基于 UDP 或 Raw Socket 实现一种可靠的传输层协议

一、实验概述

众所周知，传输控制协议（TCP，Transmission Control Protocol）是一种面向连接、面向字节流、可靠的传输层协议。与之相对地，UDP 和 Raw Socket 不是可靠的传输协议，它们无法确保数据包是否到达或者是否按序到达。他们直接将上层的报文封装成该层的 PDU，并且转发给下层。

本实验希望揭露 TCP 进行可靠传输的秘密。参与实验者需要设计并实现一种传输层协议，该协议具有类似 TCP 中可靠传输的部分功能，即确保接收的数据包没有差错、确保数据包能被接收方所接收、确认接收的数据包的先后顺序。为了保证实验效果，实验应该基于 UDP 或 Raw Socket 等没有可靠传输功能的网络协议。

二、实验要求

参与实验者需要自行设计协议的数据包格式和数据包的收发控制机制，确保其能满足如下功能：

- 连接是**单工**的，即有发送方和接收方之分，接收方除了发送必要的控制报文外不应该发送数据报文。发送的数据是面向字节流的（而非面向字符流）。
- 由于接收方并不能发送数据报文，因此不存在需要等待接收方发送数据的半关闭状态，连接双方使用两次握手建立连接（即 SYN、ACK）和断开连接（FIN、ACK）。
- 也**可以按照** TCP 的三次握手建立连接和四次握手断开连接进行实现，这并不会改变该实验的实现难度。
- 每一个数据包中的载荷（Payload，即需要传输的上层数据）的大小应该不超过**最大分段大小**（MSS，Maximum Segment Size）。最大分段大小可以被上层程序所配置，实验中需要观察设置不同最大分段大小时的传输效果。
- 数据传输按照固定窗口的方式进行，只有当窗口里面每一个数据包都确认传达，才会发送下一个窗口的数据包。数据包每一个窗口的大小应该不超过**最大窗口大小**（MWS，Maximum Window Size），它应该为最大分段大小的倍数。最大窗口大小为最大分段大小的倍数可以为上层程序所配置，实验中需要观察设置不同最大窗口大小时的传输效果）。
- 对于窗口里面的每一个数据包，为了降低实现难度，采用逐包确认的方式，而非 TCP 中对于连续收到的数据包只确认最后一个可用数据包的方式。

- 在复杂网络环境的条件下，参考 TCP 协议的实现，设计合适的收发控制机制，使得设计的协议能在复杂网络环境下工作。复杂的网络环境举例如下：数据包丢失、数据包乱序或超时到达、数据包内容有差错、确认包丢失、确认包乱序或超时到达。

为了方便在单机环境下模拟复杂网络环境，因此参与实验者需要在发送方和接收方进行人为的丢包和发送延迟/乱序。可配置的参数及要求如下：

- 丢包（Loss）：可配置的参数有丢包率（pDrop）和用于确定是否需要丢包的随机数种子（seedDrop）。
- 延时/乱序（Delay/Disorder）：可配置的参数有发送数据包的时间的最大延迟（maxDelay），延迟率（pDelay）和用于确定延迟多少的随机数种子（seedDelay）。
- 数据包如果发生丢包，则不计算延迟；**如果不发生丢包**，一个数据包可以准备就绪到实际发送的时间为
  `(pDelay <= 0.0)? 0 : max(0, random(seedDelay) - 1.0 + pDelay) / pDelay * maxDelay`
  可能出现一个数据包先准备就绪，但是由于计算得出所需的实际发送时间比较后准备就绪的大，出现乱序的情况。
- 每当发送数据包时，应该记录当前数据包是被发送出去了、延迟了（当发送时间超过当前设定的 timeout 时）还是被丢弃了，并且把日志信息输出到控制台标准错误输出（stderr），参看第三部分提交内容的说明。

参与实验者需要使用其设计的传输层协议传输任意文件到接收方，文件可能是文本文件，也可能是图片等二进制文件。

发送方程序命名为 sender，发送方的参数列表如下：

- <file>：需要发送的文件路径。
- <ip>：接收方程序的 IP 地址。
- <port>：接收方程序的端口。
- <pDrop>：发送方丢包的概率，为一个小于 1 的浮点数。
- <seedDrop>：发送方丢包的随机数种子，为一个整数。
- <maxDelay>：发送方发送数据包的最大延时。
- <pDelay>：发送方发生延迟的延迟率，为一个小于 1 的浮点数。
- <seedDelay>：发送方发生延迟的概率，为一个整数。
- <MSS>：发送方的最大分段大小。
- <MWS/MSS>：发送方最大窗口大小为最大帧大小的倍数。
- <initalTimeout>：初始的超时，为一个整数，其更新算法可参考相关文档。

接收方程序明明为 receiver，接收方的参数列表如下：

- <file>：将接收的文件保存到的路径。
- <ip>：接收方程序需要绑定到的 IP 地址。
- <port>：接收方程序需要绑定到的端口。
- <pDrop>：发送方丢包的概率，为一个小于 1 的浮点数。
- <seedDrop>：发送方丢包的随机数种子，为一个整数。
- <maxDelay>：发送方发送数据包的最大延时。
- <pDelay>：发送方发生延迟的延迟率，为一个小于 1 的浮点数。
- <seedDelay>：发送方发生延迟的概率，为一个整数。

譬如，发送 file.txt 到接收方（本地的 6666 端口）并且存盘为 receive.txt，其中有 20%的几率丢包，丢包的随机数种子为 1234，最大延迟为 200ms，发生延迟的概率为 10%，延迟的随机数种子为 5678，最大分段大小为 4096 字节，窗口大小为 8 个最大分段大小，初始延时为 100ms，则启动对应程序的指令应该为：

```
./sender file.txt 127.0.0.1 6666 0.2 1234 200 0.1 5678 4096 8 100

./receiver receive.txt 127.0.0.1 6666 0.2 1234 200 0.1 5678
```

如果使用 Java 实现，对应的指令应该为：

```
java sender file.txt 127.0.0.1 6666 0.2 1234 200 0.1 5678 4096 8 100

java receiver receive.txt 127.0.0.1 6666 0.2 1234 200 0.1 5678
```

使用其他语言（如 Python，Rust 同理）。

三、提交内容和检查方式

1. 源代码及编译后的程序（如果有，可执行程序、java 类文件等），源代码的编码规范性和注释完整程度也作为评分的依据。

2. 一份包含以下内容的文档

2.0 使用的语言（可使用任意一种语言，但仅能使用语言提供的 UDP 及 Raw Socket 的类库或类）及运行的方式等。

2.1 收发双方传输的报文的格式（编写的格式可参考维基百科上的 TCP 的报文格式），如果是基于 UDP 实现协议，则编写报文格式时应把 UDP 部分也放置

2.2 主要的收发控制机制，包含正常情况及异常情况，比如连接的建立与断开，正常收发，数据报文发生差错或丢包，控制报文发生丢包，数据包到达发生延迟等。

2.3 关键的数据结构或类的简短说明，只需说明名字及他们的功能即可。

2.3 对控制台日志输出的格式说明以及简短实例（不需要以实际运行程序输出的日志作为实例，只需要说明其格式即可）。日志的格式可包含的内容参考如下：发包的时间（相对于程序启动的时间，以秒或者毫秒为单位）、发送包的序列号和/或确认号、发包的标志位（如 SYN、ACK、FIN 等）、当前包的载荷大小（不必展现载荷内容）、当前发生了人工丢包（drop）、人工延迟（delay）或者正常发送（send）、正常接收（receive）。

2.4 在单机、人工不进行丢包或延迟的条件下，发送一个 8KB 左右的任意文本和一个 1MB 以下的图片文件，对收发双方的日志进行截图（不必粘贴完整文本）。

2.5 在单机、人工进行丢包或延迟的条件下，发送 1MB 以下图片文件的运行时间及能展现发生了丢包/延迟后如何处理相应情况部分的日志截图（比如展现发生了丢包后发送方重传的日志截图等）。

2.6 （可选）在联机、人工不进行丢包或延迟的条件下，发送一个 1MB 以下的图片文件、并且与单机、人工不进行丢包时进行对比，识别出哪些地方发生了丢包。

3. 检查时需要运行所实现的程序，传输检查者所指定的文件，并且向检查者简单解释代码。