Lab 1 - Reliable Data Transport Protocol

总体策略

- 使用Go Back N策略。
- sender为每个包维护计时器，超时后发送窗口中的包要全部重发，重新计时。由于receiver只顺序接收，窗口大小为1，所以sender收到ack后将发送窗口中序列号位于ack number之前的包移出窗口，实现发送窗口的移动。
- 接收窗口大小为1，receiver收到与期望序列号一致的包时保留，若收到之前（一定范围内）已经接收过的旧包，则可能是ack在传送时出错，则需重发该包的ack。
- 为了对上层提供简单透明的抽象，receiver对其收到的包进行暂存，当完整收到可组成原来message的包时再将包拼接后传给上层。

包头设计

包头结构如下表所示：

checksum1	chechsum2	sequence/ack number	payload size	flag	payload
2 byte	2 byte	1 byte	1 byte	1 byte	rest

- 因为是单向传输，所以发送方只发sequence number，接收方只发ack number，可以用同一个位置储存
- checksum放开头，这样包剩下的数据是连续地址，方便生成checksum。checksum2保存除checksum外所有数据计算的校验和，checksum1用同样计算方法计算包含checksum2的校验和，提高面对数据损坏的容错率。
- flag用来代表包类型，服务于传输和包的拼接。flag为0时代表包位于message的中间，flag为1时代表包位于message最后，flag为2时代表包是ACK。

实现细节

Sender

- 用链表实现发送方窗口，最先发的包位于链表头，新发的包接入链表末尾，链表长度上限为窗口大小。
- 链表中元素包含打包好的包本身以及关于该包的相关信息，如发送时间、过期间隔。当需要重设Timer的时候可通过链表头元素的信息设置计时器到期时间，从而用一个Timer满足对每个包的计时要求。
- 由于无法阻止上层的包传送，因此用队列保存等待窗口的包，当窗口空闲时依次进入发送窗口。

Receiver

- 用队列实现包的暂存和拼接。

遇到的困难及解决

1. 最初遇到timeout后没重设timer导致程序没有反应到结束。
2. 为了尽可能减少内存拷贝，链表元素保存packet的地址，而地址始终是Sender_FromUpperLayer中声明的packet的地址，在为之后的包拷贝内容时，发现窗口中原来的包内容也跟着改变了。
3. 打包后应更新next_seq_to_send；判断收到的ack number在expect ack和next seq之间时，若窗口

满了，next seq应取窗口末尾元素的seq number + 1。

    - 访问空指针内容时发生segmentation fault
    - 对假指针（比如std::list中存放指针，list为空时调用list.front()返回的元素为假指针）进行free时发生free() invalid pointer
    - free的范围越界发生free() double free or currupt

4. list.end()会返回尾巴哨兵的迭代器，并不是最后一个元素，访问最后一个元素用list.back()。

测试结果

- 能够通过0.02的正确性测试
  <img src="https://github.com/Shang-QY/DS-Labs/blob/main/images/correct_0.02.png" width="100%">
- 0.15和0.3的性能测试如下
  <img src="https://github.com/Shang-QY/DS-Labs/blob/main/images/ref_0.15.png" width="100%">
  <img src="https://github.com/Shang-QY/DS-Labs/blob/main/images/ref_0.3.png" width="100%">