消息缓冲区的队头节点组成回复帧

2）利用crc16函数检测回复帧，如果回复帧损坏则跳到步骤6

3）如果帧不是该发送者的，跳到步骤6

4）如果回复帧落入发送者的窗口内

4.1）如果是ACK回复帧则把对应的window\_flag置为1

4.2）如果是NAK回复帧则把对应的已发送的缓冲帧的计数器重置，跳到步骤6

5）如果发送窗口的帧按序被ACK确认了，则滑动窗口，滑动window\_flag标志位，滑动缓冲帧，滑动计数器。

6）进入下一轮循环

3.7.2 代码截图



## 3.8 发送者：handle\_input\_cmds函数实现

3.8.1 步骤

1）将发送者命令缓冲区的队头节点组成命令

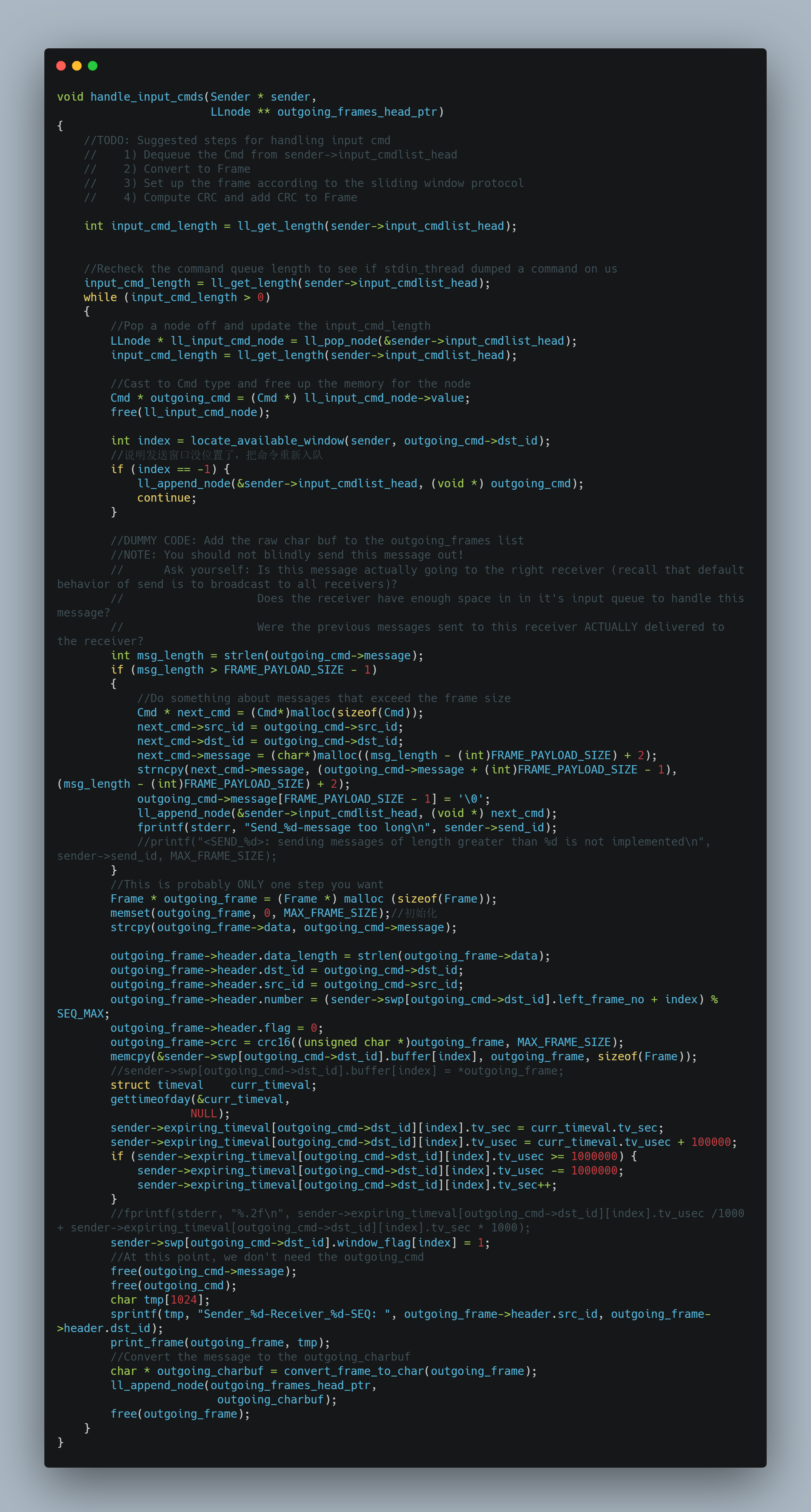
2）获取发送者可缓存的窗口位置index

3）如果idex == -1则发送者没有可用的窗口，重新把命令插入到队列之中，进入到下一轮的循环。

4）如果命令的消息长度大于帧的负载，把消息分段，切出FRAME\_PAYLOAD\_SIZE大小的消息，并且把剩余的消息重新插入到队列之中

5）把消息组成帧，加入CRC校验字段，设置帧的类型为SEQ，在index的位置缓存帧，设置window\_flag标志并且设置超时时间，向接收者发送帧。

3.8.2 代码截图



## 3.9 发送者：handle\_timedout\_frames函数实现

3.9.1 步骤

1）遍历发送者对应的所有接收者的每个帧

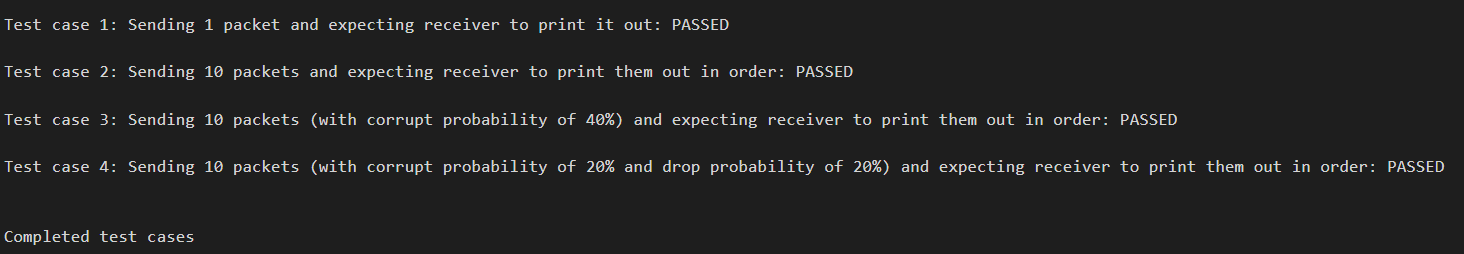
2）如果发现有帧的过期时间早于现在的时间，则该帧已经过期，从缓存里面提取帧重新发送给接收者，然后重新设置过期的时间。

3.9.2 代码截图



# 4. 实验结果与数据处理

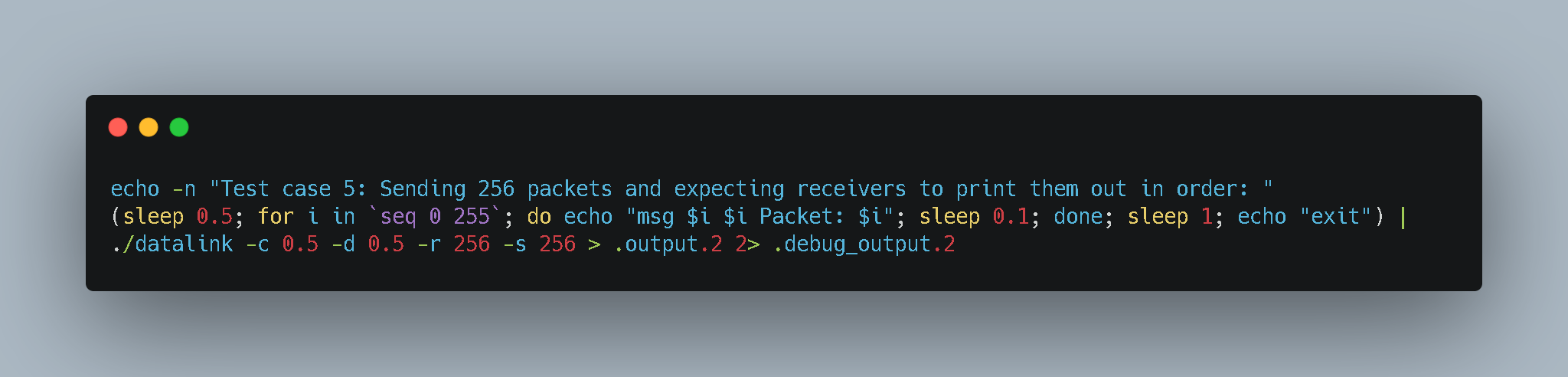
## 4.1 样例1-样例4



程序的4个标准样例全部PASSED通过了。

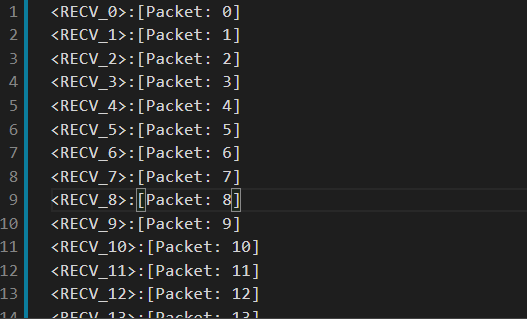
## 4.2 新增样例5：256个发送者与256个接收者短帧通信

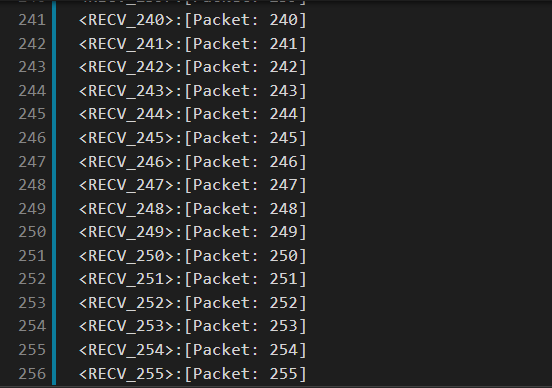
4.2.1样例介绍：



$i发送者发送Packet: $i的帧给$i接收者，并且在50%丢包率和50%误码率的情况下通信。

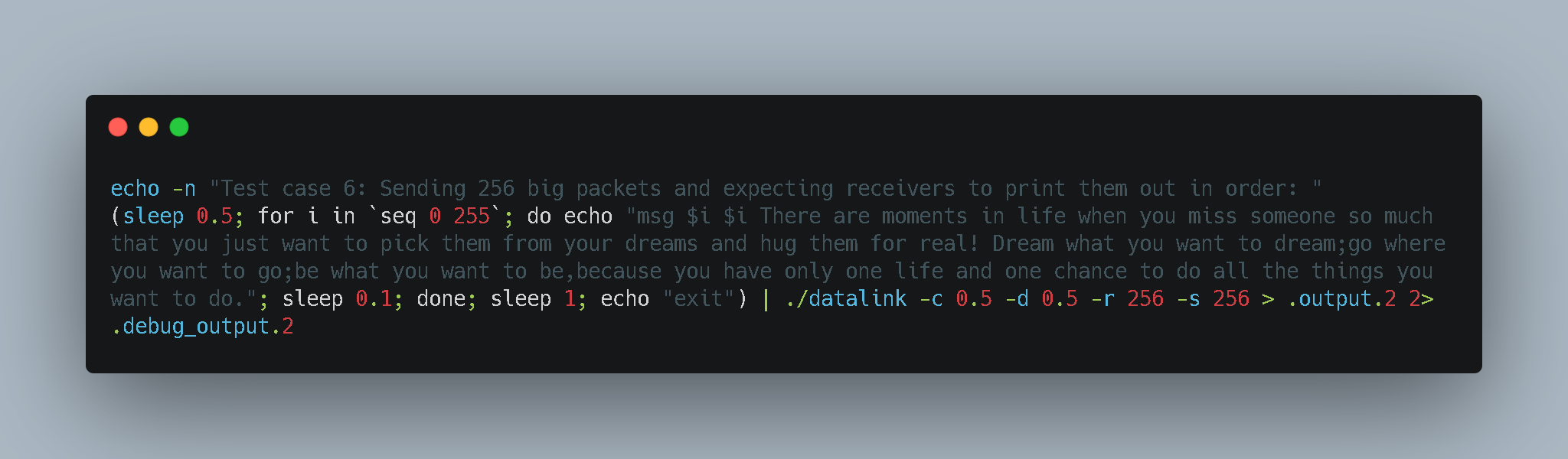
4.2.2实验截图：



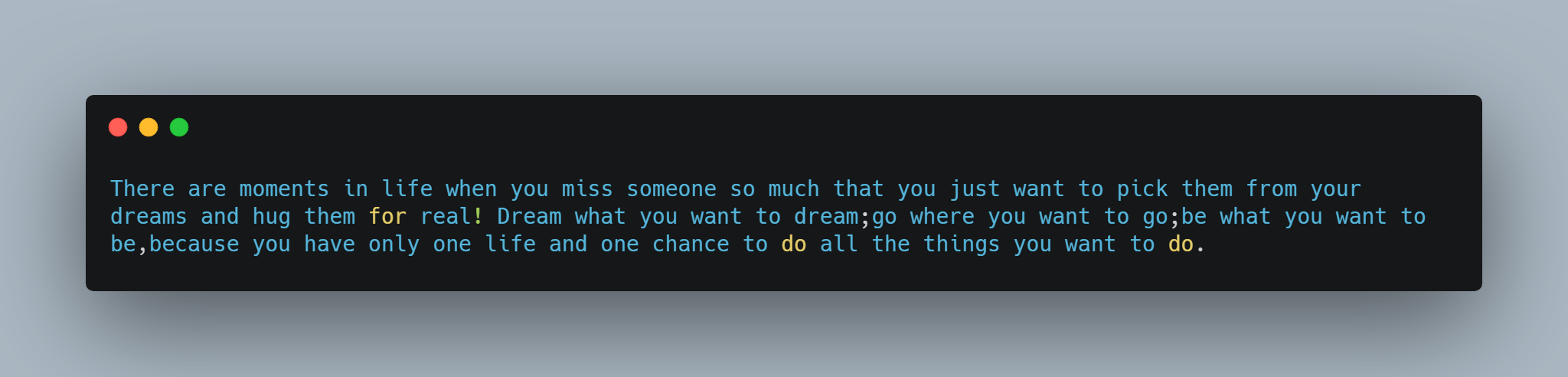


## 4.3 新增样例6：256个发送者与256个接收者长帧通信

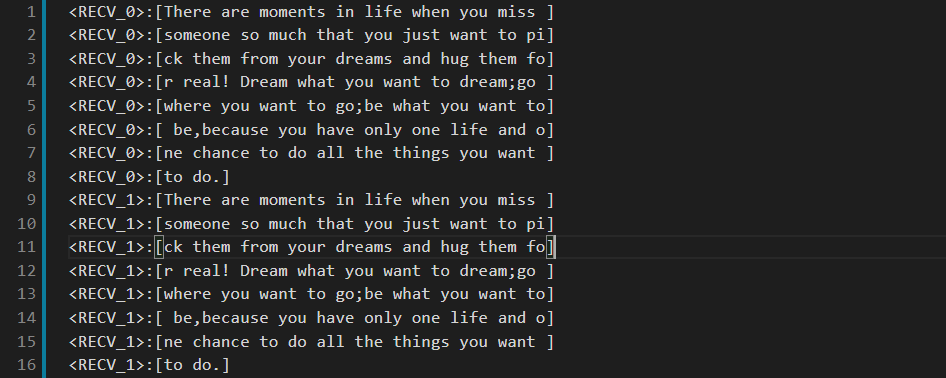
4.3.1 样例介绍：

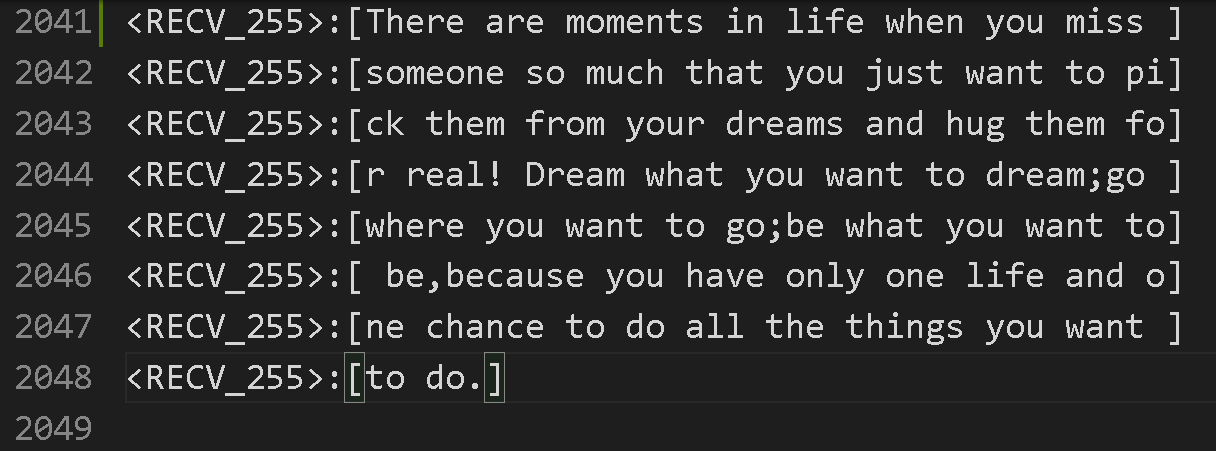


$i发送者发送Packet: $i的帧给$i接收者，并且在50%丢包率和50%误码率的情况下通信，帧的内容如下：



4.3.2 实验截图

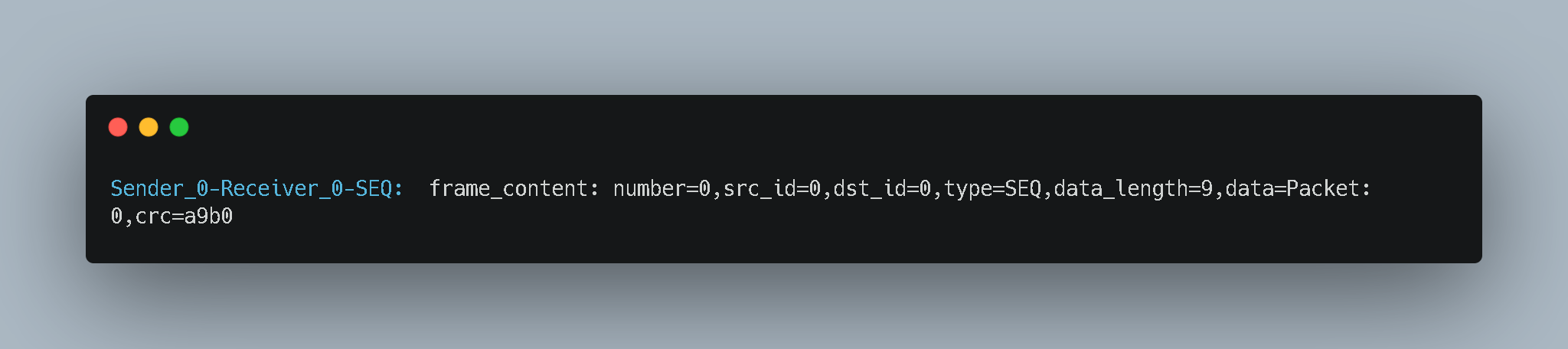




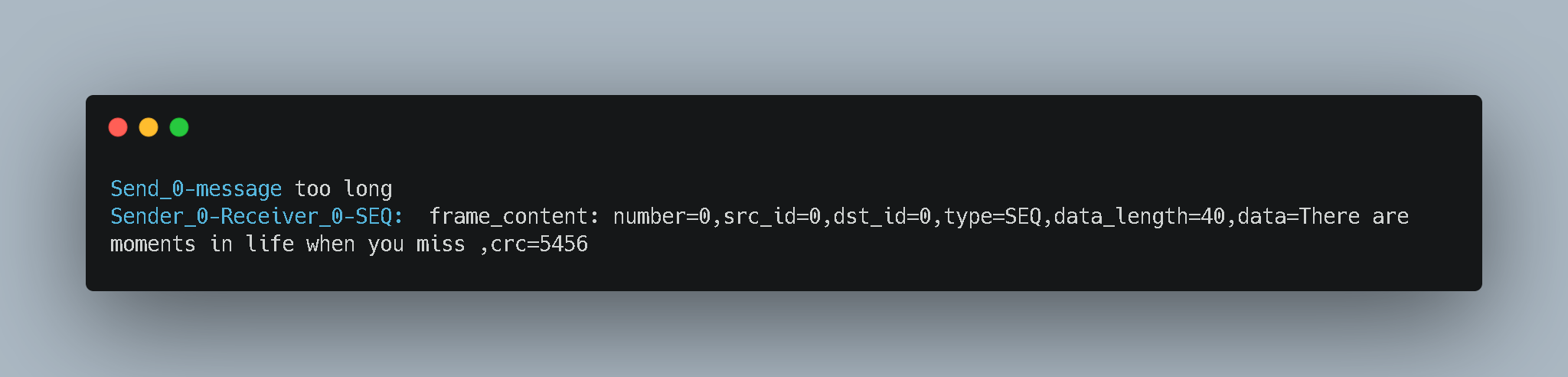
## 4.4 数据处理

4.4.1 发送帧

1）当帧为短帧时候

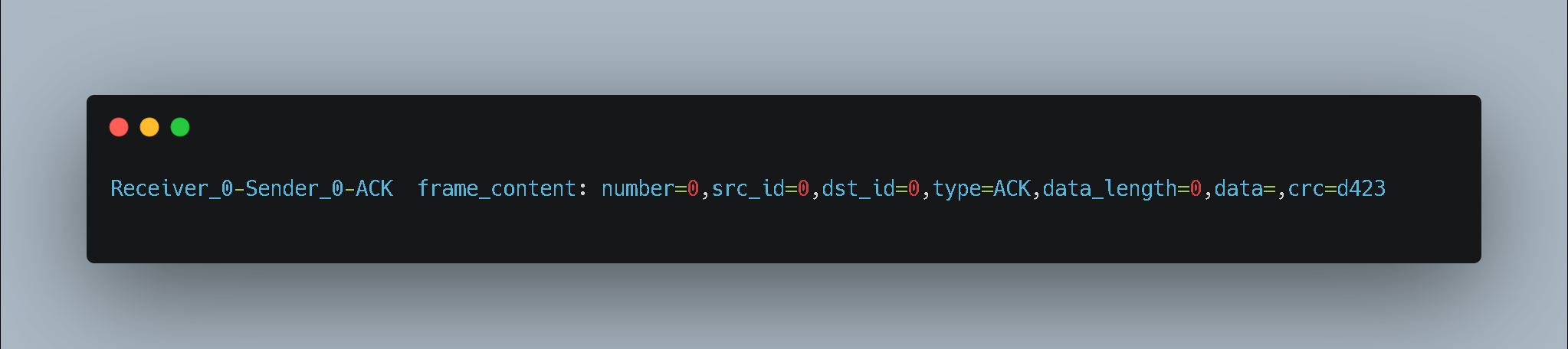


2）当帧为长帧的时候

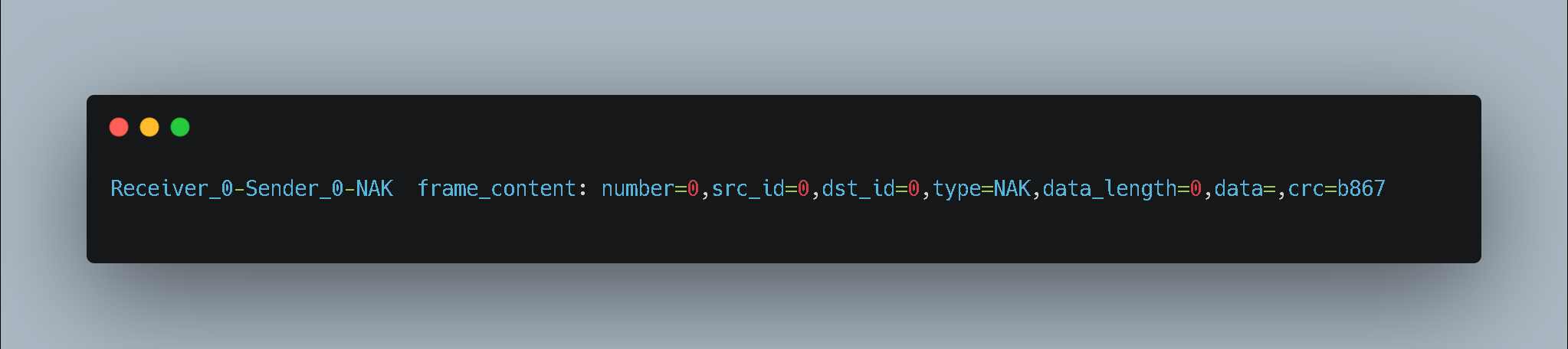


4.4.2 回复帧

1）ACK回复帧



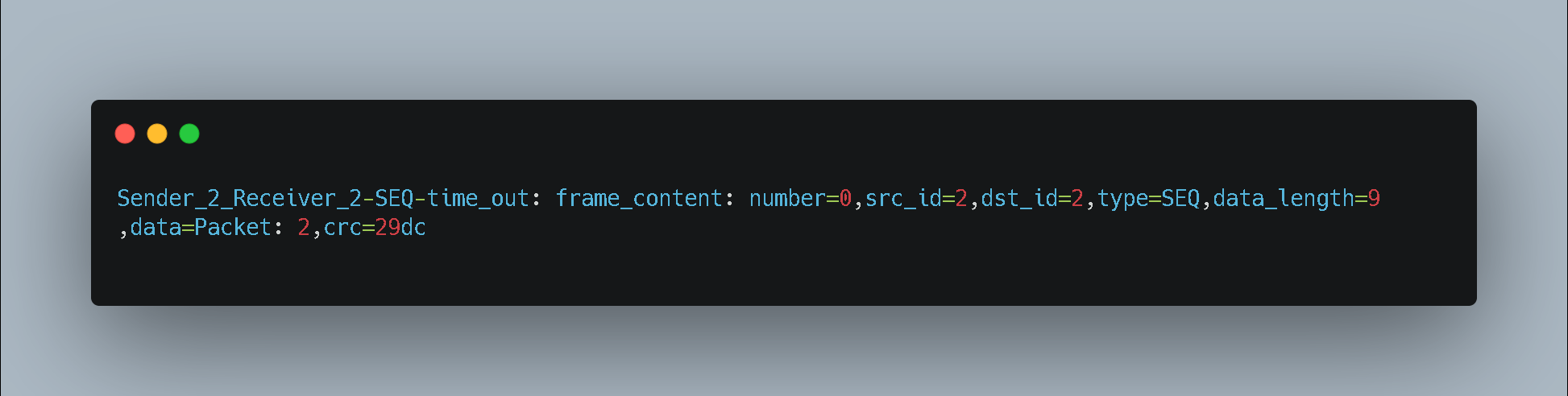
2）NAK回复帧



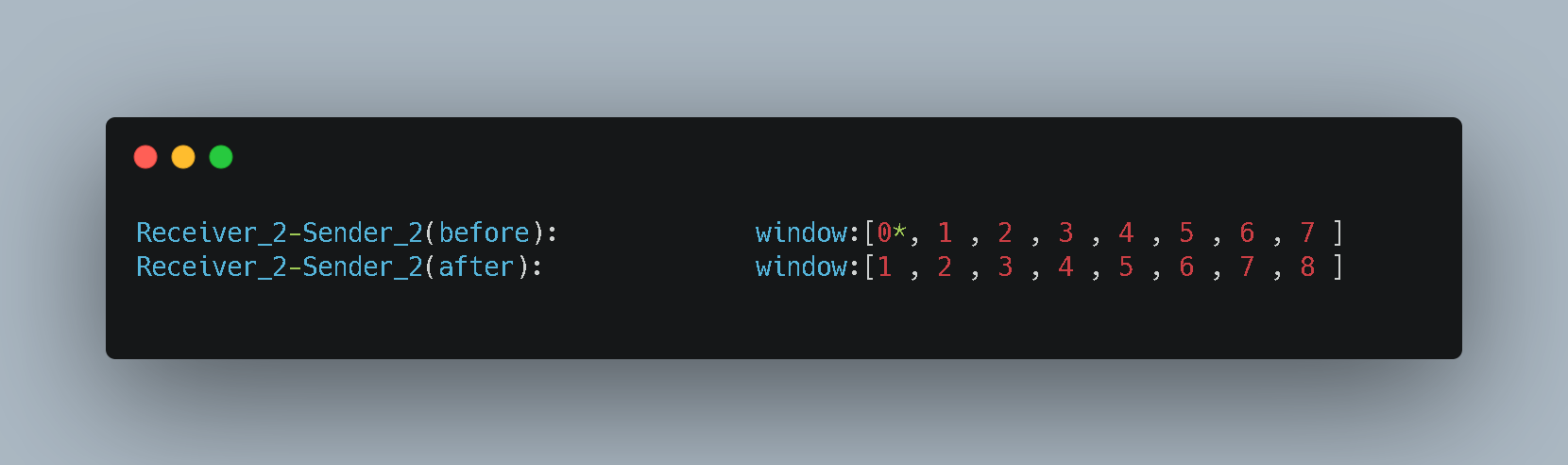
4.4.3 crc16差错检测



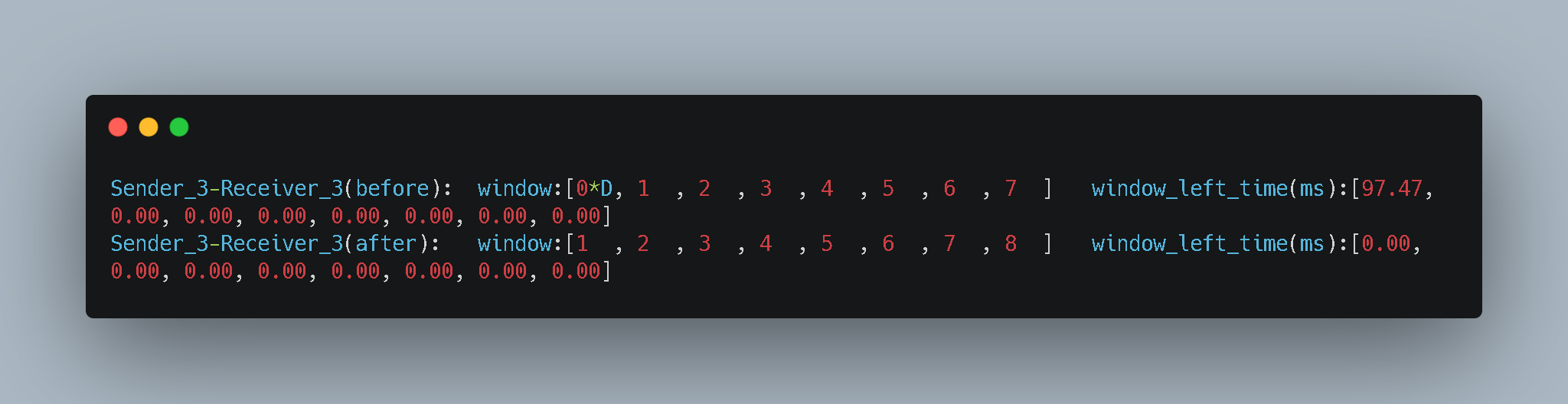
4.4.4 发送者超时重传



4.4.5 接收者窗口滑动（\*代表该缓冲位置已经有帧）



4.4.6 发送者窗口滑动（\*代表该帧已发送，D代表该帧已收到ACK



# 5. 分析与讨论

## 5.1 逻辑分析

5.1.1 可靠通信

可靠通信要保证在有噪声的信道，接收者能够按序把接收到的帧提交到网络层。实现可靠通信需要滑动窗口、选择重传协议、帧序号等等。

5.1.2 多对多通信

多对多需要每个结构体维护一个数组，该数组包括所有发送者/接收者的状态，例如：滑动窗口数组、帧过期时间数组、ACK确认数组

5.1.3 回复帧

为了实现可靠通信，需要对接收的帧进行回复，因为本实验采用选择重传协议，回复帧有ACK帧和NAK帧，当帧没有发生损坏的时候回复ACK帧，否则，发送NAK帧，让发送者立刻重发该帧。

5.1.4 超时重传

如果发送的帧或者ACK回复帧都丢失或者损坏，发送者必须重传，所以对于每个已发送的帧设置定时器。

5.1.5 滑动窗口

滑动窗口协议最重要需要定义滑动窗口的变量，例如：滑动窗口的边界，帧标志位以及帧缓存数组。

## 5.2 讨论

1）**本次作业的本质是理解他人框架并且完善既定的业务逻辑。**这与公司的工作类似，需要读懂他人代码，并且确定实现的逻辑（见第报告第三部分），然后才是写代码，最后是测试（见报告第四部分），只有这流程通过了，才能下班。

2）**代码的逻辑分析十分重要，这关乎作业完成的效率和质量。**因此我花了一天的时间，用思维导图，把整个代码框架写下来，然后确定问题的逻辑分析。

3）**不同的需求环境，不同的实现。**滑动窗口协议，其实有很多变种，对于ACK回复帧，它可以是对一个，也可以对多个帧确认，也可以是对期望的下一个帧确认；当收到损坏帧的时候，可以选择发送NAK帧，也可以选择丢弃，等到超时重传。每种方案都有优点和缺点，都需要根据客户的需求来确定。