

姓名：谢志康

学号：22307110187

实验名称：Lab 4：基于 UDP 的可靠传输

时间：24.11.1

配置 python3.12, 实现基于 UDP 的 FTP。首先编程逻辑上大致还是一个 client 一个 server, server 提供多线程服务, 允许多个 client 同时连接。然后 UDP 需要我们自己设计, 所以我选择做一个 UDP 类, 在类里面分别实现**两种可靠传输** (Go Back N (GBN) 和 Selective Repeat (SR) 两种重传策略) 和**两种拥塞控制** (基于丢包和基于延迟的任意两种拥塞控制算法)。

首先看 client.py 和 server.py, 这其实是两个壳子, 比较简单, 真正核心的逻辑都在后面 udp.py 里面实现。

Client.py:

```
mode = input("Please input mode('S' for sending, 'R' for receiving): ")
file_name = input("Please input your file name: ")
```

我这样定义交互方式, 因为要求实现从服务器上传或下载两种方式, 因此定义 S 和 R 两个 mode, 然后客户端再输入想要上传 (下载) 的文件名。

接下来 bind 地址和端口号使用 udp 连接服务器即可。然后分类, S 表示 send, 向服务器传输文件: 调用 udp.send() 即可。R 表示 receive, 调用 udp.recv()。最后计算 md5 值显示。

Server.py:

也是首先绑定 ip 和端口号, 直接一个永久循环等待 client 连接 (ctrl C 结束), 要支持连接多个 clients, 和上次 lab 逻辑一样, 使用 threading 库即可, 为每个 clients 分配新 port (写了个函数检查这个 port 确实没被用过), 进行连接。最后通知该 client: pack('B', -1, packet['seq'], port), 将其端口号传过去。Thread 类定义在 udp.py 中了, 具体逻辑比较简单, 就是注意 S (send) 是客户端 send, 对于 server 也就是调用 udp.recv。vice versa。

Udp.py:

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

为计算指标和方便传输定义 task 类, 每次 send 的 data 长度 (bytes 数) 会被累加到一个总和值 byte\_count 里面, 为后续计算流量利用率。Send 封装为 task.sendto() (调用了 socket.sendto() 函数)

```
print("传输总时间: ", elapsed_time, 's')
print("有效吞吐量 (文件大小 / 传输时间): ", self.file_size / elapsed_time / 1024, 'KB/s')
if self.byte_count > 0:
    print("流量利用率 (文件大小 / 发送的总数据量): ", self.file_size / self.byte_count)
else:
    print("没有发送任何数据, 无法计算流量利用率。")
```

实现两种可靠传输: 1、2 两种重传策略。

实现两种拥塞控制：3、4 一个基于丢包一个基于延迟的拥塞控制算法。

### 1. Go Back N （GBN）重传策略：

一种滑动窗口协议，当发送方检测到丢包或超时（即代码中触发 `resend` 时），会从丢包的起始位置开始重传整个窗口中的所有未确认的包。

```
2 usages
def resend_gbn(self):
    """Resend packets in the buffer."""
    # GBN策略：将整个窗口中的包都重传
    valid_packets = [pkt for pkt in self.buffer if pkt is not None]
    resend_count = min(len(valid_packets), self.window_size)
    if resend_count > 0:
        print(f'Resending {resend_count} packets starting from seq', pickle.loads(valid_packets[0])['seq'])
        self.num_resends += resend_count
        for packet in valid_packets[:resend_count]:
            self.send_packet(packet)
```

将整个窗口（`window_size`）中的有效包都重传，重传次数加上总数。

### 2. Selective Repeat （SR）重传策略：

只重传未被确认的包。如果序列号小于左窗口边界，表示已确认。

```
def resend_sr(self):
    """Resend packets in the buffer using Selective Repeat strategy."""
    # SR重传策略：只重传未被确认的包
    for i in range(len(self.buffer)):
        if self.buffer[i] is not None: # 检查当前包是否有效
            seq_num = pickle.loads(self.buffer[i])['seq']
            if seq_num < self.left_seq: # 如果序列号小于左窗口边界，表示已确认
                continue
            print(f'Resending packet with seq {seq_num}')
            self.send_packet(self.buffer[i]) # 重传未确认的包
            self.num_resends += 1 # 增加重传计数
```

只需将窗口右边未被确认的包重传即可。

### 3. 基于丢包的 TCP Reno 拥塞控制算法：

TCP Reno 在检测到重复的 ACK（通常为 3 个）时，表明可能发生了丢包，此时会执行“快速重传”（Fast Retransmit）并减小窗口大小。

参考 [TCP 拥塞控制算法（Tahoe/Reno/Newreno）-腾讯云开发者社区-腾讯云](#)

```
self.stop_timer() # 停止当前定时器，避免重复
# TCP Reno拥塞控制（基于丢包）：在超时时将阈值减为window大小的一半，窗口重置为1，开始重传
self.threshold = max(1, self.window_size // 2) # 缩小为window大小的一半
self.window_size = 1
self.resend_gbn() # 执行重传 gbn方法
# self.resend_sr() # 执行重传 sr方法
```

发送方每次成功接收到一个确认信息（ACK）后，会将发送窗口的大小线性增加。

```
# TCP Reno拥塞控制：在无拥塞情况下增大窗口（拥塞避免）
self.udp.window_size += 1 # 增加窗口大小
```

检测到三个重复的 ACK 时就认为发生丢包，参考链接的做法，将 `threshold` 缩小为 `window` 大小一半，然后 `window_size` 变为 `threshold+3`。（并不回到 `slow_start` 的阶段）

```

elif ack_num == left:
    self.num_acks += 1 # 重复ACK计数
    # TCP Reno拥塞控制：检测3个重复ACK时，认为发生丢包
    if self.num_acks >= 3:
        self.udp.threshold = max(1, self.udp.window_size // 2) # 缩小为window大小的一半
        self.udp.window_size = self.udp.threshold + 3 # 缩小窗口
        self.num_acks = 0

```

#### 4. 基于延迟的 TCP Vegas 拥塞控制算法：

这就需要在拥塞控制逻辑中增加**延迟测量**和相应的窗口调整机制。TCP Vegas 通过监测网络延迟来检测拥塞，并动态调整拥塞窗口，他的目的是尽量避免丢包，而不是依赖于丢包的反馈。（因为 TCP Reno 是使用丢包作为拥塞信号，通常会在检测到丢包时将拥塞窗口减半，并进入慢启动。而 TCP Vegas 是使用 RTT（往返时延）来判断网络的状况，并在延迟增加时调整拥塞窗口，防止拥塞发生）。

具体实现上我们需要有个预估的 rtt 作为 benchmark，参考[优秀的 tcp vegas 拥塞控制大揭秘-CSDN 博客](#)。在 UDP 类中新定义三个量：

```

self.rtt_sum = 0
self.rtt_count = 0
self.est_rtt = 0 # 估计的 RTT

```

初始化的时候我们先拿前十个 ack 的 rtt 时间取平均值作为初始的估计 rtt。后续接收过程中反复更新 rtt 估计值，以前 i 轮的 rtt 均值作为第 i+1 轮的指标，要是第 i+1 轮的 rtt 大了，也就说明可能网络开始不好了。

```

packet = pickle.loads(self.udp.send_socket.recv(self.udp.buffer_size))
ack_num = packet['ack']
if self.current_ack == -1: # 第一个 ACK 到达
    self.udp.start_time = time.time() # 记录开始时间
rtt = time.time() - self.udp.start_time # 计算 RTT
if self.udp.rtt_count < 10: # 仅在前 10 次 ACK 中进行 RTT 计算
    self.udp.rtt_sum += rtt
    self.udp.rtt_count += 1
    self.udp.est_rtt = self.udp.rtt_sum / self.udp.rtt_count # 更新估计的 RTT
else:
    # 更新估计的 RTT 和偏差
    alpha = 0.125 # 平滑因子
    self.udp.est_rtt = (1 - alpha) * self.udp.est_rtt + alpha * rtt

```

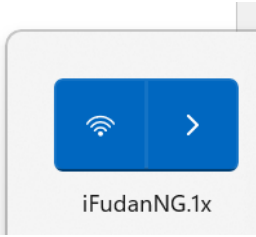
TCP Vegas 会在 rtt 增加时减少窗口，因为此时网络变差了，让 rtt 减少的时候（网络变好）增大窗口

```

# TCP Vegas拥塞控制
# 基于延迟的窗口调整逻辑。TCP Vegas 会在 RTT 增加时减小窗口，在 RTT 降低时增大窗口。
if self.udp.est_rtt > self.udp.threshold: # 如果 RTT 超过阈值，减小窗口
    self.udp.window_size = max(1, self.udp.window_size - 1)
else:
    self.udp.window_size += 1 # 否则增加窗口

```

所有运行情况的网络连接：iFudanNG.1x  
地点：IF 楼一楼大厅  
时间：2024.11.1 晚 10 点开始  
测试内容的每一项都包含有效吞吐量和流量利用率两项。



模拟环境下，server.py 文件地址：  
root@kurumi:/mnt/c/Users/12980/PycharmProjects/homework/transfer\_layer#  
client.py 文件地址：  
(base) root@kurumi:/mnt/c/Users/12980/PycharmProjects/homework#  
处于不同层级文件夹，ubuntu22.04 版本  
ip -o link show | grep 'state UP' 显示状态为 up 的接口

```
(base) root@kurumi:/mnt/c/Users/12980/PycharmProjects/homework# ip -o link show | grep 'state UP'
3: loopback0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group default qlen 1000\ link/ether 00:15:5d:b6:54:79 brd ff:ff:ff:ff:ff:ff
4: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group default qlen 1000\ link/ether 20:c1:9b:15:ff:31 brd ff:ff:ff:ff:ff:ff
```

测试编号    环境    可靠传输算法    拥塞控制算法    测试内容

1    模拟环境    SR    基于延迟 Vegas    随丢包率的变化

（随丢包率的变化全部使用 33.jpg 作为传输文件）

丢包率为 0（未设置丢包率）

上传：有效吞吐量（文件大小 / 传输时间）： 825.7561407881255 KB/s  
流量利用率（文件大小 / 发送的总数据量）： 0.9588030765988832  
下载：有效吞吐量（文件大小 / 传输时间）： 771.6142922204281 KB/s  
流量利用率（文件大小 / 发送的总数据量）： 0.9233422961797981

丢包率为 10%

上传：有效吞吐量（文件大小 / 传输时间）： 260.7516435315948 KB/S  
流量利用率（文件大小 / 发送的总数据量）： 0.9588030765988832  
下载：有效吞吐量（文件大小 / 传输时间）： 370.93603290035105 KB/S  
流量利用率（文件大小 / 发送的总数据量）： 0.9233422961797981

丢包率为 30%

上传：有效吞吐量（文件大小 / 传输时间）： 531.7697076048421 KB/S  
流量利用率（文件大小 / 发送的总数据量）： 0.9588030765988832  
下载：有效吞吐量（文件大小 / 传输时间）： 51.32996422604348 KB/S  
流量利用率（文件大小 / 发送的总数据量）： 0.25995543621093525

丢包率为 70%

上传：有效吞吐量（文件大小 / 传输时间）： 339.4016654298111 KB/S  
流量利用率（文件大小 / 发送的总数据量）： 0.8904109589041096  
下载：有效吞吐量（文件大小 / 传输时间）： 203.86385047781957 KB/S  
流量利用率（文件大小 / 发送的总数据量）： 0.8043487868475715

## 2 模拟环境 SR 基于延迟 Vegas 随延迟的变化

(随延迟的变化全部使用 33.jpg 作为传输文件)

### 延迟为 0 (未设置延迟)

上传: 有效吞吐量 (文件大小 / 传输时间): 1062.5162127107653 KB/s  
流量利用率 (文件大小 / 发送的总数据量): 0.9588030765988832  
下载: 有效吞吐量 (文件大小 / 传输时间): 589.7115011905421 KB/s  
流量利用率 (文件大小 / 发送的总数据量): 0.8904109589041096

### 延迟为 100ms

上传: 有效吞吐量 (文件大小 / 传输时间): 317.35176923338895 KB/S  
流量利用率 (文件大小 / 发送的总数据量): 0.8904109589041096  
下载: 有效吞吐量 (文件大小 / 传输时间): 98.16604973143235 KB/S  
流量利用率 (文件大小 / 发送的总数据量): 0.37804827385650785

### 延迟为 500ms

上传: 有效吞吐量 (文件大小 / 传输时间): 55.88105546075117 KB/S  
流量利用率 (文件大小 / 发送的总数据量): 0.2354402214690435  
下载: 有效吞吐量 (文件大小 / 传输时间): 211.7496346435982 KB/S  
流量利用率 (文件大小 / 发送的总数据量): 0.7792430210652509

### 延迟为 800ms

上传: 有效吞吐量 (文件大小 / 传输时间): 175.99320081212522 KB/S  
流量利用率 (文件大小 / 发送的总数据量): 0.7556570479551589  
下载: 有效吞吐量 (文件大小 / 传输时间): 508.2630794876952 KB/S  
流量利用率 (文件大小 / 发送的总数据量): 0.9588030765988832

## 3 模拟环境 SR 基于延迟 Vegas 随传输文件大小的变化

### 文件: udp.py 大小: 15KB

上传: 有效吞吐量 (文件大小 / 传输时间): 458.23797277423733 KB/s  
流量利用率 (文件大小 / 发送的总数据量): 0.9553065430190624  
下载: 有效吞吐量 (文件大小 / 传输时间): 592.0417819698328 KB/s  
流量利用率 (文件大小 / 发送的总数据量): 0.9553065430190624

### 文件: 33.jpg 大小: 36.4KB

上传: 有效吞吐量 (文件大小 / 传输时间): 732.6362135388002 KB/s  
流量利用率 (文件大小 / 发送的总数据量): 0.9588030765988832  
下载: 有效吞吐量 (文件大小 / 传输时间): 651.3886765084955 KB/s  
流量利用率 (文件大小 / 发送的总数据量): 0.9588030765988832

### 文件: 22.jpg 大小: 176.23KB

上传: 有效吞吐量 (文件大小 / 传输时间): 2436.823128287186 KB/s  
流量利用率 (文件大小 / 发送的总数据量): 0.959890412152704  
下载: 有效吞吐量 (文件大小 / 传输时间): 794.3901682817582 KB/s  
流量利用率 (文件大小 / 发送的总数据量): 0.9584664536741214

### 文件: 11.jpg 大小: 257.15KB

上传: 有效吞吐量 (文件大小 / 传输时间): 3015.2910696649155 KB/s  
流量利用率 (文件大小 / 发送的总数据量): 0.9599519184106198  
下载: 有效吞吐量 (文件大小 / 传输时间): 2148.972017913432 KB/s

流量利用率（文件大小 / 发送的总数据量）： 0.9345200023258519

#### 4 模拟环境 SR 基于丢包 Reno 随丢包率的变化

（随丢包率的变化全部使用 33.jpg 作为传输文件）

##### 丢包率为 0（未设置丢包率）

上传：有效吞吐量（文件大小 / 传输时间）： 1568.8037290739396 KB/s

流量利用率（文件大小 / 发送的总数据量）： 0.9588030765988832

下载：有效吞吐量（文件大小 / 传输时间）： 1291.442034509043 KB/s

流量利用率（文件大小 / 发送的总数据量）： 0.9588030765988832

##### 丢包率为 10%

上传：有效吞吐量（文件大小 / 传输时间）： 243.96790826069383 KB/S

流量利用率（文件大小 / 发送的总数据量）： 0.9233422961797981

下载：有效吞吐量（文件大小 / 传输时间）： 163.039137423863 KB/S

流量利用率（文件大小 / 发送的总数据量）： 0.712523979172376

##### 丢包率为 30%

上传：有效吞吐量（文件大小 / 传输时间）： 277.2616455194806 KB/S

流量利用率（文件大小 / 发送的总数据量）： 0.7792430210652509

下载：有效吞吐量（文件大小 / 传输时间）： 235.86333017993363 KB/S

流量利用率（文件大小 / 发送的总数据量）： 0.7556570479551589

##### 丢包率为 70%

上传：有效吞吐量（文件大小 / 传输时间）： 124.06388989436283 KB/S

流量利用率（文件大小 / 发送的总数据量）： 0.5543203484299333

下载：有效吞吐量（文件大小 / 传输时间）： 323.4951810209032 KB/S

流量利用率（文件大小 / 发送的总数据量）： 0.7792430210652509

#### 5 模拟环境 SR 基于丢包 Reno 随延迟的变化

（随延迟的变化全部使用 33.jpg 作为传输文件）

##### 延迟为 0（未设置延迟）

上传：有效吞吐量（文件大小 / 传输时间）： 1271.8435172784427 KB/s

流量利用率（文件大小 / 发送的总数据量）： 0.9588030765988832

下载：有效吞吐量（文件大小 / 传输时间）： 897.4940255112176 KB/s

流量利用率（文件大小 / 发送的总数据量）： 0.9588030765988832

##### 延迟为 100ms

上传：有效吞吐量（文件大小 / 传输时间）： 315.6078271987536 KB/S

流量利用率（文件大小 / 发送的总数据量）： 0.8904109589041096

下载：有效吞吐量（文件大小 / 传输时间）： 316.66896761189497 KB/S

流量利用率（文件大小 / 发送的总数据量）： 0.9233422961797981

##### 延迟为 500ms

上传：有效吞吐量（文件大小 / 传输时间）： 41.70774889705296 KB/S

流量利用率（文件大小 / 发送的总数据量）： 0.20457253332733855

下载：有效吞吐量（文件大小 / 传输时间）： 282.5528882689716 KB/S

流量利用率（文件大小 / 发送的总数据量）： 0.8311261302402045

##### 延迟为 800ms

上传：有效吞吐量（文件大小 / 传输时间）： 180.11812616957351 KB/S



流量利用率(文件大小 / 发送的总数据量): 0.712523979172376  
下载: 有效吞吐量(文件大小 / 传输时间): 357.1662310719941 KB/s  
流量利用率(文件大小 / 发送的总数据量): 0.8597477443431433

## 6 模拟环境 SR 基于丢包 Reno 随传输文件大小的变化

文件: udp.py 大小: 15KB

上传: 有效吞吐量(文件大小 / 传输时间): 475.4234206037464 KB/s  
流量利用率(文件大小 / 发送的总数据量): 0.9553065430190624  
下载: 有效吞吐量(文件大小 / 传输时间): 833.8259616572205 KB/s  
流量利用率(文件大小 / 发送的总数据量): 0.9553065430190624

文件: 33.jpg 大小: 36.4KB

上传: 有效吞吐量(文件大小 / 传输时间): 1232.7658483748542 KB/s  
流量利用率(文件大小 / 发送的总数据量): 0.9588030765988832  
下载: 有效吞吐量(文件大小 / 传输时间): 1105.943091119485 KB/s  
流量利用率(文件大小 / 发送的总数据量): 0.9588030765988832

文件: 22.jpg 大小: 176.23KB

上传: 有效吞吐量(文件大小 / 传输时间): 1143.688564292023 KB/s  
流量利用率(文件大小 / 发送的总数据量): 0.9584664536741214  
下载: 有效吞吐量(文件大小 / 传输时间): 510.1174237931443 KB/s  
流量利用率(文件大小 / 发送的总数据量): 0.8752604941947008

文件: 11.jpg 大小: 257.15KB

上传: 有效吞吐量(文件大小 / 传输时间): 4278.81688149722 KB/s  
流量利用率(文件大小 / 发送的总数据量): 0.9599519184106198  
下载: 有效吞吐量(文件大小 / 传输时间): 4480.334849058449 KB/s  
流量利用率(文件大小 / 发送的总数据量): 0.9599519184106198

## 7 模拟环境 GBN 基于延迟 Vegas 随丢包率的变化

(随丢包率的变化全部使用 33.jpg 作为传输文件)

丢包率为 0 (未设置丢包率)

上传: 有效吞吐量(文件大小 / 传输时间): 1169.4045303382068 KB/s  
流量利用率(文件大小 / 发送的总数据量): 0.9588030765988832  
下载: 有效吞吐量(文件大小 / 传输时间): 973.4679220151738 KB/s  
流量利用率(文件大小 / 发送的总数据量): 0.9233422961797981

丢包率为 10%

上传: 有效吞吐量(文件大小 / 传输时间): 123.21617027114361 KB/S  
流量利用率(文件大小 / 发送的总数据量): 0.4455106237148732  
下载: 有效吞吐量(文件大小 / 传输时间): 539.7160501871521 KB/S  
流量利用率(文件大小 / 发送的总数据量): 0.9233422961797981

丢包率为 30%

上传: 有效吞吐量(文件大小 / 传输时间): 336.684649448661 KB/S  
流量利用率(文件大小 / 发送的总数据量): 0.9233422961797981  
下载: 有效吞吐量(文件大小 / 传输时间): 583.0604982206406 KB/S  
流量利用率(文件大小 / 发送的总数据量): 0.9588030765988832

丢包率为 70%

上传：有效吞吐量(文件大小/传输时间)：675.8340586016826 KB/S  
流量利用率(文件大小 / 发送的总数据量)：0.9233422961797981  
下载：有效吞吐量(文件大小/传输时间)：568.6849191758146 KB/S  
流量利用率(文件大小 / 发送的总数据量)：0.9233422961797981

## 8 模拟环境 GBN 基于延迟 Vegas 随延迟的变化

(随延迟的变化全部使用 33.jpg 作为传输文件)

### 延迟为 0 (未设置延迟)

上传：有效吞吐量(文件大小 / 传输时间)：843.2702695059529 KB/s  
流量利用率(文件大小 / 发送的总数据量)：0.9588030765988832  
下载：有效吞吐量(文件大小 / 传输时间)：492.0867108494178 KB/s  
流量利用率(文件大小 / 发送的总数据量)：0.8597477443431433

### 延迟为 100ms

上传：有效吞吐量(文件大小 / 传输时间)：311.48484196407867 KB/S  
流量利用率(文件大小 / 发送的总数据量)：0.8904109589041096  
下载：有效吞吐量(文件大小 / 传输时间)：418.6034236363994 KB/S  
流量利用率(文件大小 / 发送的总数据量)：0.8904109589041096

### 延迟为 500ms

上传：有效吞吐量(文件大小 / 传输时间)：162.8450347328411 KB/S  
流量利用率(文件大小 / 发送的总数据量)：0.6740491092922485  
下载：有效吞吐量(文件大小 / 传输时间)：187.214747364131182 KB/S  
流量利用率(文件大小 / 发送的总数据量)：0.7858337981037368

### 延迟为 800ms

上传：有效吞吐量(文件大小 / 传输时间)：192.420341310495 KB/S  
流量利用率(文件大小 / 发送的总数据量)：0.8043487868475715  
下载：.有效吞吐量(文件大小 / 传输时间)：224.1080809705402 KB/S  
流量利用率(文件大小 / 发送的总数据量)：0.8904109589041096

## 9 模拟环境 GBN 基于延迟 Vegas 随传输文件大小的变化

### 文件：udp.py 大小：15KB

上传：有效吞吐量(文件大小 / 传输时间)：768.0550127039907 KB/s  
流量利用率(文件大小 / 发送的总数据量)：0.9553065430190624  
下载：有效吞吐量(文件大小 / 传输时间)：429.3936764144676 KB/s  
流量利用率(文件大小 / 发送的总数据量)：0.9553065430190624

### 文件：33.jpg 大小：36.4KB

上传：有效吞吐量(文件大小 / 传输时间)：870.3751919158898 KB/s  
流量利用率(文件大小 / 发送的总数据量)：0.9588030765988832  
下载：有效吞吐量(文件大小 / 传输时间)：685.0536898258125 KB/s  
流量利用率(文件大小 / 发送的总数据量)：0.8597477443431433

### 文件：22.jpg 大小：176.23KB

上传：有效吞吐量(文件大小 / 传输时间)：1357.2544378698226 KB/s  
流量利用率(文件大小 / 发送的总数据量)：0.9584664536741214  
下载：有效吞吐量(文件大小 / 传输时间)：1010.1872357559895 KB/s  
流量利用率(文件大小 / 发送的总数据量)：0.9584664536741214



文件: 11. jpg 大小: 257. 15KB

上传: 有效吞吐量 (文件大小 / 传输时间): 4798. 199683851364 KB/s

流量利用率 (文件大小 / 发送的总数据量): 0. 9599519184106198

下载: 有效吞吐量 (文件大小 / 传输时间): 4159. 619614796795 KB/s

流量利用率 (文件大小 / 发送的总数据量): 0. 9599519184106198

## 10 模拟环境 GBN 基于丢包 Reno 随丢包率的变化

(随丢包率的变化全部使用 33. jpg 作为传输文件)

丢包率为 0 (未设置丢包率)

上传: 有效吞吐量 (文件大小 / 传输时间): 929. 4466748123582 KB/s

流量利用率 (文件大小 / 发送的总数据量): 0. 9588030765988832

下载: 有效吞吐量 (文件大小 / 传输时间): 395. 98949300286046 KB/s

流量利用率 (文件大小 / 发送的总数据量): 0. 8904109589041096

丢包率为 10%

上传: 有效吞吐量 (文件大小 / 传输时间): 450. 41992435319565 KB/S

流量利用率 (文件大小 / 发送的总数据量): 0. 9588030765988832

下载: 有效吞吐量 (文件大小 / 传输时间): 337. 6705968682197 KB/S

流量利用率 (文件大小 / 发送的总数据量): 0. 9233422961797981

丢包率为 30%

上传: 有效吞吐量 (文件大小 / 传输时间): 95. 99262419311016 KB/S

流量利用率 (文件大小 / 发送的总数据量): 0. 47975537747785746

下载: 有效吞吐量 (文件大小 / 传输时间): 246. 40647853571872 KB/S

流量利用率 (文件大小 / 发送的总数据量): 0. 8904109589041096

丢包率为 70%

上传: 有效吞吐量 (文件大小 / 传输时间): 260. 10523230659726 KB/S

流量利用率 (文件大小 / 发送的总数据量): 0. 8904109589041096

下载: 有效吞吐量 (文件大小 / 传输时间): 207. 71398757570142 KB/S

流量利用率 (文件大小 / 发送的总数据量): 0. 7334569194809382

## 11 模拟环境 GBN 基于丢包 Reno 随延迟的变化

(随延迟的变化全部使用 33. jpg 作为传输文件)

延迟为 0 (未设置延迟)

上传: 有效吞吐量 (文件大小 / 传输时间): 1659. 9797365754812 KB/s

流量利用率 (文件大小 / 发送的总数据量): 0. 9588030765988832

下载: 有效吞吐量 (文件大小 / 传输时间): 1569. 8774375605442 KB/s

流量利用率 (文件大小 / 发送的总数据量): 0. 9588030765988832

延迟为 100ms

上传: 有效吞吐量 (文件大小 / 传输时间): 458. 45712425319552 KB/S

流量利用率 (文件大小 / 发送的总数据量): 0. 9588030765988832

下载: 有效吞吐量 (文件大小 / 传输时间): 322. 9444280788092 KB/S

流量利用率 (文件大小 / 发送的总数据量): 0. 8904109589041096

延迟为 500ms

上传: 有效吞吐量 (文件大小 / 传输时间): 258. 67513797393025 KB/S

流量利用率 (文件大小 / 发送的总数据量): 0. 8904109589041096

下载：有效吞吐量(文件大小 /传输时间)：190.13359586612867 KB/S

流量利用率(文件大小 /发送的总数据量)：0.7792430210652509

延迟为 800ms

上传：有效吞吐量(文件大小 /传输时间)：258.1792742309744 KB/S

流量利用率(文件大小 /发送的总数据量)：0.8043487868475715

下载：有效吞吐量(文件大小 /传输时间)：138.06852382442878 KB/S

流量利用率(文件大小/发送的总数据量)：0.47975537747785746

## 12 模拟环境 GBN 基于丢包 Reno 随传输文件大小的变化

文件：udp.py 大小：15KB

上传：有效吞吐量（文件大小 / 传输时间）：463.80666091616223 KB/s

流量利用率（文件大小 / 发送的总数据量）：0.9553065430190624

下载：有效吞吐量（文件大小 / 传输时间）：219.26650451451792 KB/s

流量利用率（文件大小 / 发送的总数据量）：0.6944756554307117

文件：33.jpg 大小：36.4KB

上传：有效吞吐量（文件大小 / 传输时间）：1084.2913660693507 KB/s

流量利用率（文件大小 / 发送的总数据量）：0.9588030765988832

下载：有效吞吐量（文件大小 / 传输时间）：635.715686692534 KB/s

流量利用率（文件大小 / 发送的总数据量）：0.8904109589041096

文件：22.jpg 大小：176.23KB

上传：有效吞吐量（文件大小 / 传输时间）：1444.8145126457744 KB/s

流量利用率（文件大小 / 发送的总数据量）：0.9584664536741214

下载：有效吞吐量（文件大小 / 传输时间）：297.5295312310836 KB/s

流量利用率（文件大小 / 发送的总数据量）：0.8388495777219813

文件：11.jpg 大小：257.15KB

上传：有效吞吐量（文件大小 / 传输时间）：4205.7938723112 KB/s

流量利用率（文件大小 / 发送的总数据量）：0.9599519184106198

下载：有效吞吐量（文件大小 / 传输时间）：2109.555711776452 KB/s

流量利用率（文件大小 / 发送的总数据量）：0.9345200023258519

## 13 真实环境 SR 基于延迟 Vegas 随丢包率的变化

丢包率为 0（未设置丢包率）

上传：有效吞吐量(文件大小 /传输时间)：38.27083265417808 KB/S

流量利用率(文件大小 /发送的总数据量)：0.9588030765988832

下载：有效吞吐量（文件大小 / 传输时间）：20.465598178644342 KB/s

流量利用率（文件大小 / 发送的总数据量）：0.43015835499881827

丢包率为 5%

上传：有效吞吐量(文件大小 /传输时间):21.083447062368762 KB/S

流量利用率(文件大小 /发送的总数据量):0.5938784833257195

下载：有效吞吐量(文件大小 /传输时间):0.3176651355112745 KB/S

流量利用率(文件大小 /发送的总数据量):0.4185888738127544

丢包率为 10%

上传：有效吞吐量(文件大小 /传输时间)：16.307983292822534 KB/S

流量利用率(文件大小 /发送的总数据量)：0.4455106237148732

下载：有效吞吐量（文件大小 / 传输时间）：20.431128068107416 KB/s

流量利用率（文件大小 / 发送的总数据量）：0.43015835499881827

丢包率为 15%

上传：有效吞吐量（文件大小 / 传输时间）：9.445366385966357 KB/S

流量利用率（文件大小 / 发送的总数据量）：0.4455106237148732

下载：几乎就下不下来了

```
$ python3.12 server.py
Server running on 172.17.40.155: 7777
Connection successful!
Preparing to send data
0.4248790740966797
0 0 1 10000
Timeout, resending packets
Timeout, resending packets
Timeout, resending packets
Timeout, resending packets
Timeout, resending packets
Timeout, resending packets
Timeout, resending packets
Timeout, resending packets
```

丢包率为 30%

几乎就一直 timeout resending packet 了

```
ubuntu@ubuntu:~/Desktop$ python3.12 client.py
Please input mode('S' for sending, 'R' for receiving): S
Please input your file name: 33.jpg
Timeout, resending packets
Timeout, resending packets
Timeout, resending packets
Timeout, resending packets
Timeout, resending packets
Timeout, resending packets
Timeout, resending packets
Timeout, resending packets
Timeout, resending packets
Timeout, resending packets
```

#### 14 真实环境 SR 基于延迟 Vegas 随延迟的变化

（随延迟的变化全部使用 33.jpg 作为传输文件）

延迟为 0（未设置延迟）

上传：有效吞吐量（文件大小 / 传输时间）：32.22184393004086 KB/S

流量利用率（文件大小 / 发送的总数据量）：0.9585459977592431

下载：有效吞吐量（文件大小 / 传输时间）：29.370093753412068 KB/s

流量利用率（文件大小 / 发送的总数据量）：0.9585459977592431

延迟为 100ms

上传：有效吞吐量（文件大小 / 传输时间）：21.4995058703354 KB/S

流量利用率（文件大小 / 发送的总数据量）：0.9585459977592431

下载：有效吞吐量（文件大小 / 传输时间）：9.502924053696498 KB/s

流量利用率（文件大小 / 发送的总数据量）：0.7534246575342466

延迟为 500ms

上传：有效吞吐量（文件大小 / 传输时间）：10.266359210596558 KB/S

流量利用率(文件大小 /发送的总数据量): 0.9585459977592431

下载: 有效吞吐量(文件大小 /传输时间): 10.257708701816473 KB/S

流量利用率(文件大小 /发送的总数据量): 0.9585459977592431

延迟为 800ms

上传: 有效吞吐量(文件大小 /传输时间): 5.576404984926837 KB/S

流量利用率(文件大小 /发送的总数据量): 0.9585459977592431

下载: 有效吞吐量(文件大小 /传输时间): 7.040260866808904 KB/s

流量利用率(文件大小/发送的总数据量): 0.9585459977592431

## 15 真实环境 SR 基于丢包 Reno 随延迟的变化

(随延迟的变化全部使用 11.jpg 作为传输文件)

延迟为 0 (未设置延迟)

上传: 有效吞吐量(文件大小 /传输时间): 150.3745349365115 KB/S

流量利用率(文件大小 /发送的总数据量): 0.9599519184106198

下载: 有效吞吐量(文件大小 /传输时间): 12,956357386439315 KB/S

流量利用率(文件大小 /发送的总数据量): 0.029728598645999037

11.jpg 比较大的情况下几乎一直在重发, 后面就用 33.jpg 测了

```

Resending packet with seq 177
Resending packet with seq 178
Resending packet with seq 179
Resending packet with seq 174
Resending packet with seq 175
Resending packet with seq 176
Resending packet with seq 177
Resending packet with seq 178
Resending packet with seq 179
Resending packet with seq 174
Resending packet with seq 175
Resending packet with seq 176
Resending packet with seq 177
Resending packet with seq 178
Resending packet with seq 179
Packet loss rate: 0.9688237276710482
传输总时间: 19.38222599029541 s
有效吞吐量(文件大小 / 传输时间): 12.956357386439315 KB/s
流量利用率(文件大小 / 发送的总数据量): 0.029728598645999037
Received ACK: 180
Received ACK: 180
Resending packet with seq 180
Resending packet with seq 181
Received ACK: 180
Received ACK: 180
Received ACK: 180
Resending packet with seq 180
Resending packet with seq 181
Received ACK: 180
Received ACK: 180
Received ACK: 180
Resending packet with seq 180
Resending packet with seq 181
Received ACK: 180
Received ACK: 180
Received ACK: 180
Resending packet with seq 180
Resending packet with seq 181
Received ACK: 180
Received ACK: 180
Received ACK: 180
Resending packet with seq 180
Resending packet with seq 181
b166d1971b2546159252c416e9db9a43 11.jpg
End of transmission.
Connection closed.

```

延迟为 100ms

上传: 有效吞吐量(文件大小 / 传输时间): 21.16588816524039 KB/s

流量利用率(文件大小 / 发送的总数据量): 0.9585459977592431

下载: 有效吞吐量(文件大小 / 传输时间): 18.11624473576623 KB/s

流量利用率(文件大小 / 发送的总数据量): 0.9585459977592431

延迟为 500ms

上传: 有效吞吐量(文件大小 / 传输时间): 47.1224406061904 KB/s

流量利用率(文件大小 / 发送的总数据量): 0.9585459977592431

下载: 有效吞吐量(文件大小 / 传输时间): 5.52550550175346 KB/s

流量利用率(文件大小 / 发送的总数据量): 0.8043487868475715

16 真实环境 SR 基于丢包 Reno 随传输文件大小的变化

文件: udp.py 大小: 15KB



上传: 有效吞吐量(文件大小 / 传输时间): 14.796222460799367 KB/S  
流量利用率(文件大小 / 发送的总数据量): 0.954661266087411  
下载: 有效吞吐量(文件大小 / 传输时间): 20.276808698977977 KB/S  
流量利用率(文件大小 / 发送的总数据量): 0.954661266087411

文件: 33.jpg 大小: 36.4KB

上传: 有效吞吐量(文件大小 / 传输时间): 25.078720873276886 KB/S  
流量利用率(文件大小 / 发送的总数据量): 0.9588030765988832  
下载: 有效吞吐量(文件大小 / 传输时间): 29.567626140111788 KB/S  
流量利用率(文件大小 / 发送的总数据量): 0.9588030765988832

文件: 22.jpg 大小: 176.23KB

上传: 有效吞吐量(文件大小 / 传输时间): 31.268710023937924 KB/S  
流量利用率(文件大小 / 发送的总数据量): 0.9584664536741214  
下载: 有效吞吐量(文件大小 / 传输时间): 30.590808460697232 KB/s  
流量利用率(文件大小 / 发送的总数据量): 0.9584664536741214

文件: 11.jpg 大小: 257.15KB

上传: 有效吞吐量(文件大小 / 传输时间): 111.07604794902647 KB/S  
流量利用率(文件大小 / 发送的总数据量): 0.949614836370082  
下载: 有效吞吐量(文件大小 / 传输时间): 38.68644650796045 KB/s  
流量利用率(文件大小 / 发送的总数据量): 0.0524333862526859

#### 17 真实环境 GBN 基于丢包 Reno 随丢包率的变化

丢包率为 0 (未设置丢包率)

上传: 有效吞吐量(文件大小 / 传输时间): 28.10705362729401 KB/S  
流量利用率(文件大小 / 发送的总数据量): 0.9588030765988832  
下载: 有效吞吐量(文件大小 / 传输时间): 30.027765145457863 KB/s  
流量利用率(文件大小 / 发送的总数据量): 0.9588030765988832

丢包率为 10%

上传: 有效吞吐量(文件大小 / 传输时间): 21.53070739333921 KB/S  
流量利用率(文件大小 / 发送的总数据量): 0.3616205368674124  
下载: 有效吞吐量(文件大小 / 传输时间): 27.224544691874886 KB/s  
流量利用率(文件大小 / 发送的总数据量): 0.9586852316822643

丢包率为 15%

上传: 有效吞吐量(文件大小 / 传输时间): 27.320706474160332 KB/S  
流量利用率(文件大小 / 发送的总数据量): 0.5116803216276308  
下载: 有效吞吐量(文件大小 / 传输时间): 33.97150528225292 KB/s  
流量利用率(文件大小 / 发送的总数据量): 0.9586852316822643

丢包率为 25%

上传: 有效吞吐量(文件大小 / 传输时间): 14.878586399371843 KB/S  
流量利用率(文件大小 / 发送的总数据量): 0.39034364181323916  
下载: 基本下载不下来了, 一直 resend

#### 18 真实环境 GBN 基于丢包 Reno 随传输文件大小的变化

文件: udp.py 大小: 15KB

上传: 有效吞吐量(文件大小 / 传输时间): 12.557023464941679 KB/S

流量利用率(文件大小 /发送的总数据量):0.8716373396957948

下载:有效吞吐量(文件大小 /传输时间):20.637140534267292 KB/S

流量利用率(文件大小 /发送的总数据量):0.954661266087411

文件: 33.jpg 大小: 36.4KB

上传:有效吞吐量(文件大小 /传输时间): 31.49949488669532 KB/S

流量利用率(文件大小 /发送的总数据量):0.9588030765988832

下载:有效吞吐量(文件大小 /传输时间):16.631140913868517 KB/s

流量利用率(文件大小 /发送的总数据量):0.3578150049954692

文件: 22.jpg 大小: 176.23KB

上传:有效吞吐量(文件大小 /传输时间): 31.614314104869642 KB/S

流量利用率(文件大小 /发送的总数据量):0.9584664536741214

下载:有效吞吐量(文件大小 /传输时间): 31.14820034913638 KB/s

流量利用率(文件大小 /发送的总数据量): 0.9584664536741214

文件: 11.jpg 大小: 257.15KB

上传:有效吞吐量(文件大小 /传输时间): 143.54484154532724 KB/S

流量利用率(文件大小 /发送的总数据量): 0.9599519184106198

下载:有效吞吐量(文件大小 /传输时间): 17.021312629539107 KB/s

流量利用率(文件大小 /发送的总数据量): 0.028121706239005305

(重传了 n 多遍)

## 19 真实环境 GBN 基于延迟 Vegas 随丢包率的变化

丢包率为 0 (未设置丢包率)

上传:有效吞吐量(文件大小 /传输时间): 32.34426492614419 KB/S

流量利用率(文件大小 /发送的总数据量):0.9585459977592431

下载:有效吞吐量(文件大小 /传输时间): 26.09792597318399 KB/s

流量利用率(文件大小 /发送的总数据量): 0.9585459977592431

丢包率为 5%

上传:有效吞吐量(文件大小 /传输时间):26.413435182013618 KB/S

流量利用率(文件大小 /发送的总数据量):0.9585459977592431

下载:有效吞吐量(文件大小 /传输时间): 17.621333416767676 KB/s

流量利用率(文件大小 /发送的总数据量): 0.5703492463242102

丢包率为 20%

上传:有效吞吐量(文件大小 /传输时间):8.124987320486262 KB/S

流量利用率(文件大小 /发送的总数据量):0.8112949109682858

下载:有效吞吐量(文件大小 /传输时间): 18.144384698326853 KB/s

流量利用率(文件大小 /发送的总数据量): 0.5276140879813622

丢包率为 25%

上传:有效吞吐量(文件大小 /传输时间): 11.591622726845891 KB/S

流量利用率(文件大小 /发送的总数据量): 0.7812896352290599

下载:有效吞吐量(文件大小 /传输时间): 4.604131707199576 KB/s

流量利用率(文件大小 /发送的总数据量): 0.4397487150199886

## 20 真实环境 GBN 基于延迟 Vegas 随延迟的变化

(随延迟的变化全部使用 11.jpg 作为传输文件)

延迟为 0（未设置延迟）

上传：有效吞吐量（文件大小 / 传输时间）： 24.6973199713319 KB/S  
流量利用率（文件大小 / 发送的总数据量）： 0.9588030765988832  
下载：有效吞吐量（文件大小 / 传输时间）： 24.04624559509487 KB/s  
流量利用率（文件大小 / 发送的总数据量）： 0.9588030765988832

延迟为 100ms

上传：有效吞吐量（文件大小 / 传输时间）： 22.58332107115251 KB/S  
流量利用率（文件大小 / 发送的总数据量）： 0.9588030765988832  
下载：有效吞吐量（文件大小 / 传输时间）： 22.60314334278798 KB/s  
流量利用率（文件大小 / 发送的总数据量）： 0.8904109589041096

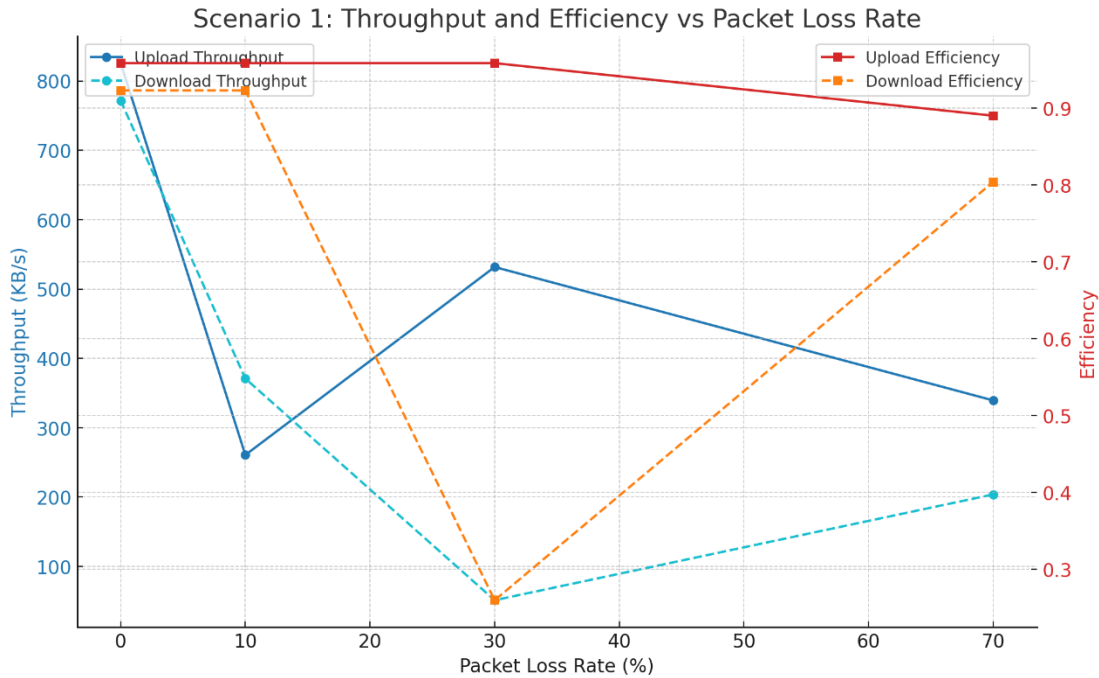
延迟为 500ms

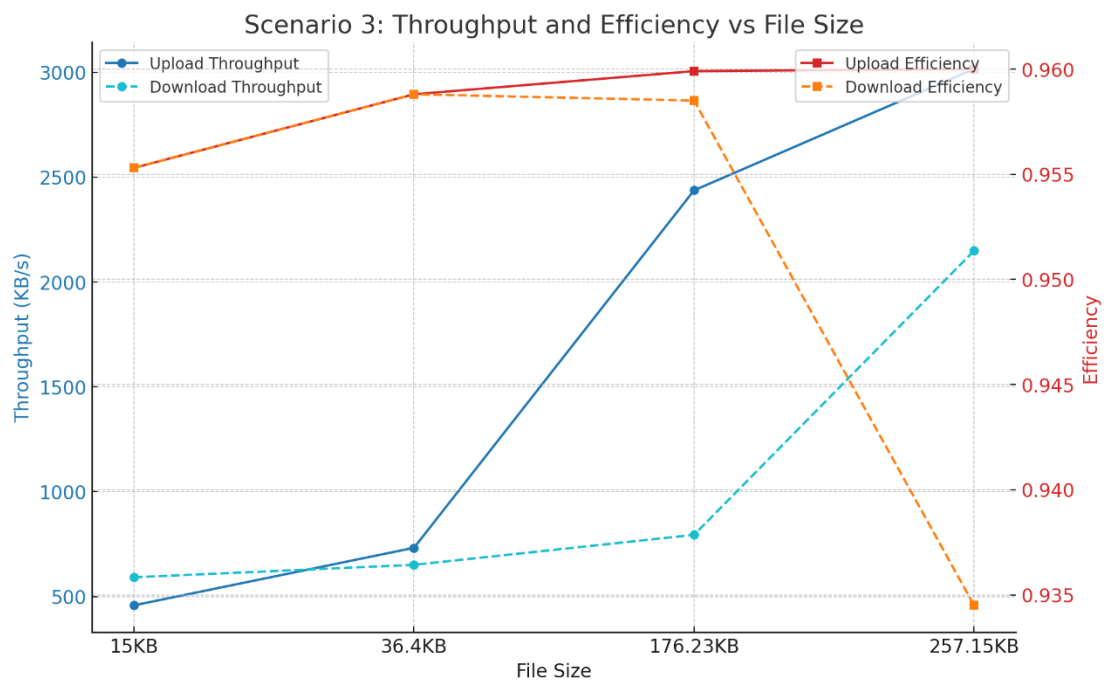
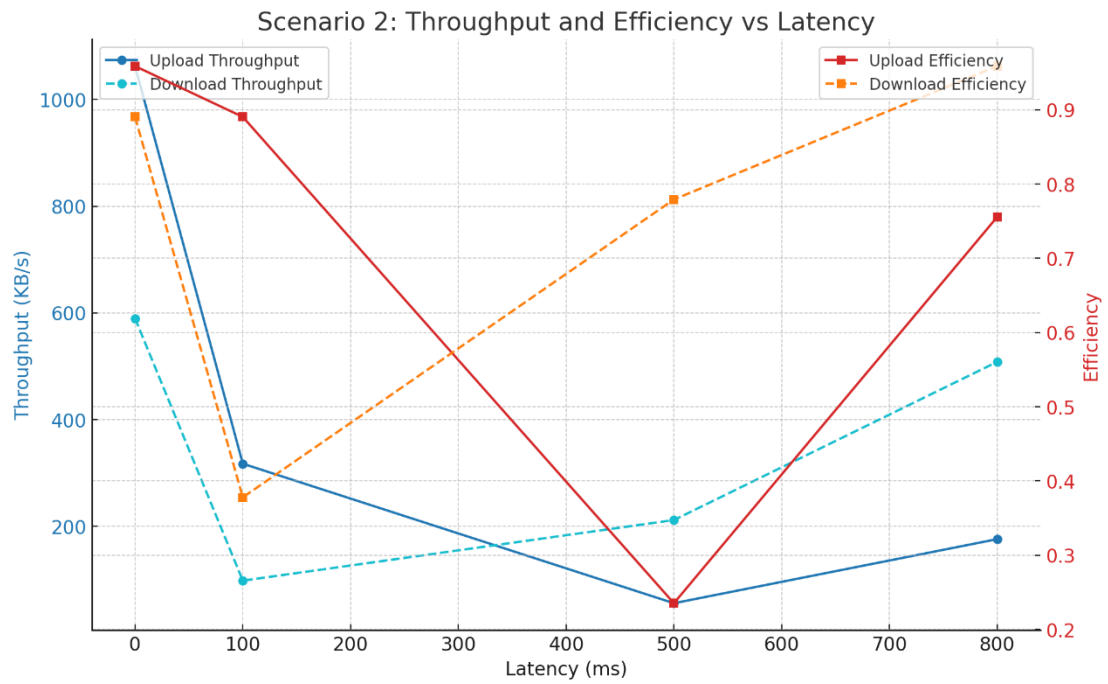
上传：有效吞吐量（文件大小 / 传输时间）： 9.14748776684626 KB/S  
流量利用率（文件大小 / 发送的总数据量）： 0.9588030765988832  
下载：有效吞吐量（文件大小 / 传输时间）： 6.3207675498312925 KB/S  
流量利用率（文件大小 / 发送的总数据量）： 0.3578150049954692

延迟为 1500ms

上传：有效吞吐量（文件大小 / 传输时间）： 4.325309513813759 KB/S  
流量利用率（文件大小 / 发送的总数据量）： 0.9588030765988832  
下载：有效吞吐量（文件大小 / 传输时间）： 4.140673565776216 KB/s  
流量利用率（文件大小 / 发送的总数据量）： 0.9585459977592431

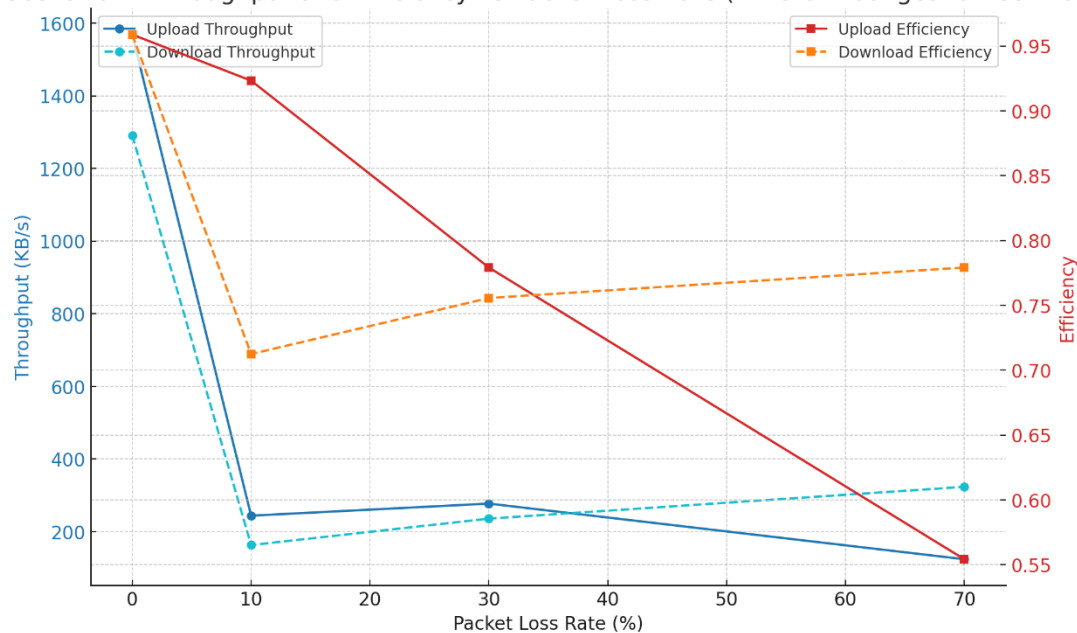
以上是所有实验数据  
以下进行了部分可视化：（实验组号 1234）感觉网络质量不太好



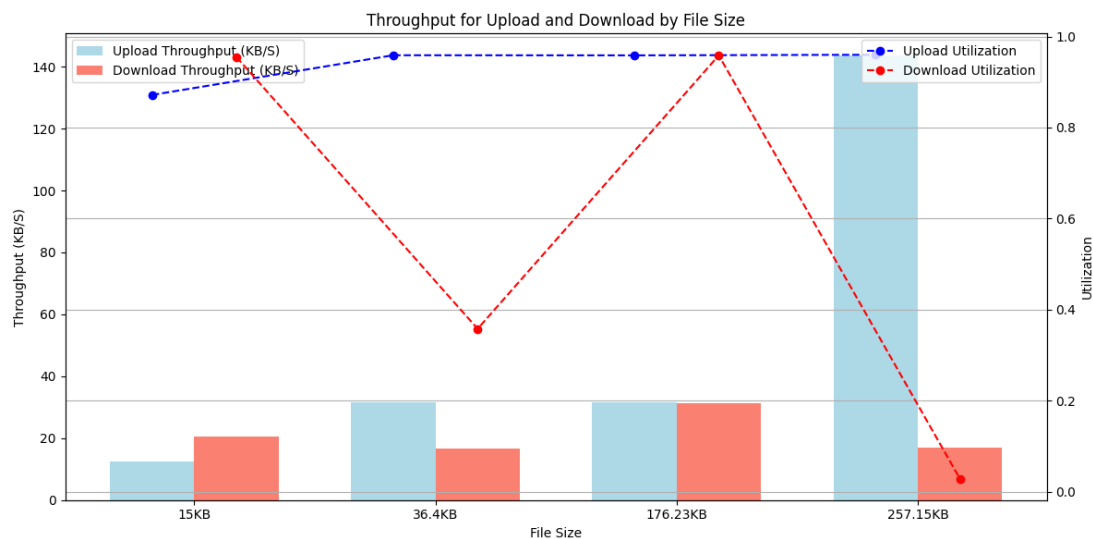


第4个点的 efficiency（橙色）应该是误差，趋势应该是越来越高。

Scenario 4: Throughput and Efficiency vs Packet Loss Rate (Different Congestion Control)



由 18-20 三组数据进行图像绘制：



基本上文件越大，上传的 throughput 越大，utilization 越大。而下载方面由于网络极不稳定，难以看出规律。算作是实验误差。

附录一些实验截图

```

Terminal: Ubuntu-22.04 - Ubuntu-22.04 (2) - Ubuntu-22.04 (3) + v
0 0 1 10000
10 5 6 10000
20 13 13 10000
Received ACK: 20
Packet loss rate: 0.0
传输总时间: 0.04304766654968262 s
有效吞吐量 (文件大小 / 传输时间): 825.7561407881255 KB/s
流量利用率 (文件大小 / 发送的总数据量): 0.958803076598832
End of transmission.
Connection closed.
f538d4e9e14911ebb4f6b50904fa3952 33.jpg

10 6 7 1
20 13 13 1
Received ACK: 20
Packet loss rate: 0.03571428571428571
传输总时间: 0.04606819152832031 s
有效吞吐量 (文件大小 / 传输时间): 771.6142922204281 KB/s
流量利用率 (文件大小 / 发送的总数据量): 0.9233422961797981
End of transmission.
Connection closed.
f538d4e9e14911ebb4f6b50904fa3952 33.jpg
    
```



```

Received ACK: 160
传输总时间: 0.11904072761535645 s
有效吞吐量 (文件大小 / 传输时间): 2109.555711776452 KB/s
流量利用率 (文件大小 / 发送的总数据量): 0.9345200023258519
Received ACK: 180
b166d1971b2546159252c416e9db9a43 11.jpg
End of transmission.
Connection closed.

```

超时重传，实际包丢失率为 0.15625

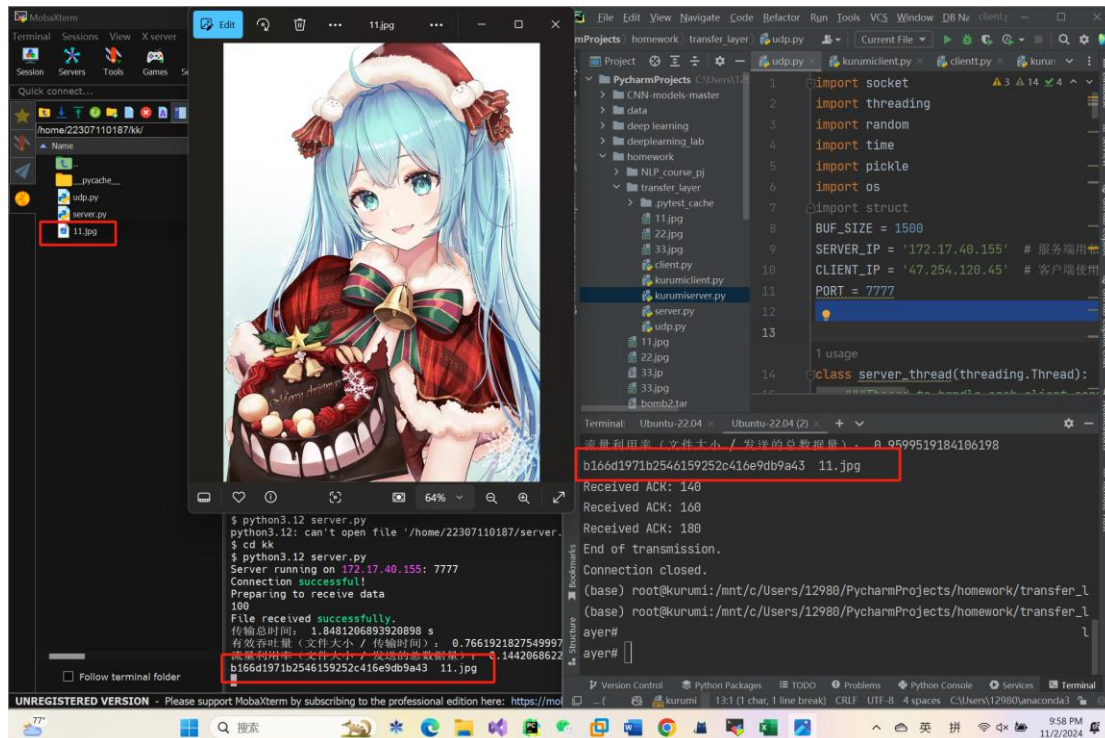
```

ubuntu@ubuntu: ~/Desktop
0 0 1 10000
Timeout, resending packets
Timeout, resending packets
Timeout, resending packets
Resending packet with seq 4
Timeout, resending packets
Timeout, resending packets
Timeout, resending packets
10 10 4 1
Timeout, resending packets
Resending packet with seq 12
Resending packet with seq 13
Resending packet with seq 14
20 17 5 3
Resending packet with seq 17
Received ACK: 20
Packet loss rate: 0.15625
传输总时间: 0.17436575889587402 s
有效吞吐量 (文件大小 / 传输时间): 203.86385047781957 KB/s
流量利用率 (文件大小 / 发送的总数据量): 0.8043487868475715
End of transmission.
Connection closed.
f538d4e9e14911ebb4f6b50904fa3952 33.jpg

```

第一次从本地传输东西到服务器:

传输 11.jpg，速度很快。Md5 值相等，流量利用率还是挺高的。



The screenshot shows a MobaXterm window with a file transfer in progress. The terminal output is as follows:

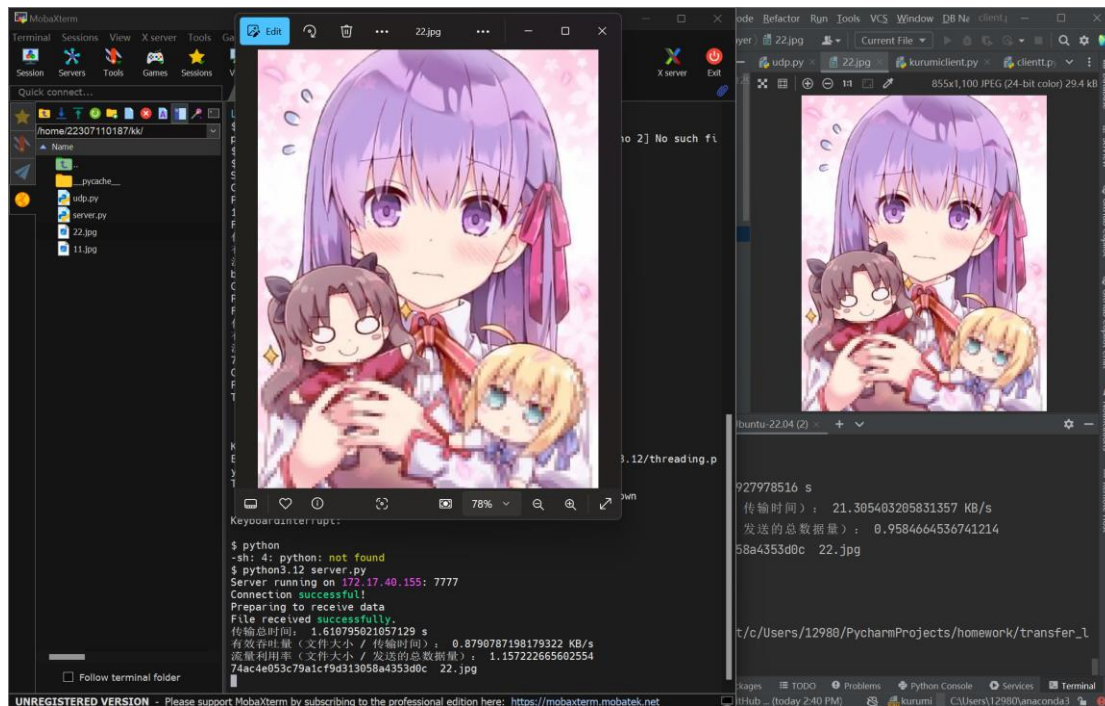
```

$ python3.12 server.py
python3.12: can't open file '/home/22307110187/server.py': [Errno 2] No such file or directory
$ cd kk
$ python3.12 server.py
Server running on 172.17.40.155: 7777
Connection successful!
Preparing to receive data
100
File received successfully.
传输总时间: 1.8481206893920898 s
有效吞吐量 (文件大小 / 传输时间): 0.7661921827549997 KB/s
流量利用率 (文件大小 / 发送的总数据量): 0.1442068622
b166d1971b2546159252c416e9db9a43 11.jpg
Received ACK: 140
Received ACK: 160
Received ACK: 180
End of transmission.
Connection closed.
(base) root@kurumi: /mnt/c/Users/12988/PycharmProjects/homework/transfer_1
(base) root@kurumi: /mnt/c/Users/12988/PycharmProjects/homework/transfer_1
ayer#
ayer#

```

The file transfer is successful, and the terminal shows the file name and MD5 hash: b166d1971b2546159252c416e9db9a43 11.jpg.

22.jpg



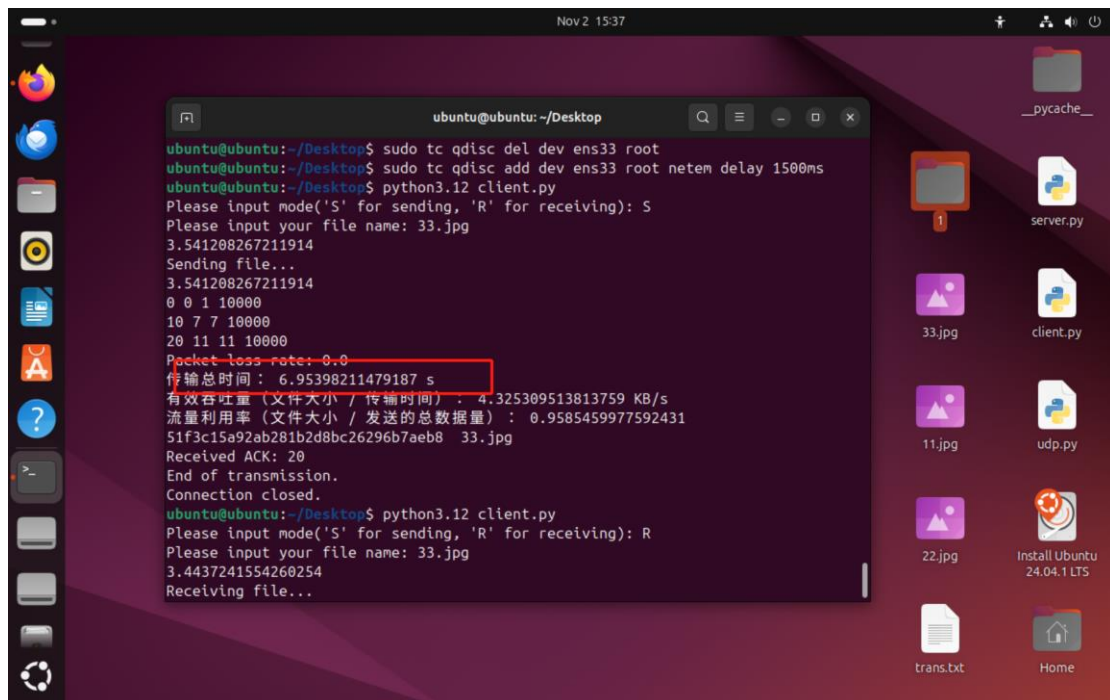
```

Resending 7 packets starting from seq 8
Resending 6 packets starting from seq 8
20 16 6 3
Resending 6 packets starting from seq 16
Resending 6 packets starting from seq 16
Resending 6 packets starting from seq 16
Resending 6 packets starting from seq 16
Packet loss rate: 0.6166666666666667
传输总时间: 4.758619070053101 s
有效吞吐量 (文件大小 / 传输时间): 6.3207675498312925 KB/s
流量利用率 (文件大小 / 发送的总数据量): 0.3578150049954692
Received ACK: 20
51f3c15a92ab281b2d8bc26296b7aeb8 33.jpg
Timeout, resending packets
Timeout, resending packets
Timeout, resending packets
End of transmission.
Connection closed.
Connection successful!
Preparing to receive data
File received successfully.
传输总时间: 8.67592453956604 s
有效吞吐量 (文件大小 / 传输时间): 0.16321207250505057 KB/s
流量利用率 (文件大小 / 发送的总数据量): 1.1094108645753635
51f3c15a92ab281b2d8bc26296b7aeb8 33.jpg
Connection successful!
Preparing to send data
3.44905948638916
0 0 1 10000
10 7 6 10000
20 14 12 10000
Packet loss rate: 0.0
传输总时间: 7.264065742492676 s
有效吞吐量 (文件大小 / 传输时间): 4.140673565776216 KB/s
流量利用率 (文件大小 / 发送的总数据量): 0.9584549977592431
51f3c15a92ab281b2d8bc26296b7aeb8 33.jpg
Received ACK: 20
End of transmission.
Connection closed.
Traceback (most recent call last):
  File "/home/22307110187/kk/server.py", line 32, in <module>
    packet, client_address = server_socket.recvfrom(BUF_SIZE)
KeyboardInterrupt

$
$

```

延迟设大一点一个 200k 的文件甚至要传输 7s。



The screenshot shows an Ubuntu desktop with a terminal window open. The terminal displays the following commands and output:

```
ubuntu@ubuntu: ~/Desktop
ubuntu@ubuntu:~/Desktop$ sudo tc qdisc del dev ens33 root
ubuntu@ubuntu:~/Desktop$ sudo tc qdisc add dev ens33 root netem delay 1500ms
ubuntu@ubuntu:~/Desktop$ python3.12 client.py
Please input mode('S' for sending, 'R' for receiving): S
Please input your file name: 33.jpg
3.541208267211914
Sending file...
3.541208267211914
0 0 1 10000
10 7 7 10000
20 11 11 10000
Packet loss rate: 0.0
传输总时间: 6.95398211479187 s
有效吞吐量 (文件大小 / 传输时间): 4.325309513813759 KB/s
流量利用率 (文件大小 / 发送的总数据量): 0.9585459977592431
51f3c15a92ab281b2d8bc26296b7aeb8 33.jpg
Received ACK: 20
End of transmission.
Connection closed.
ubuntu@ubuntu:~/Desktop$ python3.12 client.py
Please input mode('S' for sending, 'R' for receiving): R
Please input your file name: 33.jpg
3.4437241554260254
Receiving file...
```

The desktop environment includes a sidebar with application icons (Firefox, Telegram, Files, GNOME Software, LibreOffice, Dash, Help, Terminal, and a dock with various system icons) and a right sidebar with file icons (33.jpg, 11.jpg, 22.jpg, trans.txt, server.py, client.py, udp.py, and an Ubuntu installer icon).

丢包率设大一点重传次数爆炸增长，有时根本无法传输成功。