Lab 4：基于UDP的可靠传输

一、实验目标

1. 基于 UDP 进行可靠文件传输，比较 GBN 和 SR 的区别。
2. 实现基于丢包和基于延迟的拥塞控制，加强对拥塞控制算法的理解。
3. 在模拟和真实网络环境中进行测试，熟悉网络算法测试流程。
4. 进行数据可视化和结果解释，提升数据处理和分析能力。

二、实验原理

使用网络的应用需要可靠的数据传输。网络层 IP 协议提供的是尽力而为的服务，这意味着它是不可靠的，不能确保报文段按时交付，也不能保证报文段的完整性。因此，需要传输层来实现可靠传输，例如 TCP。

TCP 是可靠的、面向连接的传输层协议，而 UDP 是无连接的、不可靠的传输协议。TCP 的保序性和拥塞控制导致丢包时延迟增大，不适用于即时类游戏等延迟敏感的应用。 所以有些应用开发者会选择 UDP 传输，同时在应用层增加一定的可靠性设计，解决丢包、超时、乱序等问题，并满足在延迟方面的需求。另一方面，使用 UDP，可靠性设计的决定权在开发者手上，而无需受限于 TCP 的设计，灵活度更高。目前已有的基于 UDP 的可靠传输协议包含 UDT，QUIC 等。

TCP 的另一个重要功能是拥塞控制，该功能对保持网络稳定、优化网络性能、保障用户公平有着重要的意义。因此当我们基于 UDP 进行可靠传输时，也需要实现一定的拥塞控制功能。不同拥塞控制算法的区别主要在于识别网络拥塞的依据，以及通过调整发送窗口大小进行拥塞避免的策略。

三、实验内容

File Transfer Protocol（FTP）是用于在客户端和服务器之间进行文件传输的应用层协议，传统的 FTP 协议使用 TCP 传输。 FTP 服务一般运行在 20 和 21 两个端口。在本实验中，我们将

通过 Python 的 socket 库，实现基于 UDP 的 FTP，并进行拥塞控制。为了和服务器设置保持一致，要求使用 Python 3.12。

提交内容：代码和报告，后者应包含设计思路、实现细节、测试结果及解释。

评分标准：

- 实现两种可靠传输 25%
- 实现两种拥塞控制 25%
- 模拟环境测试及分析 25%
- 真实环境测试及分析 25%

3.1 算法实现

类似 Lab 3，本次实验要实现客户端和服务端程序，这次通信的两个进程会位于不同主机。客户端通过命令行向服务器上传或下载大文件，服务器同时支持多个客户端，双方交互的命令行格式自由设计。每次传输完成后，应通过比较原文件的 md5 和 接收到文件的 md5 验证文件完整性，若文件出错则报错并停止系统运行。整体设计目标为：在保障可靠性传输的基础上，尽可能提高平均上传下载速度，降低丢包率。

要求实现 **Go Back N（GBN）** 和 **Selective Repeat（SR）**两种重传策略，以及**基于丢包**和**基于延迟**的任意两种拥塞控制算法。基于丢包的拥塞控制算法可参考 TCP Reno、TCP Cubic 等，基于延迟的拥塞控制算法则可参考 TCP Vegas，也可自主设计拥塞控制算法。由于测试时要切换和组合不同的算法，子类等面向对象编程技术可能会让代码更加整洁和易于测试，但不对实现方式和编程风格作强制要求。

3.2 性能测试

鉴于网络设计涉及多方面的权衡，经常需要测量多个指标来全面评估算法的性能。以下是本次实验要求测量的指标：

- 有效吞吐量：文件大小 / 传输时间
- 流量利用率：文件大小 / 发送的总数据量

本实验包含模拟环境和真实环境测试，要求对每个测试内容获取不少于 4 组数据，以统计图的方式呈现结果。测试结果的好坏不影响本实验的评分，但希望同学们尊重实验原始数据，在报告中详细说明参数设置使得实验可复现，并结合代码实现和所学知识对观察到的现象给出解释。

测试编号	环境	可靠传输算法	拥塞控制算法	测试内容
1	模拟	GBN/SR	基于丢包	有效吞吐量 随 丢包率 变化
2	模拟	GBN/SR	基于丢包	流量利用率 随 丢包率 变化
3	模拟	SR	基于丢包/延迟	有效吞吐量 随 丢包率 变化
4	模拟	SR	基于丢包/延迟	流量利用率 随 丢包率 变化
5	模拟	SR	基于丢包/延迟	有效吞吐量 随 延迟 变化
6	模拟	SR	基于丢包/延迟	流量利用率 随 延迟 变化
7	真实	GBN/SR	基于丢包	有效吞吐量 随 上 传文件大小 变化
8	真实	GBN/SR	基于丢包	流量利用率 随 上 传文件大小 变化
9	真实	SR	基于丢包/延迟	有效吞吐量 随 下 载文件大小 变化
10	真实	SR	基于丢包/延迟	流量利用率 随 下 载文件大小 变化

3.2.1 模拟环境测试

由于真实网络环境非常复杂，并且在真实网络中部署测试可能需要大量资源和时间，网络研究中常常先使用模拟环境测试观察算法效果。模拟环境提供了一个安全、可控、成本效益高的平台，有助于开发人员更好地理解和改进算法，是真实测试前的必经步骤。

在本部分实验中，推荐将服务端和客户端分别部署在本机和 Linux 虚拟机，以便在发包时在客户端使用 Linux 环境下的 netem 和 tc 控制网络条件。也可以使用其他有模拟网络延迟和丢包功能的模拟器，但要在报告中注明使用方法和配置。

netem 是 Linux 2.6 及以上内核版本提供的一个网络模拟功能模块。该功能模块可以用来在性能良好的局域网中，模拟出复杂的互联网传输性能，诸如低带宽、传输延迟、丢包等等情况。

tc 是 Linux 系统中的一个工具，全名为 traffic control 。每个网络接口（Network Interface Card，NIC，网卡）与一个队列相关联，数据包在从网络接口被发送出去之前都要先进入这个队列，队列规则决定数据包的发送顺序，也即所有流量整形都发生在队列中。tc 能够创建和管理队列规则，它可以调用 netem ，为某个队列模拟丢包或延迟。tc 在发包端设置，使用示例如下：

```
# 查询系统中所有网络接口的信息，找到状态为 UP （启用）的接口，例如 ens33
ip a
# 在 ens33 接口添加一个队列规则（qdisc），使用 netem 网络模拟器，将丢包率设置为 5%
sudo tc qdisc add dev ens33 root netem loss 5%
# 删除 ens33 接口的队列规则
sudo tc qdisc del dev ens33 root
# 设置延迟为 10ms、抖动为 ±1ms、下一个包和上一个包的延迟相关系数为 10%
sudo tc qdisc add dev ens33 root netem delay 10ms 1ms 10%
# 修改已存在的 netem 配置，将丢包率改为 20%
sudo tc qdisc change dev ens33 root netem loss 20%
# 查看当前队列规则
sudo tc qdisc show
```

Elearning 上提供了一个简单的在本机和虚拟机之间进行不可靠传输的示例，每次运行只从客户端向服务端上传一个固定的文件。将 server.py 和 client.py 分别放置在主机和虚拟机，bomb2.tar 放在虚拟机上 client.py 同一目录下。使用 ipconfig、ifconfig 等命令查询服务端所在机器的 IP（例如 WLAN IPv4 地址），修改两个代码内的 SERVER_IP。先后运行服务端和客户端，使用 md5sum 命令检查接收到的文件 md5 值，应当和原文件一致。使用 tc 和 netem 添加丢包后，再次运行代码，收到的文件 md5 值应当与原文件不同。

由于每个同学实现的算法和传输的数据都不完全相同，不统一规定实验中使用的参数范围，但是同学们应根据自己的设计自行探索调试，使用效果相对明显的设置进行测试。将测试配置作为参数，编写脚本自动进行测试可以帮助同学们减少手动测试的负担，但不做强制要求。

3.2.2 真实环境测试

服务器的基本配置为：阿里云轻量应用服务器，美国硅谷节点，2核4G内存，Ubuntu 24.04 镜像。

服务器 IP 见 Elearning 上 `Lab 4 服务器.xlsx` 文件。

服务器的使用方法为：

- 用例如 `ssh 19307130296@47.254.22.72` 的指令登录服务器。用户名为学号，初始密码为 123456，登陆后请尽快使用 `passwd` 指令修改密码，并记住自己的密码。
- 使用 `scp` 等指令将本地代码和文件上传到服务器测试，上传代码时请注意文件权限。
-

- 由于服务端需要部署在有公有 IP 的设备上，而个人电脑一般只有私有 IP，应当在服务器上部署服务端，在本机部署客户端。在客户端代码中使用服务器的公有 IP 进行寻址，在服务端代码使用服务器的私有 IP。

使用服务器的注意事项如下：

- 由于 vscode 占用内存过多，禁止使用 vscode 连接服务器，只能通过命令行登录和操作。
- 尽量在本地调试和修改代码，在服务器上仅作程序的运行测试。
- 测试传输的文件不得超过 200M，以减轻内存压力。
- 为防止服务器不稳定，重要代码请在本地保留备份。
- 请合理使用服务器，不要做危害服务器安全或影响其他同学测试的行为。
- 本地 IP 可能经过 NAT 转换，可能需要先向服务端上传文件，服务端获取本地 IP 后再进行下载文件操作。

Previous
Lab 3：Socket编程

Next
附录A：Wireshark 教程