# **实验4报告模板**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 姓名： |  | 学号： |  | 班级： |  |
| 姓名： |  | 学号： |  | 班级： |  |
|  |  |  |  |  |  |

# **4 实验四 传输层的TCP协议分析**

## 4.1 实验目的

1. 理解TCP报文首部格式和字段的作用，TCP连接的建立和释放过程，TCP数据传输中的编号与确认（应答）的过程。能够使用工具分析TCP的应答机制。
2. 理解TCP的错误恢复的工作原理和字节流的传输模式，能够使用工具分析实际的TCP流中的错误恢复机制。
3. 理解TCP的流量控制的工作原理，能够使用工具分析TCP流的流量控制。
4. 理解TCP的拥塞控制的工作原理，能够使用工具分析TCP流的拥塞控制中的慢启动和快速重传过程。
5. 能够编写程序实现TCP的大数据量的传输。

## 4.2 实验内容及原理

1. 使用基于TCP的应用程序（比如浏览器下载文件）传输文件，在客户端和服务器均要捕获TCP报文。
2. 分析TCP报文首部信息、TCP连接的建立和释放过程、TCP数据的编号与确认机制。观察几个典型的TCP选项：MSS、SACK、Window Scale、Timestamp等，查资料说明其用途。
3. 观察和估计客户机到服务器的RTT，双方各自的MSS。
4. 观察TCP的流量控制过程，和拥塞控制中的慢启动、快速重传过程。
5. 编写程序实现TCP的大数据量的传输。

## 4.3 实验步骤

### 4.3.1 实验环境与分组

每**2**名同学一组，以现有校园网络环境及云服务器搭建网络，标注内网、公网地址。



### 4.3.2 TCP协议分析

步骤1：PC2登录到云服务器上，在云服务器上启动web服务器（Apache、IIS等），或者运行Python的简单WEB服务器。在服务器上放一个约1MB的文件。

步骤2：在PC1和云服务器上启动报文捕获软件，开始截获报文（注意从建立连接开始抓包）。注意：云服务器如果是Linux系统，可以用tcpdump来截获报文。

步骤3：在PC1上运行**实验三编写的HTTP**客户端程序，接收一个约1MB的文件。文件传输完成后，停止报文截获。

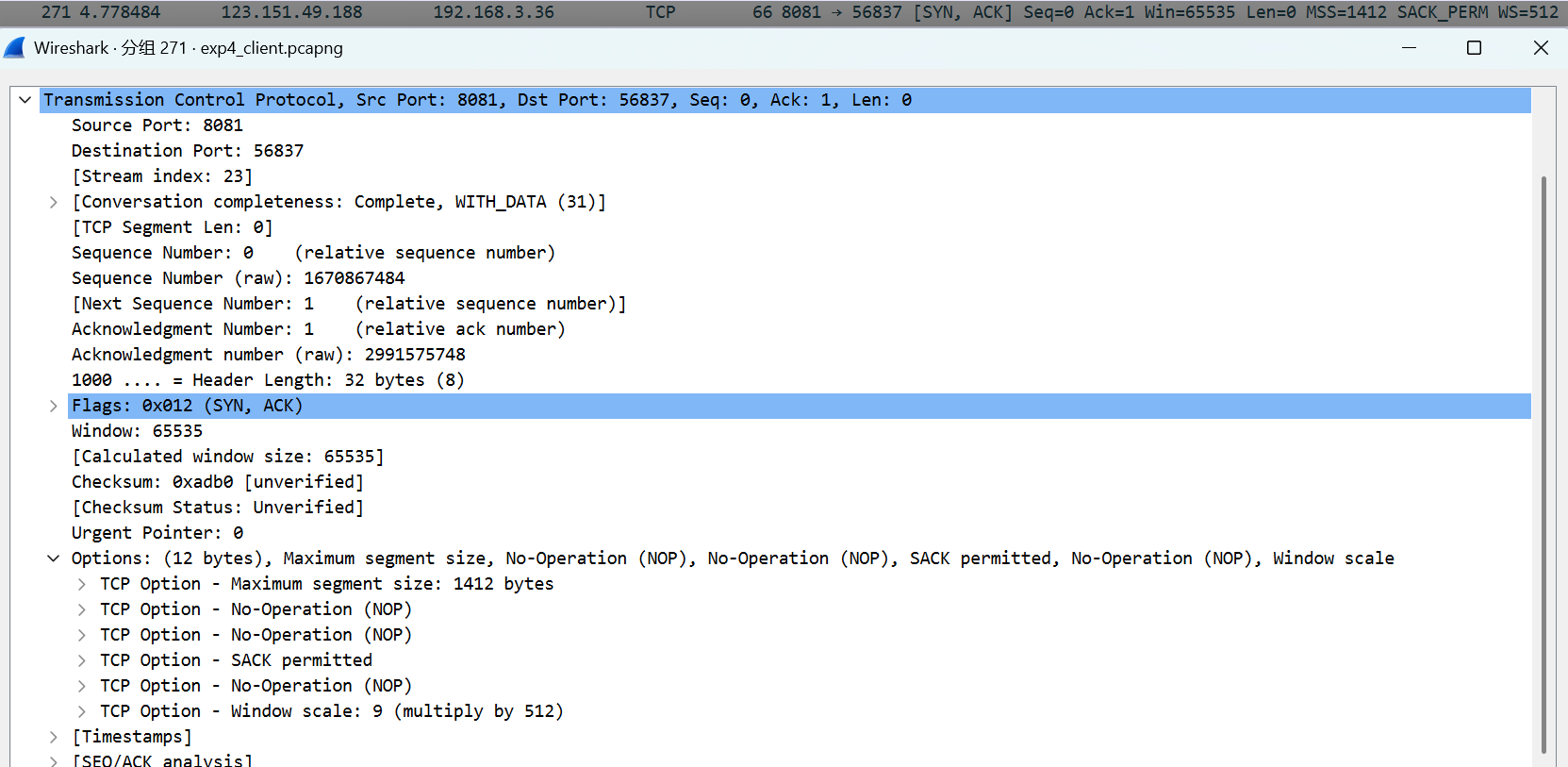
步骤4：对比观察PC1和云服务器截获的报文，分析TCP协议的选项：MSS、SACK、Window Scale、Timestamp等，查资料说明其用途。

1. MSS:MSS是TCP连接中允许的最大数据段大小（不包括TCP头部）。通过指定MSS，双方在建立连接时可以避免发送超过网络路径MTU（最大传输单元）的数据包，从而降低分片的可能性。

2. SACK :SACK允许接收方告知发送方哪些数据段已经成功接收，哪些未接收。在丢包情况下，SACK可以提高TCP的传输效率，允许发送方只重传丢失的数据段，而不是重传整个窗口中的数据。

3. Window Scale:Window Scale选项用于扩展TCP窗口大小的范围，允许更大的接收窗口以提高性能。在高延迟和高带宽的网络（如长距离连接）中，TCP的默认窗口大小可能不足以充分利用带宽。Window Scale通过乘以一个比例因子来增加窗口大小。

4. Timestamp:Timestamp选项用于记录数据包的发送和接收时间，以帮助解决TCP延迟和序列号的处理。通过在数据包中包含时间戳，接收方可以更准确地计算往返时间（RTT），并用于进行更有效的拥塞控制和时间戳回显。



步骤5：分析TCP数据传送阶段的报文，分析其**错误恢复**和**流量控制**机制，并填表。务必从建立连接开始抓包，捕获完整的TCP传输过程。【注：出现明显的流量控制的地方，Wireshark会有应答窗口的变化，乃至[TCP Window Full] 或[TCP Zero Window]标记的报文出现。如果没有观察到明显的流量控制过程，可以**修改HTTP客户端**程序，缓慢接收数据。】

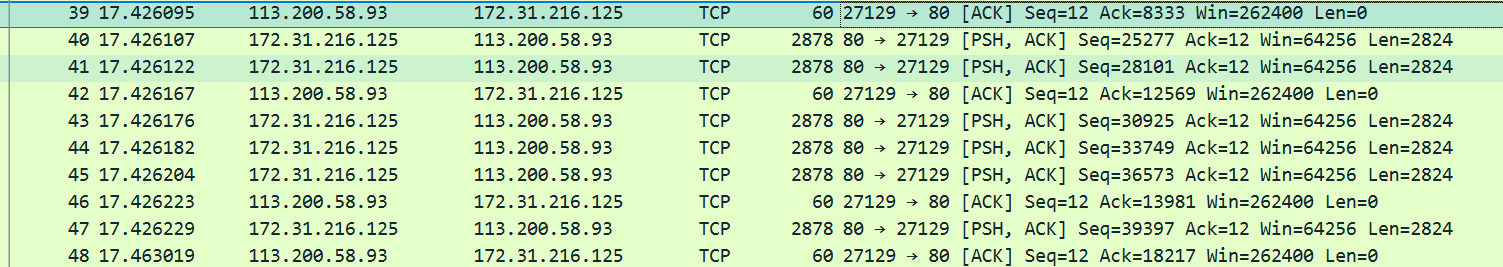
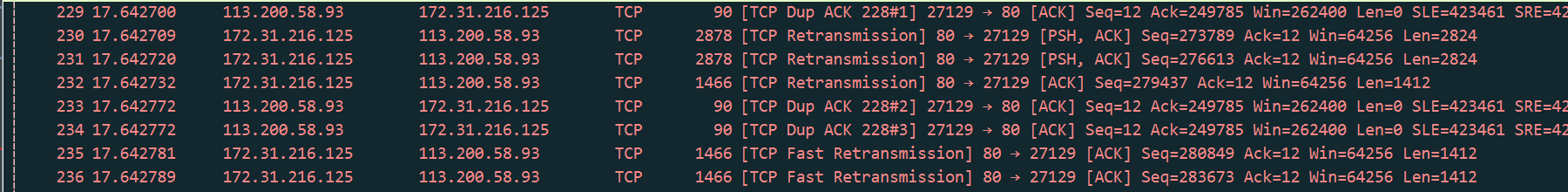


表4-1 记录TCP数据传送阶段的报文

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 报文号No. | 报文种类  (数据/确认) | 序号字段  Seq Number | 确认号  Ack Number | 窗口大小 | 数据长度 | 确认到哪条报文(填报文号) |
| 39 | 确认 | 12 | 8333 | 262400 | 0 | 30 |
| 40 | 数据 | 25277 | 12 | 64256 | 2824 |  |
| 41 | 数据 | 28101 | 12 | 64256 | 2824 |  |
| 42 | 确认 | 12 | 12569 | 262400 | 0 | 32 |
| 43 | 数据 | 30925 | 12 | 64256 | 2824 |  |
| 44 | 数据 | 33749 | 12 | 64256 | 2824 |  |
| 45 | 数据 | 36573 | 12 | 64256 | 2824 |  |
| 46 | 确认 | 12 | 13981 | 262400 | 0 | 32 |
| 47 | 数据 | 39397 | 12 | 64256 | 2824 |  |
| 48 | 确认 | 12 | 18217 | 262400 | 0 | 36 |

（1）错误恢复：



发现报文229，233，234: [TCP Dup ACK 228#1/2/3]

报文230，231，232: [TCP Retransmission]

报文235，236: [TCP Fast Retransmission]

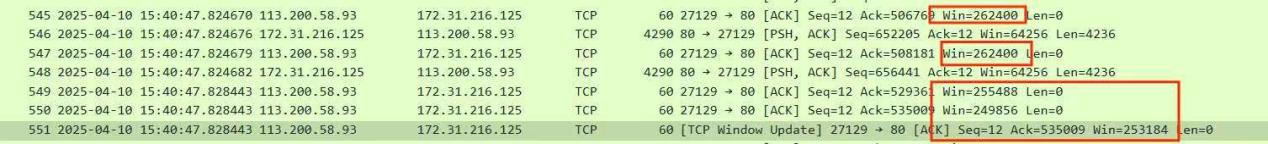
其中：

[TCP Dup ACK 228#1]: 重复确认，发生在接收方已经成功接收到一个数据包，但随后的预期数据包丢失或延迟到达。接收方继续发送对之前成功接收的数据包的确认（ACK），而不是等待丢失的数据包。其中#N表示第N次重复确认。

[TCP Retransmission]：重传报文，当发送方在没有收到预期的确认（ACK）时，会重新发送之前的数据包。这可能是由于数据包在传输过程中丢失、损坏或由于网络延迟导致确认（ACK）未能及时到达发送方。

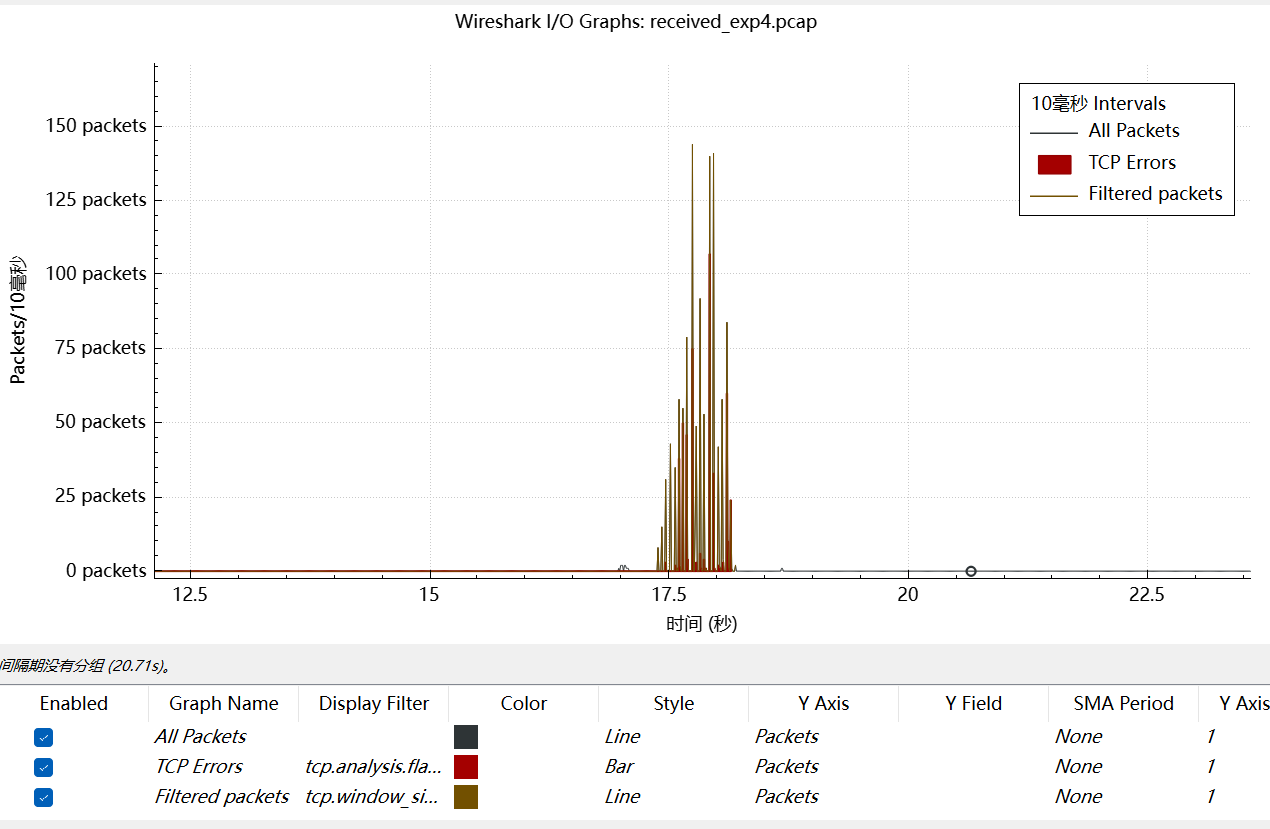
[TCP Fast Retransmission]：快重传。这是TCP协议中的一种机制，用于快速响应接收方的重复确认。当发送方在一定时间内（通常是三个重复确认）收到三个或更多的重复确认时，它会立即重传丢失的数据包，而不必等待重传计时器（Retransmission Timer）超时。

(2)流量控制



[TCP Window Update]: 表明接收方在该报文中更新了其接收窗口的大小，该动态调整机制为TCP流量控制的重要组成。

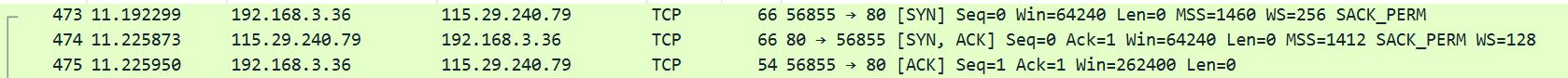
结合报文分析，549-551报文均对接收窗口大小进行了更新，窗口大小由262400-255488-249856-253184，窗口大小先减小后增大，可能是由于接收方进行了一些数据处理，导致接收速度减小，剩余缓存区减小，随后接收方接收速度提升，或是数据传输将完成，接收窗口大小逐渐恢复。



结合IO图也可以看到包传输过程存在折点，也可以粗略反应TCP流量控制的机制。

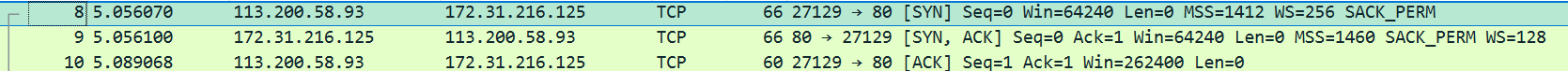
步骤6、分析客户机和服务器两边各自捕获到的分组，分析整个TCP流，估计双方的RTT和各自的MSS。

客户端



握手报文相隔时间估计RTT=11.225873s-11.192299s=33.574ms，MSS=1460

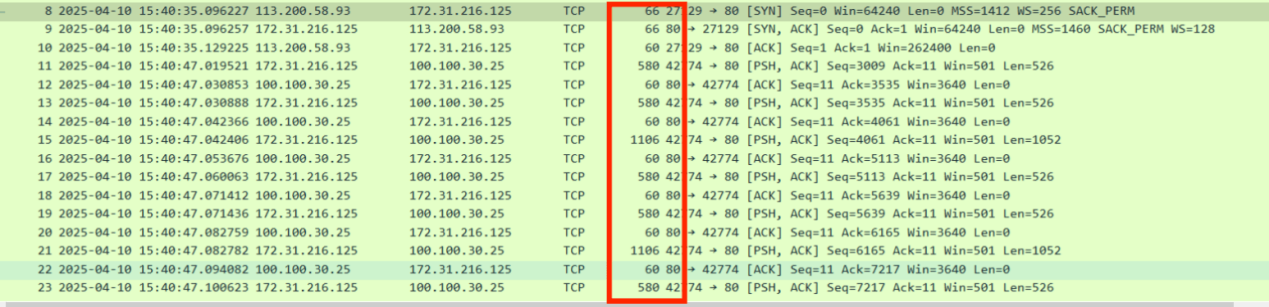
服务器

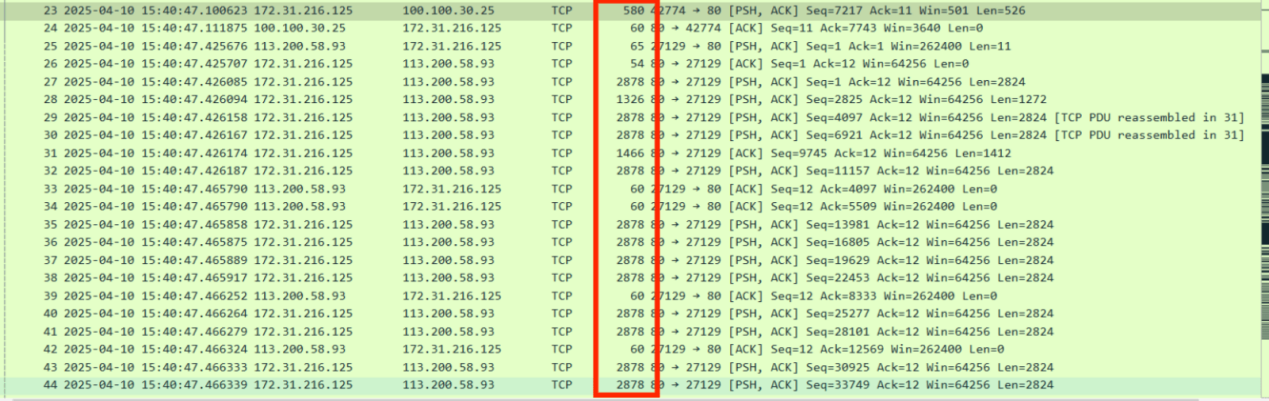


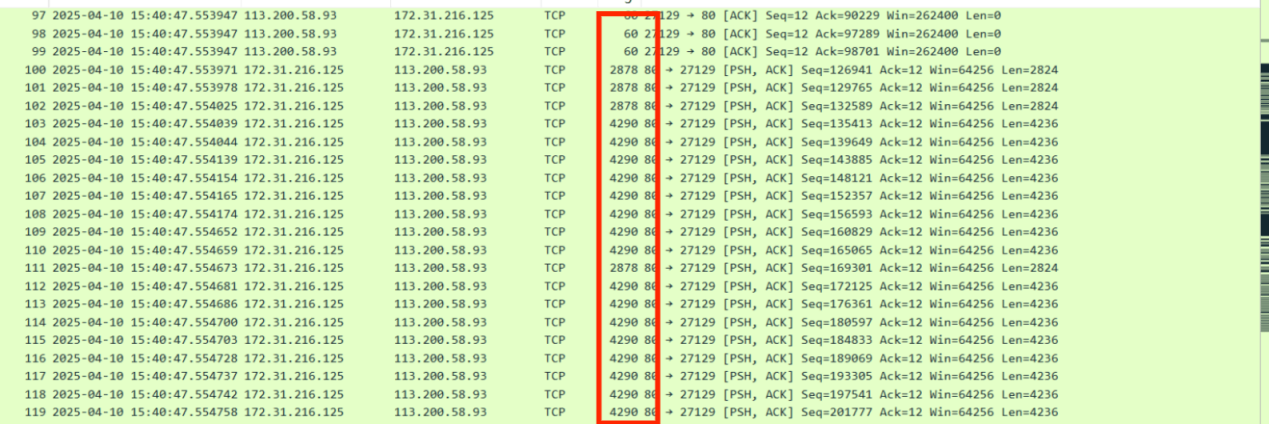
握手报文相隔时间估计RTT=5.089068s-5.056100s=32.968ms，MSS=1460

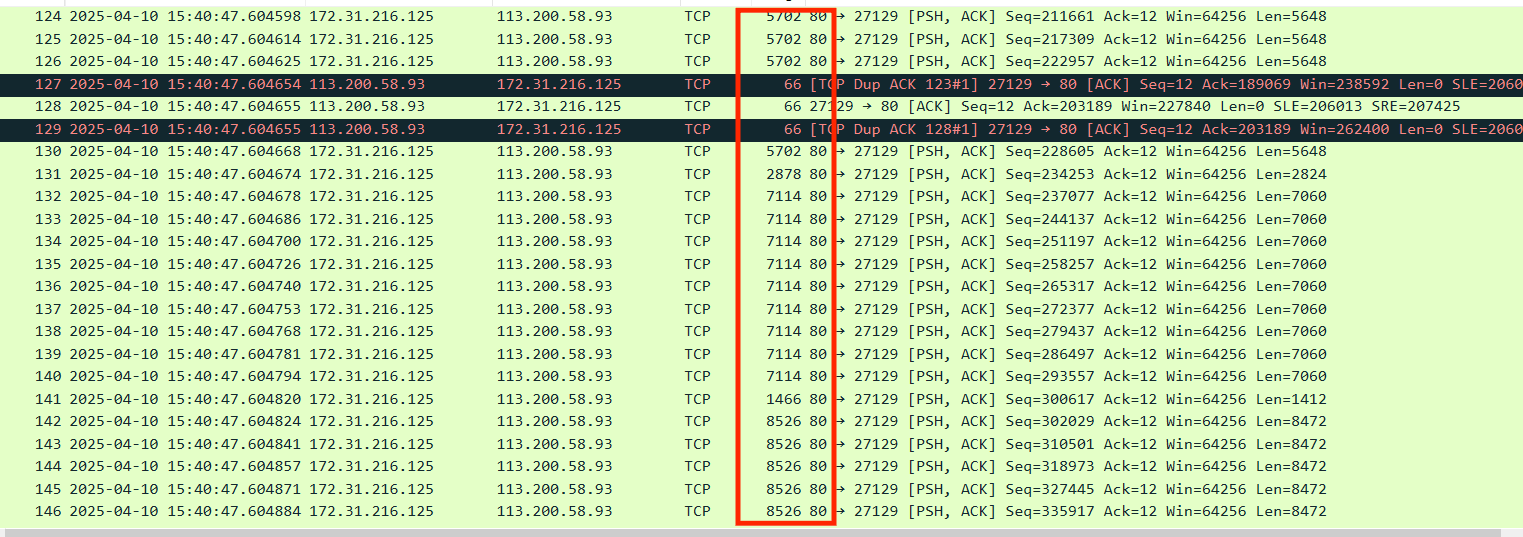
步骤7、分析整个TCP流的拥塞控制，找到拥塞控制的几个典型过程（慢启动、快速重传）。如果拥塞控制的相关过程不明显，请设计合适的方法再次测试。**【注：**注意观察特殊数据包：如[TCP Retransmission]、[TCP Dup ACK]、[TCP Fast Retransmission]等，结合前/后数据包，分析出现的原因及处理方法。**】**

慢启动：

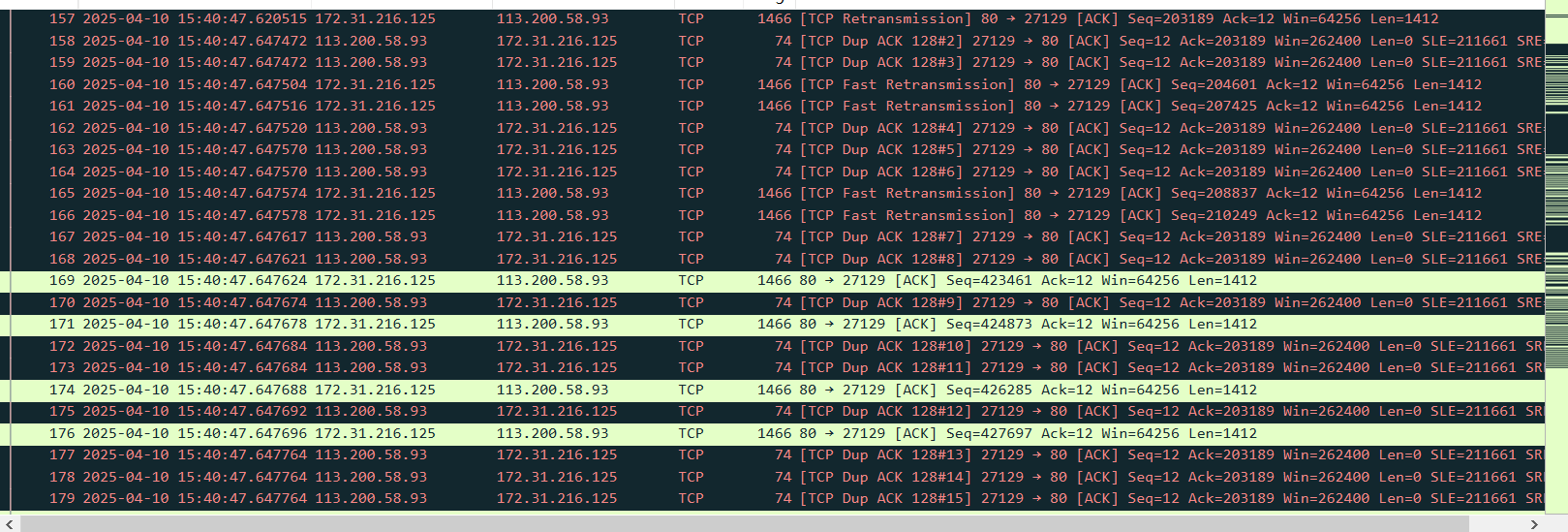








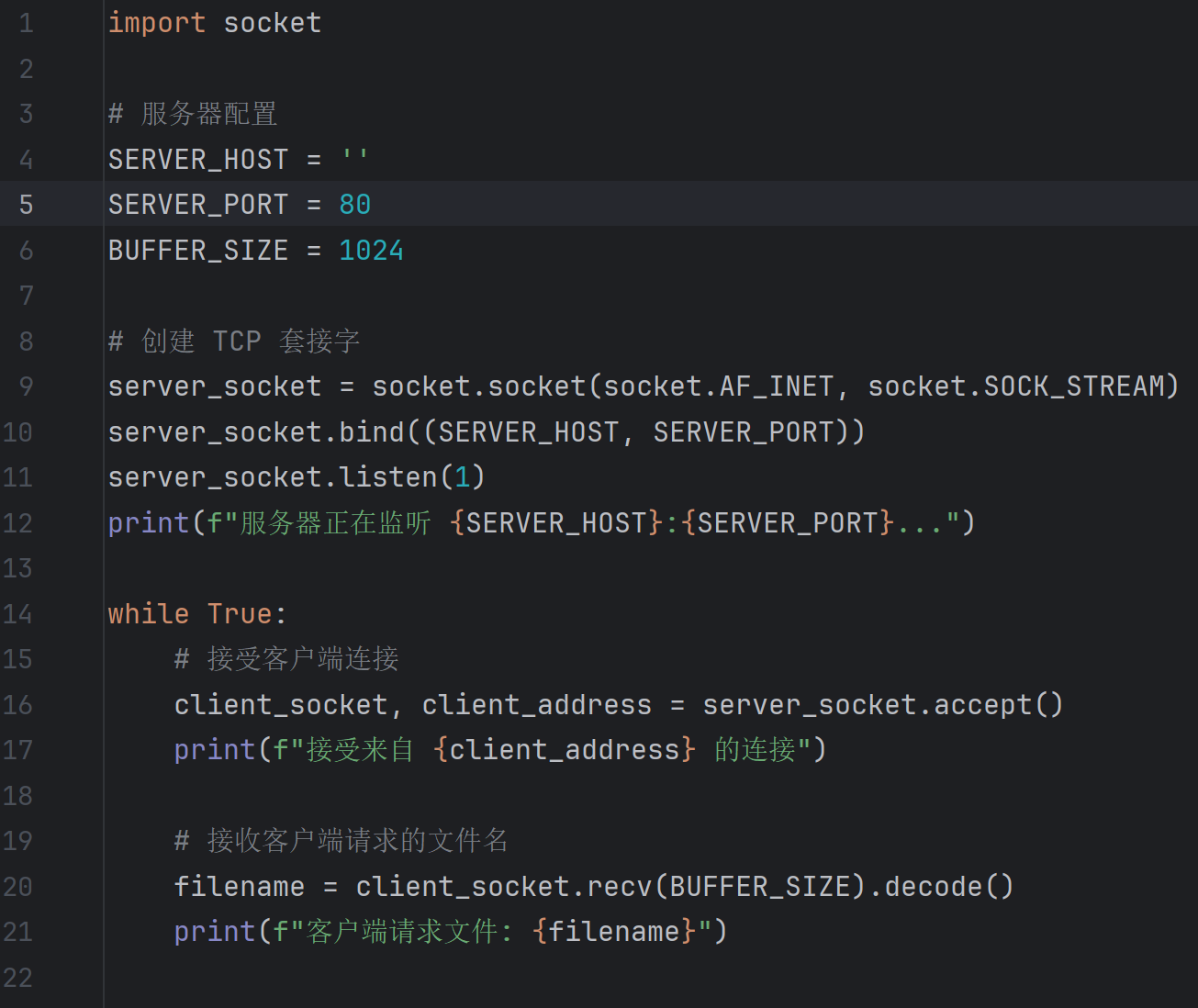
可以看到，拥塞出现以前，现在慢启动阶段发送方发送的报文长度成倍增加，到达一定阈值后不再指数增加，而是线性增加，也就是拥塞避免。一旦遇到拥塞：



发送方收到3个重复ACK，认为发生拥塞，迅速减小发送窗口，然后进行快重传。

### 4.3.3 TCP文件传输的实验过程及结果分析

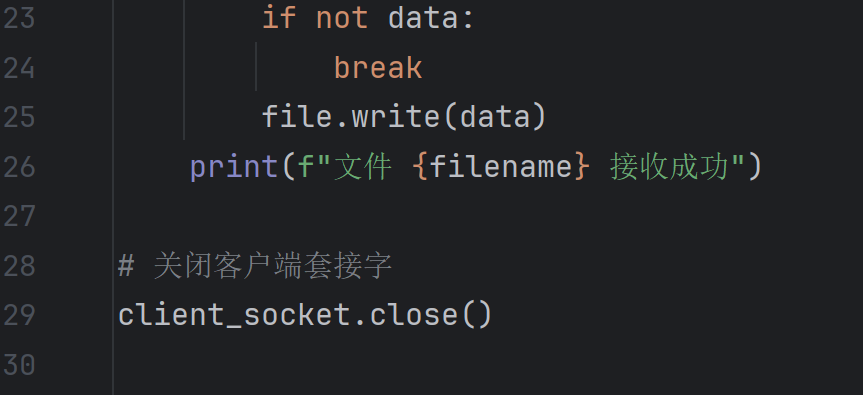
步骤1：编写基于TCP协议的文件传输服务器程序。该服务器程序能够在客户端请求文件时，打开对应的文件，将文件内容通过TCP协议传输给客户端。





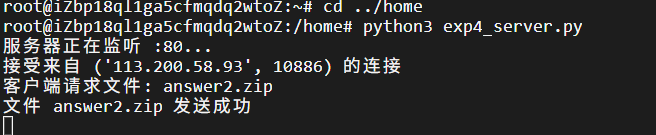
步骤2：编写基于TCP协议的文件传输客户端程序。该客户端程序能够连接到服务器，根据用户输入请求一个文件，并从服务器接收文件内容，保存为一个文件。

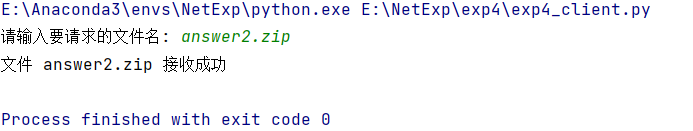


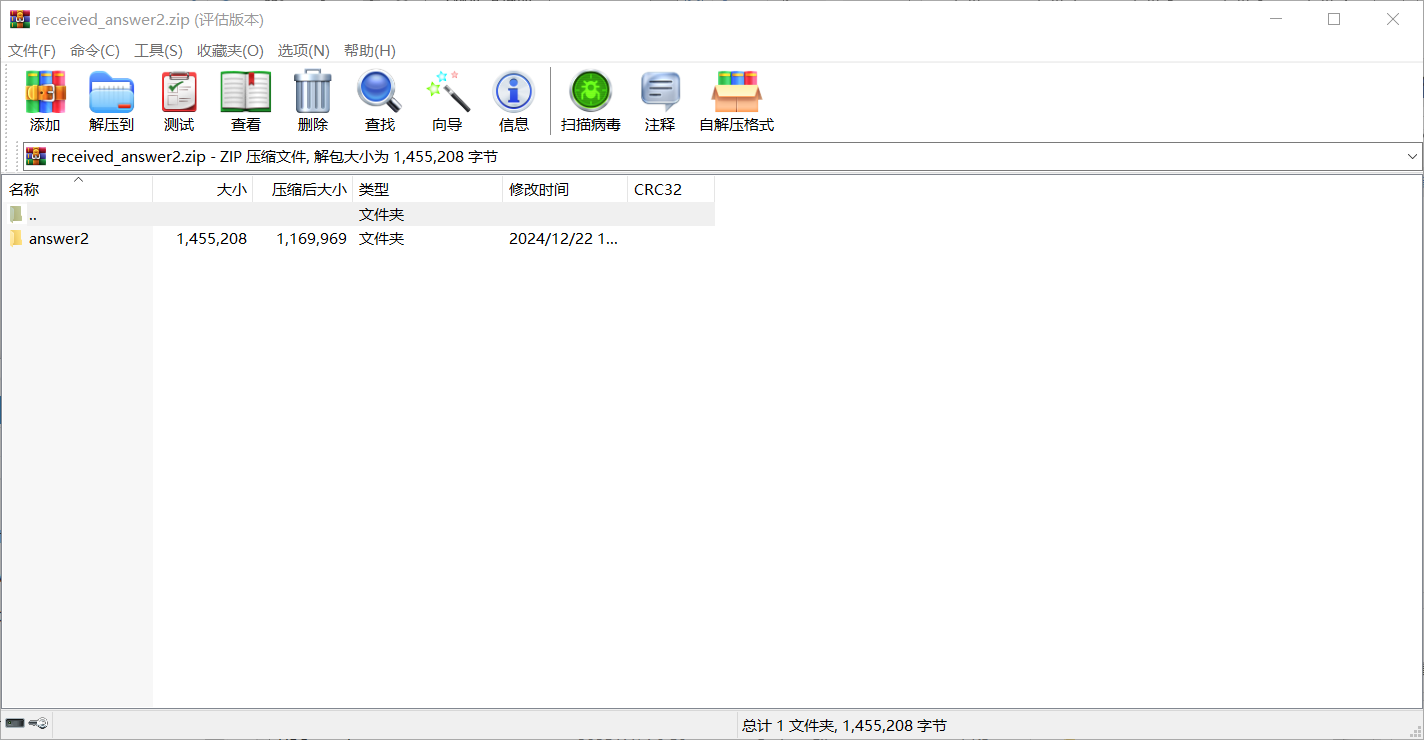


步骤3：建议将服务器部署在云服务器上，在本地机器上运行客户端程序，测试。

测试截图







文件传输无误

## 4.4 互动讨论主题

1）TCP的拥塞控制是哪一方（接收、发送）来主导的？拥塞控制的目的是什么？

TCP的拥塞控制是由发送方主导的。拥塞控制是TCP协议中的一个重要机制，其目的是防止过多的数据包同时在网络上传输，从而避免网络拥塞的发生。网络拥塞会导致数据包丢失、延迟增加，甚至可能导致网络瘫痪。

2）TCP的流量控制是哪一方（接收、发送）来主导的？流量控制的目的是什么？

CP的流量控制是由接收方主导的。流量控制的目的是防止发送方发送数据的速度过快，以至于接收方来不及处理，从而导致接收方的缓冲区溢出，数据丢失。

3）讨论传输层与其上下相邻层的关系，包括报文封装、解封、数据的传递、缓冲区的管理、可靠性的保证等。

传输层位于网络层之上，应用层之下。它负责端到端的通信，确保数据的完整性和可靠性。传输层与其上下相邻层的关系具体如下：

**与应用层的关系**

1. 报文封装：

传输层从应用层接收数据，将其封装成传输层的报文。

对于TCP，封装包括添加TCP头部，其中包含源端口、目的端口、序列号、确认号、数据偏移、保留位、标志位、窗口大小、校验和。

对于UDP，封装包括添加UDP头部，其中包含源端口、目的端口、长度和校验和。

2. 数据的传递：

应用层通过系统调用或API与传输层交互，发送或接收数据。

3. 缓冲区的管理：

发送缓冲区用于暂存待发送的数据，接收缓冲区用于暂存接收到的数据，直到应用层读取。

4. 可靠性的保证：

对于TCP，传输层提供可靠的数据传输服务，包括错误检测、重传机制、流量控制和拥塞控制。

对于UDP，由应用层自行处理可靠性问题。

**与网络层的关系**

1. 报文解封：

网络层将接收到的数据包传递给传输层，传输层负责解封，去除TCP或UDP头部，将剩余的数据传递给应用层。

2. 数据的传递：

传输层将封装好的报文传递给网络层，网络层负责将数据包通过网络路由到目的地。

网络层通过接口与传输层交互，接收和发送数据包。

3. 缓冲区的管理：

发送缓冲区：TCP会将未确认的数据暂存，直到收到ACK；UDP通常无发送缓冲区（直接交给网络层）。

接收缓冲区：TCP对乱序到达的数据排序后提交给应用层；UDP按到达顺序直接传递（无缓冲或简单缓冲）。

4. 可靠性的保证：

网络层主要负责数据包的路由和转发，不保证数据的可靠性。

传输层利用自己的机制（如重传、流量控制）来保证数据的可靠性。

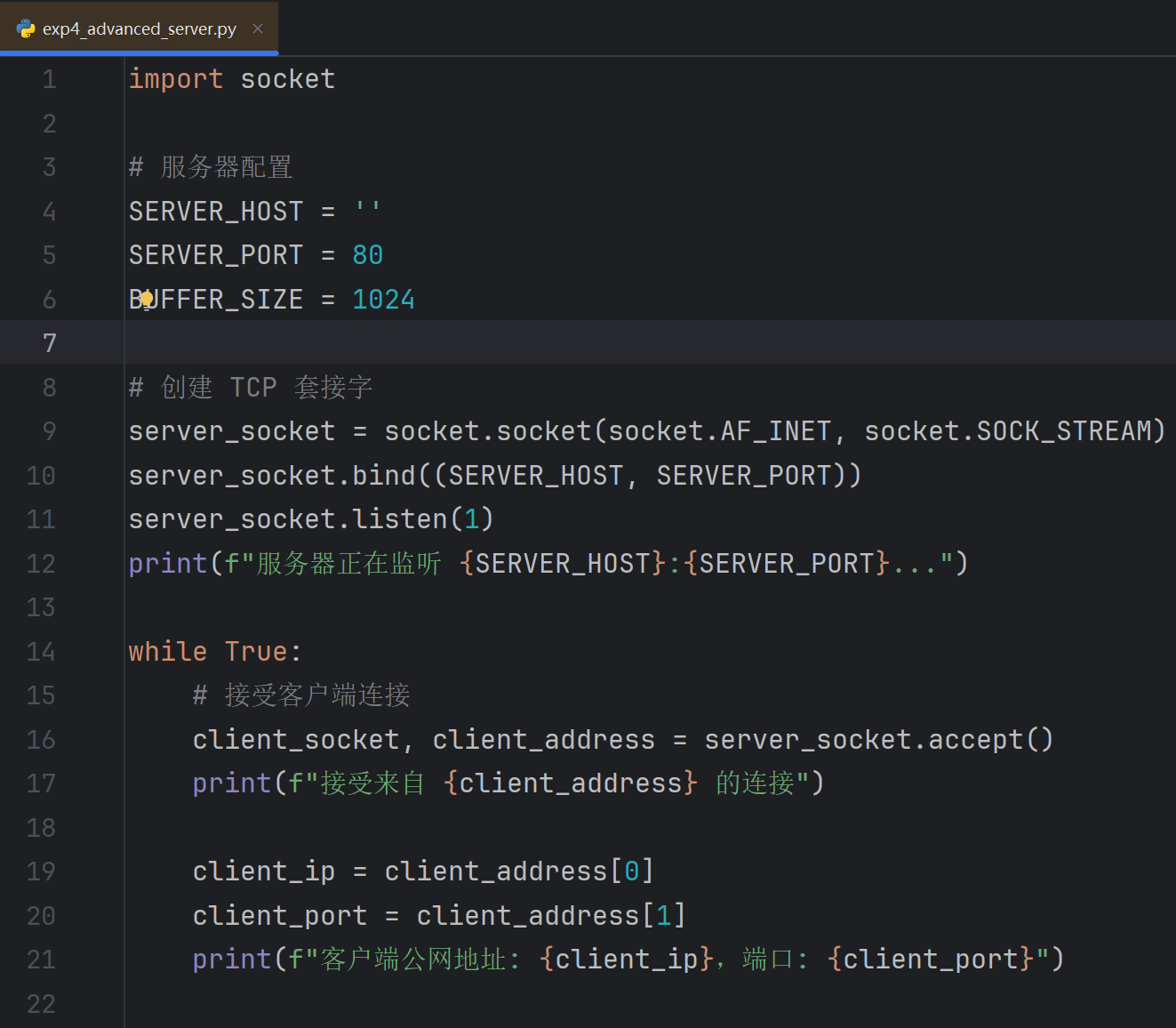
## 4.5 进阶自设计\*

1. 编写一个简单的TCP服务器程序，在云服务器上运行，告诉客户端的公网地址（及端口）和云服务器的私网地址。



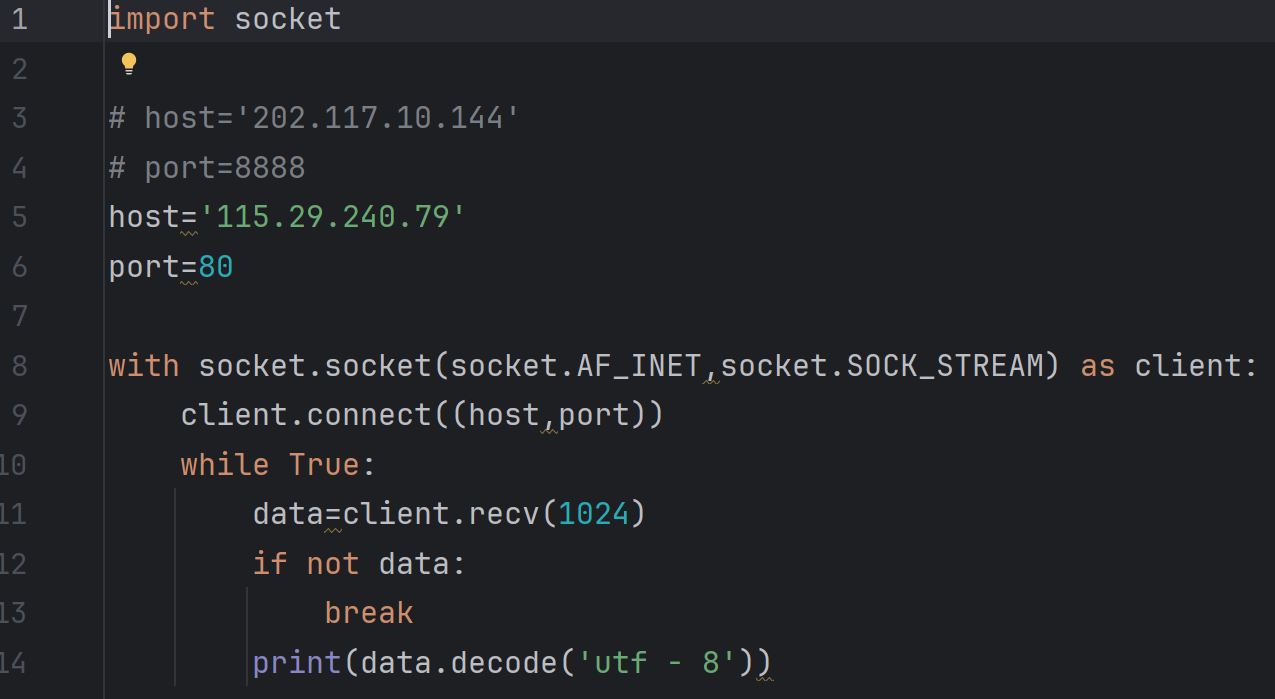
程序源码：

服务器





客户端：



2. 分析TCP的拥塞控制时，观察服务器的初始cwnd是多少（个MSS）。

终端输入ss -nli命令查看到服务器初始cwnd为10个MSS：

