

CS305 2023Spring Programing Assignment 2

12110304 徐春晖

Task 1:

脚本评分：85分

```
C:\Windows\system32\cmd.exe
Microsoft Windows [版本 10.0.19045.2846]
(c) Microsoft Corporation。保留所有权利。

(PA2) C:\Users\Xproever>d:

(PA2) D:\>cd "D:\学习文件\作业\计网\CS305-23S-Assignment2-main"

(PA2) D:\学习文件\作业\计网\CS305-23S-Assignment2-main>python 12110304_grader
RENAME THIS FILE TO your_sid
Test passed! packet_info
Test passed! http_stream_analyzer
Test passed! tcp_stream_analyzer
Test passed! tcp_stream_analyzer
TOKEN:2bf40b, HELLO! 12110304 your grade is 85

(PA2) D:\学习文件\作业\计网\CS305-23S-Assignment2-main>
```

Task 2:

功能展示:

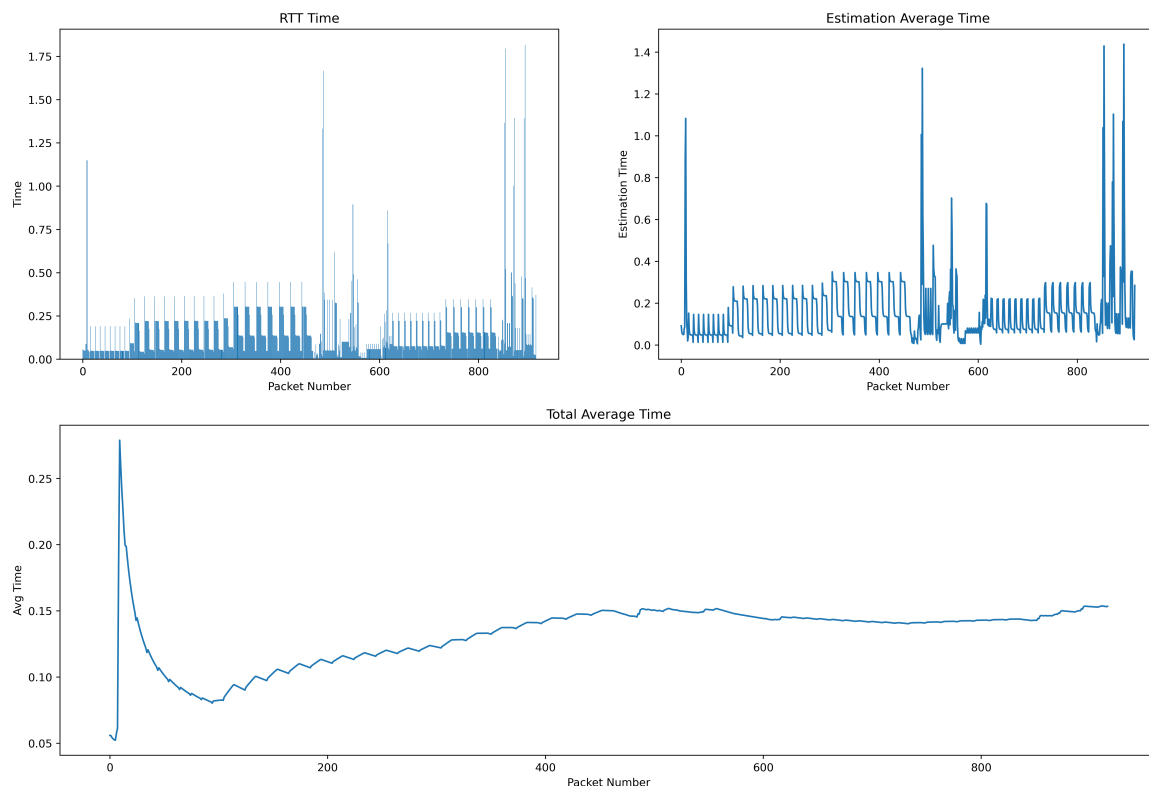
我测试的行为是 `RTT`，通过随意访问 GitHub 页面并进行抓包来判断同服务器的 RTT。

```
PS C:\Users\Xproever> ping github.com

正在 Ping github.com [20.205.243.166] 具有 32 字节的数据:
```

No.	Time	Source	Destination	Protocol	Length	Info
2173	11.000270	10.12.112.136	20.205.243.166	TCP	66	14351 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM...
2175	11.048636	20.205.243.166	10.12.112.136	TCP	66	443 → 14351 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1436 SACK_PERM...
2176	11.048839	10.12.112.136	20.205.243.166	TCP	54	14351 → 443 [ACK] Seq=1 Ack=1 Win=66048 Len=0
2177	11.049222	10.12.112.136	20.205.243.166	TCP	571	14351 → 443 [PSH, ACK] Seq=1 Ack=1 Win=66048 Len=517
2180	11.104706	20.205.243.166	10.12.112.136	TCP	1490	443 → 14351 [ACK] Seq=1 Ack=518 Win=67584 Len=1436
2181	11.104706	20.205.243.166	10.12.112.136	TCP	1431	443 → 14351 [PSH, ACK] Seq=1437 Ack=518 Win=67584 Len=1377

抓包文件为 `catch.pcapng`



其中，左上图为原始的每个包的 RTT 数据，右上图为 RTT 的加权累加平均值，类似于 TCP 协议中 RTT 估计值的算法，下图为 RTT 的无权累加平均值。

因为是在一段时间内连续访问抓下来的包，可以近似认为是服务端和客户端连续通信。个别 RTT 较高的值可能是因为多个访问之间出现间隔，将间隔也一并记录了，不过不影响整体计算。

从上图可以看出，平均 RTT 值逐渐稳定。日常使用场景下，本机同 GitHub 的 RTT 约为 **0.15s**（包括了服务器数据准备与传输的时间）

实现思路：

```

1 def read_ack(path, client_ip_prev, server_ip_prev, client_port_prev, server_port_prev):
2     stream_info = (client_ip_prev, server_ip_prev, client_port_prev, server_port_prev)
3     packets = rdpcap(path)
4     times = []
5     avg = []
6
7     for pkt in packets:
8         if not pkt.haslayer('TCP'):
9             continue
10
11         tcp = pkt.getlayer('TCP')
12         if pkt.haslayer('IP'):
13             ip = pkt.getlayer('IP')
14
15             if (ip.src, ip.dst, tcp.sport, tcp.dport) == stream_info: # 客户端发送
16                 times.append((tcp.ack, pkt.time)) # 添加记录
17
18             elif (ip.dst, ip.src, tcp.dport, tcp.sport) == stream_info: # 服务端送回
19                 for t in times:
20                     if tcp.seq == t[0]: # seq==ack
21                         avg.append(pkt.time - t[1]) # 记录差值

```

```
22
23     return avg
```

对于客户端发送给服务端的每一个 ACK，记录下 ACK 包发送时间，并在客户端接受来自服务端的相同值的 SEQ 时，记录两包的时间差值。

然后以 `avg` 中的值进行绘图，代码如下：

```
1  def draw_graph(ackss): # ackss 即为上函数返回的 avg
2      plt.figure(figsize=(18, 12), dpi=400)
3      x_values = list(range(len(ackss)))
4
5      ax1 = plt.subplot(2, 2, 1)
6      ax2 = plt.subplot(2, 1, 2)
7      ax3 = plt.subplot(2, 2, 2)
8
9      """
10     ax1: 原始数据导出，根据以上得出的 avg 绘图
11     """
12     plt.sca(ax1)
13     plt.bar(x_values, ackss)
14     plt.xlabel('Packet Number')
15     plt.ylabel('Time')
16     plt.title('RTT Time')
17     """
18     ax3: 根据原始数据计算总平均绘图
19     """
20     cnt = 0
21     tt = 0
22     av = [] # 纵轴值集合
23     for a in ackss:
24         cnt = cnt + 1
25         tt = tt + a
26         av.append(tt / cnt) # 总平均时间
27
28     plt.sca(ax2)
29     x_values = list(range(len(av)))
30     plt.plot(x_values, av)
31     plt.xlabel('Packet Number')
32     plt.ylabel('Avg Time')
33     plt.title('Total Average Time')
34     """
35     ax2: 加权平均绘图
36     """
37     est = [] # 纵轴值集合
38     e = 0.2
39     for a in ackss:
40         e = 0.25 * e + 0.75 * a # 加权平均
41         est.append(e)
42
43     plt.sca(ax3)
44     x_values = list(range(len(est)))
45     plt.plot(x_values, est)
46     plt.xlabel('Packet Number')
47     plt.ylabel('Estimation Time')
48     plt.title('Estimation Average Time')
49
```

```
50 # Plot 展示
51 plt.show()
```