项目2 –可靠的传输

本作业中, 你将构建一个简单可靠的传输协议, “BEARS-TP” (BTP)。你的协议必须提供顺序（in-order），可靠的UDP报文传输, 并且必须在存在包丢失, 延时, 损坏, 重复, 和顺序重排的情形之下实现。

有多种方式确保消息从发送端到接收端的可靠传输。我们将提供接收端的一个参考实现 (你必须使用它), 它在每次接收到数据包时将返回一个累积的ACK。后面将使用一个例子进一步解释。你的任务是实现一个发送器，当发送包到接收端时，得到可靠的发送。对于附加分，你可以选择实现某个下面描述的性能改进。

协议描述

我们的简单协议有四种类型: `start`, `end`, `data`, 和 `ack`. `Start`，`end`和`data`，消息都遵循下面的一般格式：

```
start|<sequence number>|<data>|<checksum>
data|<sequence number>|<data>|<checksum>
end|<sequence number>|<data>|<checksum>
```

为了开始一个连接，发送一个`start` 消息。接收端将使用所提供的序列号（sequence number）作为该连接所有包的序列号。发送`start` 消息后，在同一连接中使用data消息类型发送更多包，合理调整序列号。显然，连接中最后发送的数据应该是`end`消息类型，以便告知接收端连接完成。你的发送端应该用下列形式进行接收回应:

```
ack|<sequence number>|<checksum>
```

一个重要的限制是包的最大尺寸。UDP协议有一个8字节的包头，而在其下的IP协议有一个约20字节的头。因为我们要使用以太网，其最大帧尺寸为1500字节，这给你的整个包留下了1472个字节(消息类型，序列号，数据和校验和).

尖括号 (“<” 和 “>”) 不是协议的一部分。但是，你应该确保在分隔线和包字段之间没有多余的空间。对于具体格式的细节，参看所提供的示例代码。

接收端需求说明

我们将提供一个简单的接收端；所提供的参考实现同时会作为评分，因此请确保发送端与其兼容[1]。BEARS-TP 接收端累加应答来响应数据包。当接收到类型为`start`, `data`, 或 `end`消息时，接收端产生一个 `ack` 消息，并带上一个下一次希望接收的包的序列号，此序列号是未收到的最小序列号。换言之，如果它希望收到的包序列号是N，下面两种情形将发生。

1. 如果收到的包的序列号不等于N，它将发送“`ack`|N”.
2. 如果收到的包的序列号是N, 它将检查已收到排好序（in-order）包的最大序列号(如M)并发送“`ack`|M+1”. 例如，它已经收到了包N+1和N+2 (即 M = N+2), 但没有比N+2大的，那么它将发送“`ack`|N+3”.

下面举例说明。假定已经发送包0, 1, 和 2, 但包1在到达接收端之前已经丢失。接收端将在接收到包0时，发送 “`ack`|1”, 然后在接收到包2包再次发送“`ack`|1”(*上面的情形1*)。而在接收到包1后 (由于发送端两次发送)，它将发送 “`ack`|3” (由于其已经接收到2)(*上面的情形2*)，而在接收到此应答后，发送方认为对方已经成功收到

[1] 我们将对于多种情形来测试你的发送端：如包丢失和数据损坏。仅在一个测试环境下你的发送端是兼容的，并不能确保你能获得完全分数！你的发送端应能处理本文档中所给的各种场景。

所有三个包。

如果下一下期望包是N，接收端将丢弃所有序列号大于N+4的包；即，接收端以5个包为操作窗口，并丢弃所有在此范围之外的包。当下一个期望包是N+1时(由于N已经到达)，则接收端将接收包到N+5。

可以假定一旦包被接收端应答，它已经被正确接收。接收端的缺省失效时间是10秒；在此时间内没有接收到包，它将自动关闭连接。

发送端需求说明

发送端应该读入一个输入文件，并使用UDP套接字来发送到一个指定的接收端。它应该将输入文件分割成合适大小的数据块，为连接指定一个初始序列号，然后对每个包附加一个校验和。对于连接中的每个新包序列号应加1。校验和的产生和验证函数已经提供给你(见Checksum.py).

你的发送端必须实现一个可靠的传送算法(如滑动窗口)。接收端的窗口大小是5个包，而多于此将忽略。此发送端必须能接收来自接收端的ack包。任何带有无效校验和的ack包将被忽略。

你的发送端应该在下列网络环境下提供可靠的服务：

- 丢包(Loss): 你应该能够处理100%丢包的情形。
- 损坏(Corruption): 任意类型和频率。
- 重新排序(Re-ordering): 可能以任何顺序到达，并且
- 重复(Duplication): 你可能会看到一个包许多次。
- 延时(Delay): 包可能无限期地延时(但在实际中，通常不超过10s)。

你的发送端应该使用下列命令来激活：

```
python Sender.py -f <input file> -a <destination adddress> -p <port>
```

关于发送端的一些最后注解：

- **发送端应实现一个500ms的重发时钟来自动重发从未应答的包(可能是由于ack包丢失)。**我们不期望你使用自适应过时(但这是附加分bells and whistles的选项).
- 你的发送端应该支持窗口大小为5的包(即, 5个未应答的包)。
- 你的实现应该大致满足或者超过应当实现的基于BEARS-TP发送端的滑动窗口的性能(包括时间和完成一次传递需要的包的个数)。
- 你的发送端应能处理任意消息数据(即, 它可以和发送文本文件一样容易地发送图像文件)。如果没有提供输入文件，你的发送端应该从STDIN中读取输入。
- 收到校验和为无效的任何包都应被忽略。
- 实现应能在运行合适Python版本的任何系统下工作(已对运行于Ubuntu 11.04下的2.7.1版和SunOS5.10下的2.6.5版做过测试)
- 你的发送端应该使用Python编写，并遵循PEP 8, Style Guidelines for Python Code中所给标准
- 在正常运行时，你的发送端绝不能产生控制台输出；Python的异常消息是可以的，但是要避免你的代码崩溃。

我们会从正确性，完成传递的时间，包发送次数（及重发）来评估你的发送端。传递完成时间和包发送次数将用我们自己的基于发送端的滑动窗参考实现来评估。注意“Stop-And-Go”发送端(以下所描述的)将不会有足够的效率。

测试Testing:

你需要负责自己项目的测试。为了使这个过程简单一些，示例代码中已经包括了一个测试装置，你可以

使用它来对你的项目进行开发测试。阅读项目代码中的README，TestHarness.py 以及test文件中的测试用例的举例，你可以获取更多细节。需要注意的是，在本次作业中通过测试用例的测试是得到好成绩的必要条件，但不是充分条件！这些测试只是意味着基本功能的测试与你自己测试用例的测试；你必须考虑发送端可能遇到的边界情况类型以及需要保证正确性的各种情况。

提示与技巧

开始时，仅关注你的包中没有任何坏事发生的简单情形。在此情形能工作后，你可以考虑如何处理丢包。

开始时实现一个完全可靠的发送端是有益的。最简单的可靠传递机制是“Stop-And-Go”，此时发送端传送一个简单包并等待接收到应答包，然后才发送更多的。可以先构建一个“Stop-And-Go”发送端，然后以此为基础扩展成完整的滑动窗。理解“Stop-And-Go”发送端仅是铺路石，我们需要你提供完整的BEARS-TP发送端来测试。

Bells and Whistles (附加分)

其中一些可能需要修改所提供的接收端实现;如果这样做了,请确保提交你的接收端实现。

(1)可变大小滑动窗口: 对于包丢失，损坏，延时及重排极少的网络，一个大的窗口将获得高性能。而对于有损的网络，大窗口将由于重传而产生大量耗费。修改你的发送端基于网络条件来动态调整窗口大小。

(2)选择性应答: 如果发送端确切的知道接收端接收到哪些包，就可以仅传递丢失的包 (而不是简单的最后N个包). 修改接收端来明确的应答接收到哪些包，并为你的发送端添加对选择性重发的支持。

(3)兼容可变环路时间: 等待一个未应答包多长时间进行重发与发送端和接收端的环路时间紧密相关。如果知道环路时间是20ms, 等待500ms再重发是低效的: 在20ms后再等待几毫秒还没有收到回应，你就能比较肯定该包丢失了。修改你的发送端来确定发送端与接收端之间环路的时间，并对重发失效时间进行适当修改。

(4)双向传递: 尽管我们定义的协议仅支持单向数据传递，它可以修改后允许双向传递(即, 连接的两端能同时发送和接收)。实现此功能，根据需要修改发送和接收端。在协议有更新的情况下，确保在你的README.txt文件中提供更新协议的描述。

(5)我是超级黑客: 对于广泛布署的TCP实现找出其一个问题, 然后增强BEARS-TP来处理此问题并优于此TCP。 描述并实现你的改进，然后对你的解答和标准TCP进行评估性测试。

README

和你的解决方案一起提供的为README.txt文件，这应包括：

(1) 你 (和你的搭档)的姓名
(2) 在实现你的发送端时遇到些什么挑战与问题? 你如何进行了解决？花费了多长时间解决？
(3*) 指出你实现了哪些附加分; 描述发送端和接收端做了些什么及它们如何工作。

下载

本项目的示例代码, 以及接收端和效验和实现，都可在https://github.com/shaddi/bears-tp 找到.

提交什么

提交一个文件名中有你和你的搭档姓的.tar文件(如project1-shenker-stoica.tar或project1-katz.tar)。后缀.tar的文件应该包括：

✧ 一个Python文件,***Sender.py***。实现上面描述的RIPRouter的*基本协议*(**不包括**附加分Bells and Whistles!)。

✧ **README.txt/README.pdf**文件.

✧ 你可能还需要包含一些实现附加分协议版本的一些文件，如**bears-tc-sender2.py**和**bears-tc-receiver2.py**。

合作策略

本大作业的设计是希望独立解决问题，但如果你愿意可以找一个伙伴一起工作。不管你是一个人做，还是选择伙伴一起做，得分都是一样的;并且无论你与伙伴的工作量如何分配，你们最后的也是得分一样(因此选择合适的伙伴!).

除了你的同伴之外，**不要**与其他同学共享代码. 你**可以**和别人讨论作业要求或者你的解决方案(如,使用何种数据结构来存储路由表) --远离电脑并且不要共享代码 –不要讨论你方案的细节(如., 使用何种算法来计算路由表)。作业疑似作弊将根据学生手册处理。尽管某初级大学的23%学术腐败发生在计算机科学中，但我们希望你能有一个高的学术道德，并以你自己的工作而自豪。

出错
发包数
时间