### 实验结果分析

1、实验结果简介：

此次实验，本小组采用了无nak以及捎带确认的Go-Back-N协议以及有nak和捎带确认的Go-Back-N协议以及选择重传协议，能够较长时间地在有误码信道环境中无差错传输，同时，在传输过程中，如果出现错误，无nak的协议会丢弃错误帧，使发送方计时器超时重发，有nak的协议会发送nak给发送方，发送方收到相应nak后就会重发相应帧，因此能够可靠地实现通信。

1. 协议参数的选取：

一个数据帧共有256 + 3 + 4 = 263字节，因此发送时延为263 \* 8 / 8000 = 263ms，传播时延为270ms，则α = 270 / 263 = 1.027，对不带捎带确认的协议U = 1 / (1 + 2 \*α) = 0.327，发送窗口为4即可达到最大；对捎带确认的协议，U = 1 / (2 + 2 \*α) =0.247,发送窗口数为5即可使利用率最大，在实验中我们选取MAX\_SEQ为15，而NR\_BUFS = ((MAX\_SEQ + 1) / 2) = 8。

对ack\_timer和data\_timer的选取，ack\_timer作用是当检测到发送方没有发送数据帧无法捎带确认时主动发送ack，而一个ack算上校验和只有6字节，因此比传播时延稍长即可，设为276ms。data\_timer由于我们采用累计ack确认的方式，所有首先保证窗口内帧都发完，其中帧与帧之间有1ms的间隔，然后为了提高效率，我们并没有选择等待窗口最后一个帧收到后的ACK\_TIMER超时发送ack直到收到的时间为止才超时，而是在更早一些的地方超时，因此其值为263 \* 窗口大小 + ACK\_TIMER +窗口大小 - 1

1. 理论分析：

由于需要携带帧讯息，因此最大的信息利用率为约为256/263≈97.3%。由于信道的最大比特率为8000bps，可得出每传输一个字节耗时1ms；每帧的附加信息固定为10，耗时10ms；若出现转义字符，则可能增加时间。现在假设信道上始终有数据需要传送，这样就可以简化模型。在有10-5的错误率的信道上，算下来在这100000个比特中可以传送100000/（263\*8）≈48个数据包，则可得出每48个数据包会有一个出错，假设每出错一次，在限定时间内可以重传该帧使其成为正确帧。则每传送48个数据包需要传送48+1=49次，此时信道利用率≈95.3%，而实际上由于程序设计的原因，当一个数据包超时后，往往要重复多次传输给数据包造成信道的浪费，若有k次重传，则信道利用率为（256/263）\*（48/（48+1+k））。

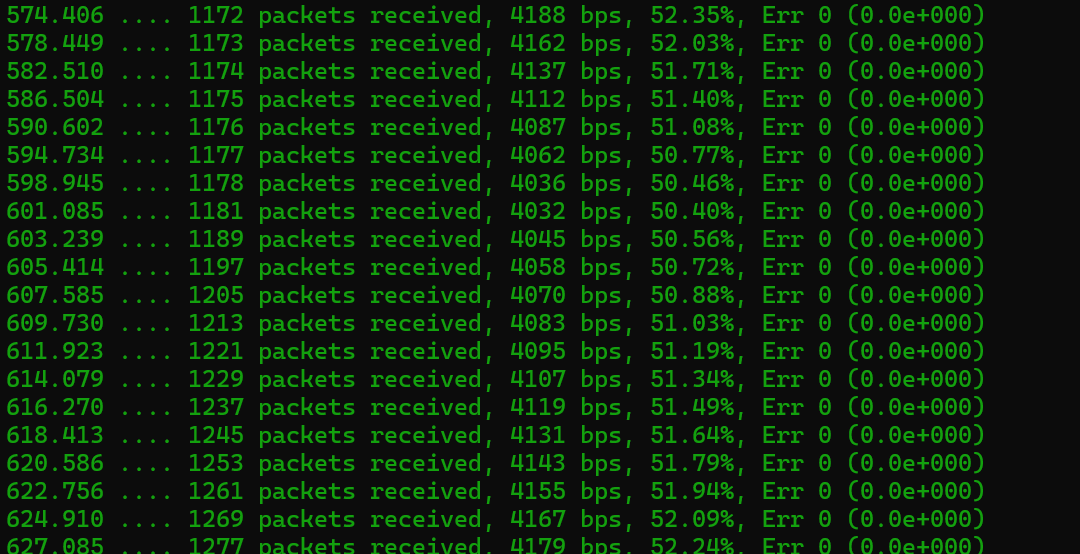
若为回退N协议，出错后需要重传后面所有窗口，则利用率会更低。

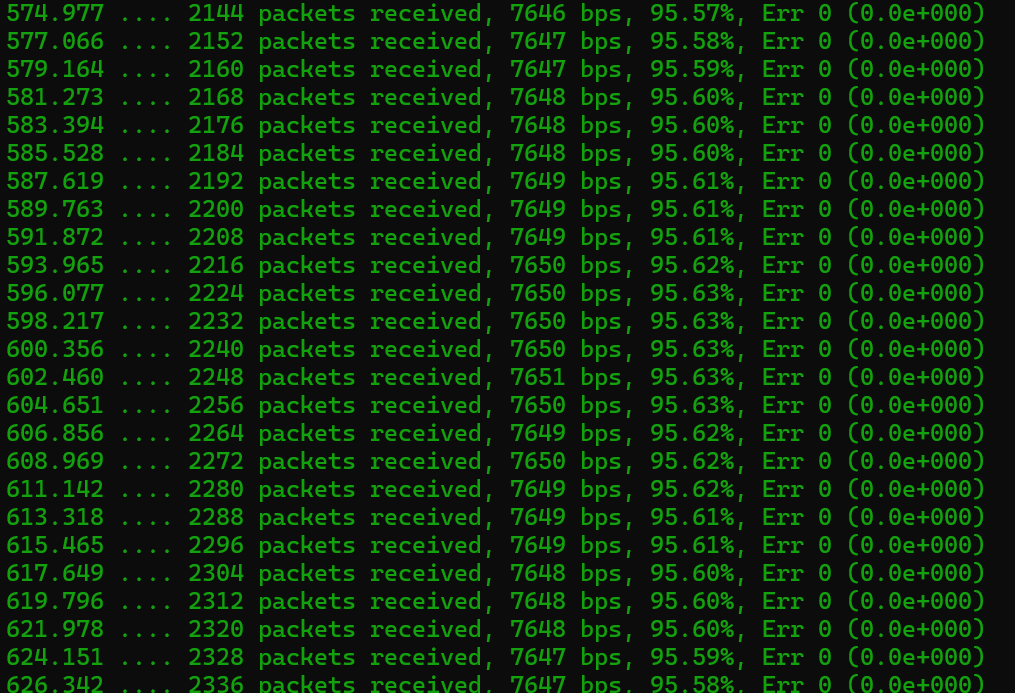
4、实验测试：

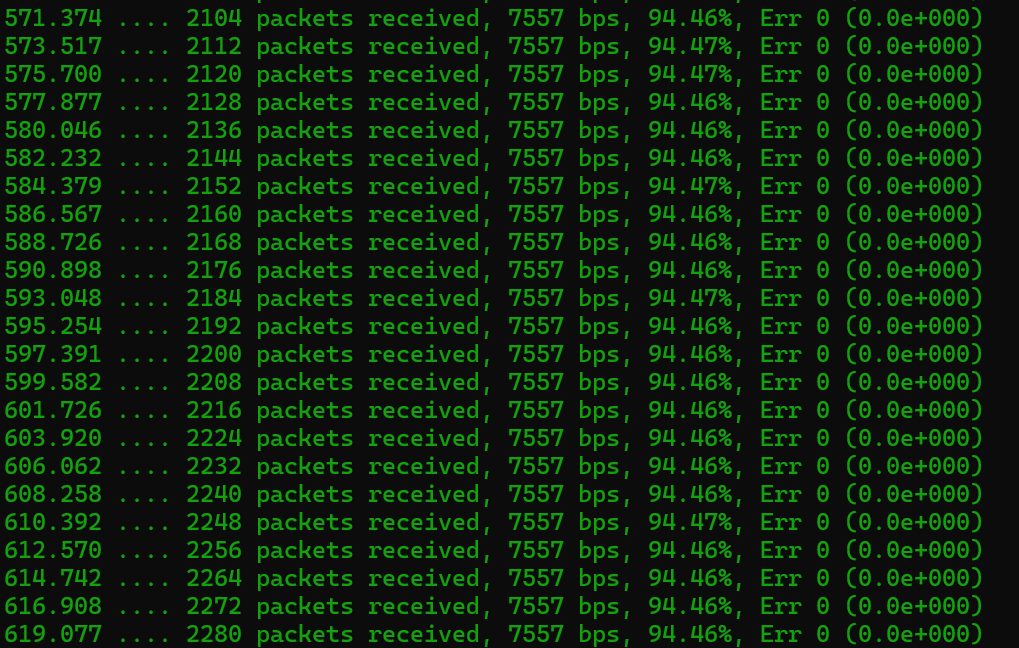
#### 性能测试记录表

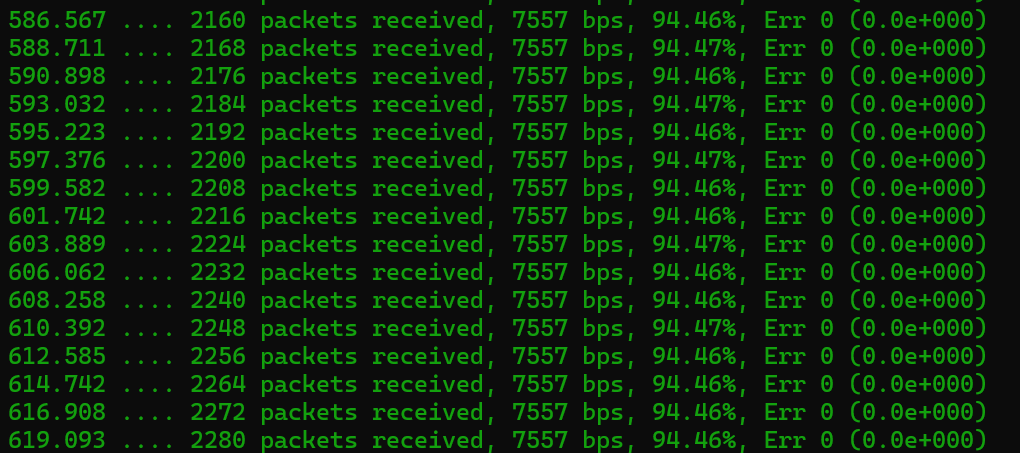
|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 序号 | 命令选项 | 说明 | 运行时间(秒） | GoBackN算法（无nak）  线路利用率(%) | | Selective算法  线路利用率(%) | | GoBackN算法  线路利用率(%) | |
| A | B | A | B | A | B |
| 1 | -–utopia | 无误码信道数据传输 | 626 |  |  | 55.70 | 96.96 | 95.58 | 52.24 |
| 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” | 626 |  |  | 53.96 | 90.51 | 43.70 | 75.87 |
| 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 | 619 |  |  | 96.97 | 96.97 | 94.46 | 94.46 |
| 4 | –-flood | 站点A/B的分组层都洪水式产生分组 | 605 |  |  | 93.03 | 92.64 | 77.65 | 74.06 |
| 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 | 621 |  |  | 56.94 | 57.38 | 26.50 | 26.80 |

运行时间要求超过10分钟。









**三 研究探索**

1. CRC 校验能力

CRC校验码具有以下检错能力：

检查出全部单个错；检查出全部离散的二位错；检查出全部奇数个错；检查出全部长度小于或等于K位的突发错。

1. 由于本次试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，使得误码率波动比较大的，例如，下雨天和晴天，高噪声和低噪声的情况，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。

3.对等协议实体之间的流量控制

在我们设计的协议当中，流量的控制主要通过接收窗口，发送窗口还有确认机制来实现。因为有窗口大小的限制，发送方不会一次性发送过多信息导致接收方被信息洪流所淹没，导致信息丢失。这样可能会导致信道的利用率降低，但是如果合理的设计窗口大小，依然可以达到较高的信号利用率。

4.start\_timer是直接覆盖的，而start\_ack\_timer则是当ack\_timer为0时，才能让ack\_timer开始计时。

**四 实验总结**

实际上机调试6小时

无

编译需要加-lws2\_32

（1）对具体ACK\_TIMER以及DATA\_TIMER的设置

在初步设置TIMER之后，尝试对时间进行调整，比较对效率的影响。ACK\_TIMER一定程度增大及缩小，对于无差错的情况来说，基本会产生不到1%的效率下降。而DATA\_TIMER一定程度缩小，效率变化夜基本在1%之内，一定程度增大，效率基本不产生变化，持续增大会导致效率降低。据此判断，初步设置的TIMER数据基本合理。

1. 对于SR协议，刚开始运行时，发现在两边会稳定在序列号中间数产生bad packet错误，经过增加大量debug语句以及对传输过程的分析，最终发现超时重复，连续重发相同帧的问题，然后注意到在设置计时器时对帧序号进行了模运算，导致了不同的帧超时使得重发同一个帧，且重发只对低序号有效。
2. Gdb调试不方便，由于发送数据和时间相关，但调试只能让程序暂停执行。

开发库

收获：

C 语言方面：

在编写协议的过程中，我们使用了结构体来定义数据帧，并使用缓冲数组来管理数据帧的发送和接收，实验中也涉及到了大量关于数据帧的操作，加深了我们对指针和结构体的了解，同时除我们所写协议外的其他函数代码都已由老师提供，这也增强了我们进行项目工程合作，理解函数接口的用法。

协议软件设计与实现以及理论学习：

在本次实验中，我们认识到了：

1、在具有nak的回退N协议和选择重传协议中，nak的作用不是如我们预期的通知发送方重发CRC校验错误的出错帧，而是通知发送方发送接收方未收到的预期帧。

2、稍待确认是接收方每一次发数据帧都会携带的，确认序号为接收方期待帧的前一帧，并不是接收方需要发ack时才会发送。

3、GobackN协议发送窗口大小为，但暂存的buffer数组大小应该为，这样buffer数组多出的空间就可以避免窗口滑动时，发送窗口数据使用同一个buffer空间暂存。

我们深刻理解了GobackN协议和SR协议的工作原理，包括窗口滑动、帧的确认与重传、计时器的设置等，特别是在处理有噪音信道环境下的误码检测和纠正方面，通过 CRC 校验技术，有效地确保了数据传输的可靠性，纠正了我们曾经对这两个协议的一些错误理解，我们对协议进行了一系列的调试和优化，通过分析信道利用率，合理配置滑动窗口的大小和重传定时器时限，最大限度地利用了传输信道的带宽，从而提高了协议的性能。

软件工程实践：

模块化编程：在实现协议的过程中，我们采用了模块化编程的方法，将协议的各个功能模块（如数据帧的发送与接收、窗口管理等）独立实现，提高了代码的可读性和可维护性。

调试与测试：我们通过灵活使用调试模式，更改输出的颜色，系统地进行协议软件的调试，找出程序中的存在的问题和不足并进行了修正。这一过程提升了我们在软件调试和测试方面的能力。

性能分析与优化：在实现基本功能后，我们对程序的性能进行了分析，通过优化参数配置（如滑动窗口大小、重传定时器时限等），显著提高了信道利用率。这一过程使我们对软件性能优化有了更深入的理解。

通过本次实验，我们不仅巩固了所学的计算机网络理论知识，还在实际编程和软件工程实践中得到了全面的锻炼。这些经验和技能的积累，将为我们今后在计算机网络和软件开发领域的学习和工作打下坚实的基础。