通信与网络实验二：传输层 TCP 协议实验报告

无 06
闫珺博
2020010796

一、网络仿真环境运行和实验网络搭建

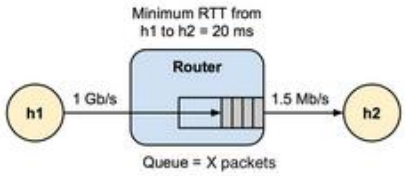
jupyter Exp2_TCP Last Checkpoint: 10/05/2022 (autosaved)

File Edit View Insert Cell Kernel Help

Run Markdown

实验2:传输层TCP协议实验

5.1 网络仿真环境运行和实验网络搭建



In [1]:

```
from mininet.topo import Topo
from mininet.node import CPULimitedHost, OVSController
from mininet.link import TCLink
from mininet.net import Mininet
from mininet.log import lg, info
from mininet.util import dumpNodeConnections

class BBTopo(Topo):
    "Simple topology for bufferbloat experiment."

    def __init__(self, queue_size):
        super(BBTopo, self).__init__()

        # Create router s0 (还要不区分交换机和路由器,统一用addSwitch命令添加)
        s0 = self.addSwitch('s0')

        # Create two hosts with names "h1" and "h2"
        h1 = self.addHost('h1')
        h2 = self.addHost('h2')

        # Add links with appropriate bandwidth, delay, and queue size parameters.
        # Set the router queue size using the queue_size argument
        # Set bandwidths/latencies using the bandwidths and minimum RTT given in the network diagram above
        self.addLink(h1, s0, bw=1000, delay='10ms', max_queue_size=queue_size)
        self.addLink(h2, s0, bw=1.5, delay='10ms', max_queue_size=queue_size)
        return

import os
# Set the cwnd control algorithm to "reno"
os.system("sysctl -w net.ipv4.tcp_congestion_control=reno")
# create the topology with queue_size=10
topo = BBTopo(queue_size=10)

# validate the built topology
print('构建网络中包含的节点:')
print(topo.nodes())
print('构建网络中包含的链路:')
for i, link in enumerate(topo.links()):
    print('第%d条链路:' % (i+1))
    print(link)
    print('第%d条链路信息:' % (i+1))
    print(topo.linkInfo(link[0], link[1]))


from subprocess import call
# Clean mininet files
call(["mn", "-c"])
# Create the network
net = Mininet(topo=topo, host=CPULimitedHost, link=TCLink, controller= OVSController)
net.start()
# Print the network topology
dumpNodeConnections(net.hosts)
# Performs a basic all pairs ping test to ensure the network set up properly
print("\ntest all pair pings")
print("packet loss percentage: %f" % net.pingAll())
# Print the IP of hosts
for host in net.hosts:
    print(host.name, host.IP())
```

运行上述代码，实现网络拓扑搭建和 Mininet 实例化

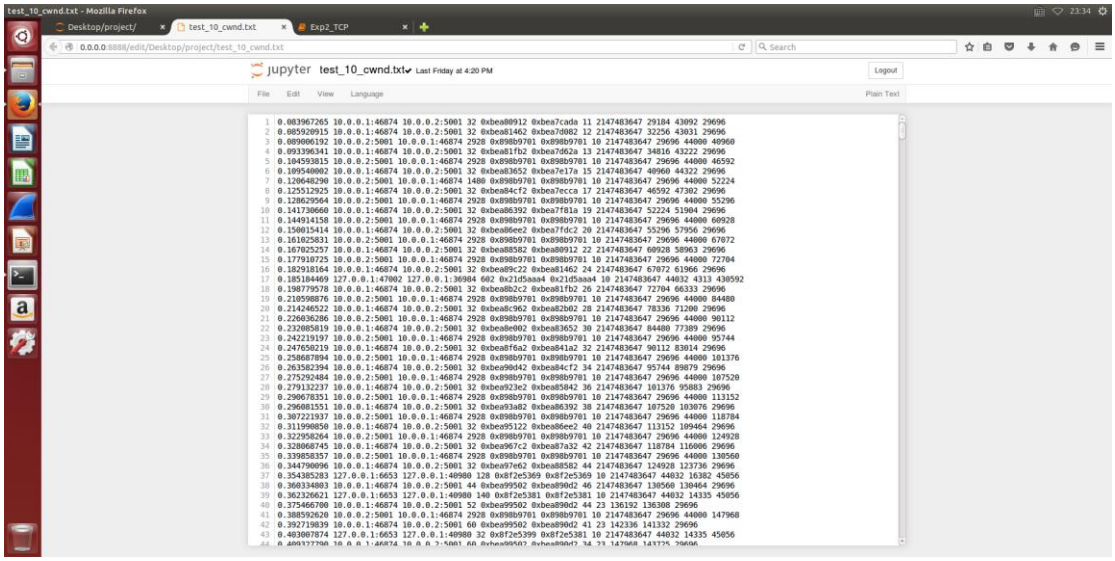
```
构建网络中包含的节点:
['h1', 'h2', 's0']
构建网络中包含的链路:
第1条链路:
('h2', 's0')
第1条链路信息:
{'delay': '10ms', 'bw': 1.5, 'max_queue_size': 10, 'node1': 'h2', 'node2': 's0', 'port2': 2, 'port1': 0}
第2条链路:
('h1', 's0')
第2条链路信息:
{'delay': '10ms', 'bw': 1000, 'max_queue_size': 10, 'node1': 'h1', 'node2': 's0', 'port2': 1, 'port1': 0}
h1 h1-eth0:s0-eth1
h2 h2-eth0:s0-eth2
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)

test all pair pings
packet loss percentage: 0.000000
('h1', '10.0.0.1')
('h2', '10.0.0.2')
```

代码运行结果，成功建立了 h1 到路由器、路由器到 h2 的网络结构。利用 ping 测试 h1 和 h2 双向连接都能正常工作。

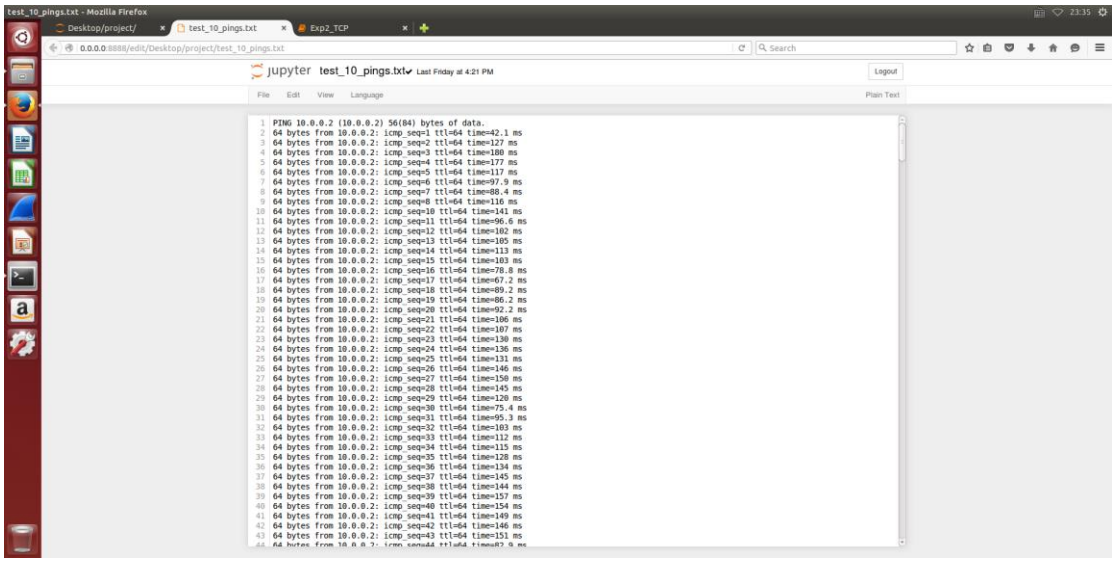
二、TCP 流量产生和数据包抓取

```
Starting iperf server
Starting iperf client
Starting ping...
simulation finished
```

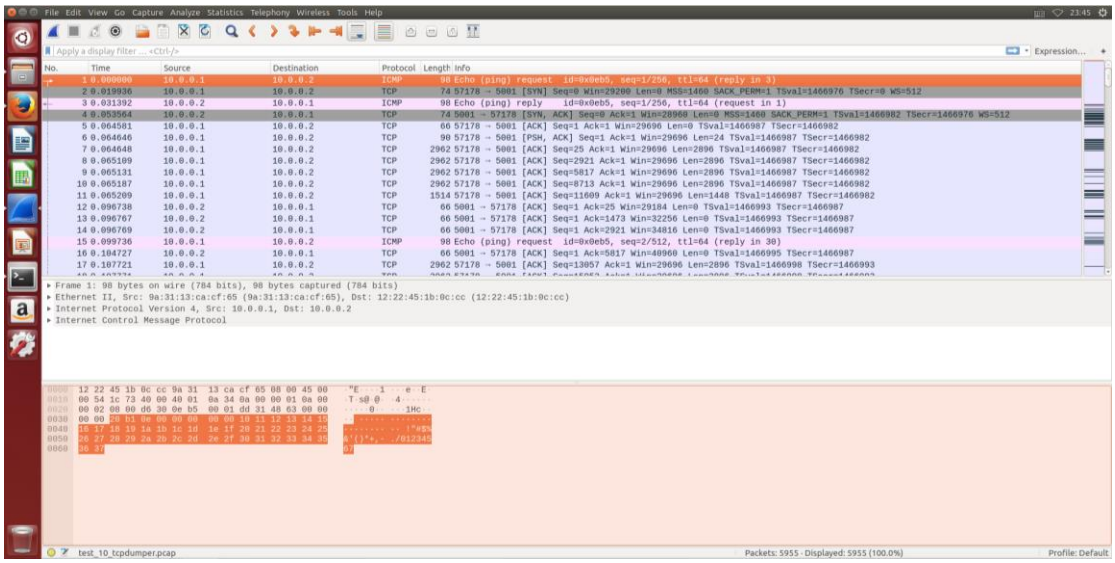
运行代码块，得到记录 tcpdump 抓包结果的 test_10_tcpdumper.pcap、记录 RTT 和拥塞窗口等指标的 test_10_cwnd.txt 和测量样本 RTT 的 test_10_pings.txt



test_10_cwnd.txt

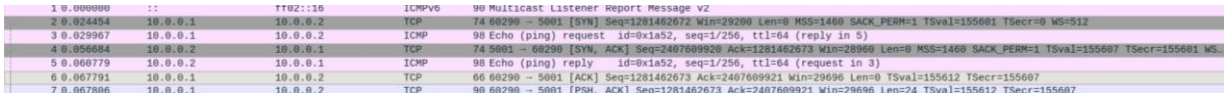


test_10_pings.txt

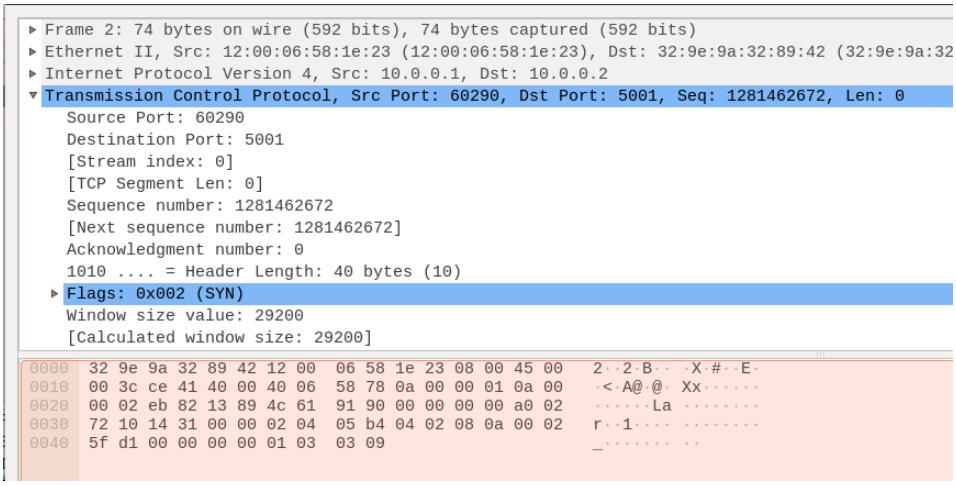


test_10_tcpdumper.pcap

三、TCP 连接管理

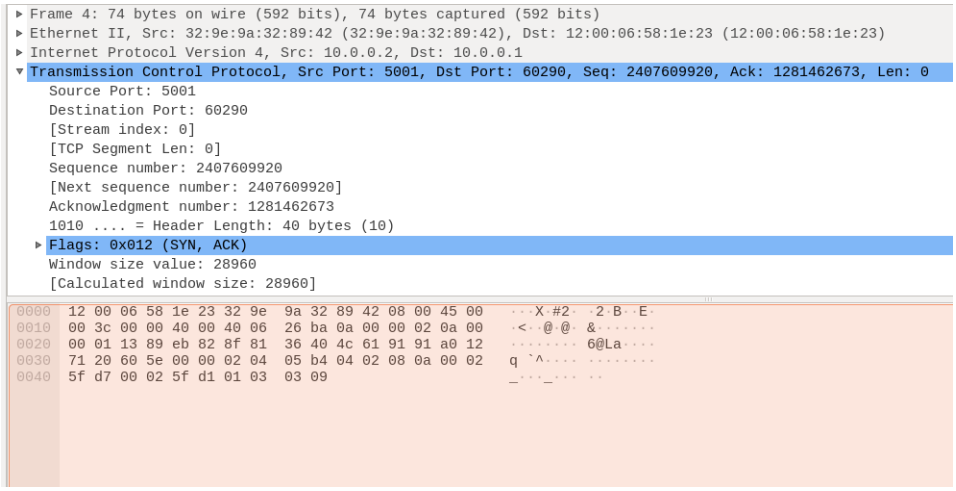


图中三条加深的数据包为三次握手过程



第一次握手

源端口号：60290								目的端口号：5001
序号：1281462672								
确认号：0								
首部长度	保留位用	URG	ACK	PSH	RST	SYN	FIN	



第二次握手

源端口号：5001								目的端口号：60290
序号：2407609920								
确认号：1281462673								
首部长度	保留位用	URG	ACK	PSH	RST	SYN	FIN	

```
▸ Frame 6: 66 bytes on wire (528 bits), 66 bytes captured (528 bits)
▸ Ethernet II, Src: 12:00:06:58:1e:23 (12:00:06:58:1e:23), Dst: 32:9e:9a:32:89:42 (32:9e:9a:32:89:42)
▸ Internet Protocol Version 4, Src: 10.0.0.1, Dst: 10.0.0.2
▾ Transmission Control Protocol, Src Port: 60290, Dst Port: 5001, Seq: 1281462673, Ack: 2407609921, Len: 0
    Source Port: 60290
    Destination Port: 5001
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence number: 1281462673
    [Next sequence number: 1281462673]
    Acknowledgment number: 2407609921
    1000 .... = Header Length: 32 bytes (8)
  ▸ Flags: 0x010 (ACK)
    Window size value: 58
    [Calculated window size: 29696]
```

```
0000  32 9e 9a 32 89 42 12 00  06 58 1e 23 08 00 45 00   2··2·B·· ·X·#··E·
0010  00 34 ce 42 40 00 40 06  58 7f 0a 00 00 01 0a 00   ·4·B@·@· X·······
0020  00 02 eb 82 13 89 4c 61  91 91 8f 81 36 41 80 10   ·····La ····6A··
0030  00 3a 14 29 00 00 01 01  08 0a 00 02 5f dc 00 02   ·:·)···· ····_···
0040  5f d7                                             _·
```

第三次握手

源端口号：60290								目的端口号：5001
序号：1281462672								
确认号：2407609921								
首部长度	保留位用	URG	ACK	PSH	RST	SYN	FIN	

```
5547 32.363208   10.0.0.1   10.0.0.2   TCP    2746 60290 → 5001 [FIN, PSH, ACK] Seq=1287227185 Ack=2407609921 Win=29696 Len=2680 TSval=163686 TSecr=163681
5548 32.369020   10.0.0.2   10.0.0.1   TCP      66 5001 → 60290 [ACK] Seq=2407609921 Ack=1287215601 Win=11534848 Len=0 TSval=163685 TSecr=163668
5549 32.377186   10.0.0.2   10.0.0.1   TCP      66 5001 → 60290 [ACK] Seq=2407609921 Ack=1287217049 Win=11537920 Len=0 TSval=163687 TSecr=163670
5550 32.385949   10.0.0.2   10.0.0.1   TCP      66 5001 → 60290 [ACK] Seq=2407609921 Ack=1287218497 Win=11540992 Len=0 TSval=163689 TSecr=163672
5551 32.393592   10.0.0.2   10.0.0.1   TCP      66 5001 → 60290 [ACK] Seq=2407609921 Ack=1287219945 Win=11543552 Len=0 TSval=163691 TSecr=163674
5552 32.400867   10.0.0.2   10.0.0.1   TCP      66 5001 → 60290 [ACK] Seq=2407609921 Ack=1287221393 Win=11546624 Len=0 TSval=163693 TSecr=163676
5553 32.410282   10.0.0.2   10.0.0.1   ICMP     98 Echo (ping) reply    id=0x1a52, seq=319/16129, ttl=64 (request in 5541)
5554 32.410286   10.0.0.2   10.0.0.1   TCP      66 5001 → 60290 [ACK] Seq=2407609921 Ack=1287222841 Win=11549696 Len=0 TSval=163695 TSecr=163679
5555 32.418348   10.0.0.2   10.0.0.1   TCP      66 5001 → 60290 [ACK] Seq=2407609921 Ack=1287224289 Win=11552256 Len=0 TSval=163697 TSecr=163680
5556 32.426869   10.0.0.2   10.0.0.1   TCP      66 5001 → 60290 [ACK] Seq=2407609921 Ack=1287227185 Win=11558400 Len=0 TSval=163699 TSecr=163682
5557 32.428839   10.0.0.1   10.0.0.2   ICMP     98 Echo (ping) request  id=0x1a52, seq=320/16385, ttl=64 (reply in 5560)
5558 32.441343   10.0.0.2   10.0.0.1   TCP      66 5001 → 60290 [ACK] Seq=2407609921 Ack=1287229866 Win=11563520 Len=0 TSval=163703 TSecr=163686
5559 32.450240   10.0.0.2   10.0.0.1   TCP      66 5001 → 60290 [FIN, ACK] Seq=2407609921 Ack=1287229866 Win=11563520 Len=0 TSval=163705 TSecr=163686
5560 32.460605   10.0.0.2   10.0.0.1   ICMP     98 Echo (ping) reply    id=0x1a52, seq=320/16385, ttl=64 (request in 5557)
5561 32.460611   10.0.0.1   10.0.0.2   TCP      66 60290 → 5001 [ACK] Seq=1287229866 Ack=2407609922 Win=29696 Len=0 TSval=163711 TSecr=163705
```

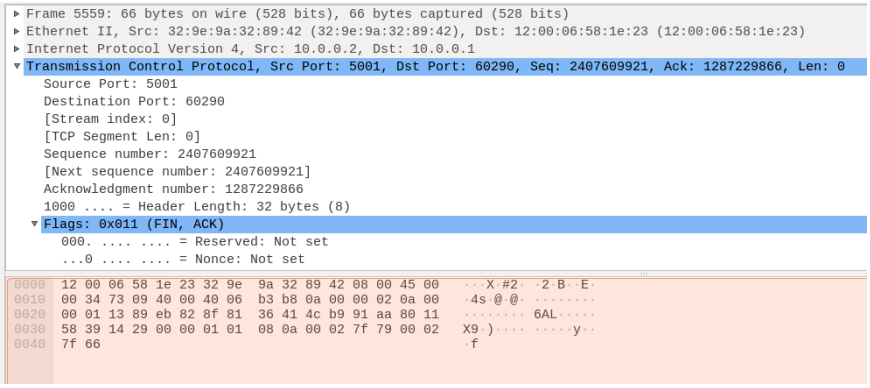
图中序号 5547，5559，5561 是挥手过程

```
▸ Frame 5547: 2746 bytes on wire (21968 bits), 2746 bytes captured (21968 bits)
▸ Ethernet II, Src: 12:00:06:58:1e:23 (12:00:06:58:1e:23), Dst: 32:9e:9a:32:89:42 (32:9e:9a:32:89:42)
▸ Internet Protocol Version 4, Src: 10.0.0.1, Dst: 10.0.0.2
▾ Transmission Control Protocol, Src Port: 60290, Dst Port: 5001, Seq: 1287227185, Ack: 2407609921, Len: 2680
    Source Port: 60290
    Destination Port: 5001
    [Stream index: 0]
    [TCP Segment Len: 2680]
    Sequence number: 1287227185
    [Next sequence number: 1287229866]
    Acknowledgment number: 2407609921
    1000 .... = Header Length: 32 bytes (8)
  ▾ Flags: 0x019 (FIN, PSH, ACK)
      000. .... .... = Reserved: Not set
      ...0 .... .... = Nonce: Not set
```

```
0000  32 9e 9a 32 89 42 12 00  06 58 1e 23 08 00 45 00   2··2·B·· ·X·#··E·
0010  0a ac de 3a 40 00 40 06  3e 0f 0a 00 00 01 0a 00   ···:@·@· >·······
0020  00 02 eb 82 13 89 4c b9  87 31 8f 81 36 41 80 19   ·····L· ·1··6A··
0030  00 3a 1e a1 00 00 01 01  08 0a 00 02 7f 66 00 02   ·:····· ·····f··
0040  7f 61 32 33 34 35 36 37  38 39 30 31 32 33 34 35    a234567 89012345
0050  36 37 38 39 30 31 32 33  34 35 36 37 38 39 30 31   67890123 45678901
0060  32 33 34 35 36 37 38 39  30 31 32 33 34 35 36 37   23456789 01234567
0070  38 39 30 31 32 33 34 35  36 37 38 39 30 31 32 33   89012345 67890123
0080  34 35 36 37 38 39 30 31  32 33 34 35 36 37 38 39   45678901 23456789
0090  30 31 32 33 34 35 36 37  38 39 30 31 32 33 34 35   01234567 89012345
00a0  36 37 38 39 30 31 32 33  34 35 36 37 38 39 30 31   67890123 45678901
00b0  32 33 34 35 36 37 38 39  30 31 32 33 34 35 36 37   23456789 01234567
00c0  38 39 30 31 32 33 34 35  36 37 38 39 30 31 32 33   89012345 67890123
00d0  34 35 36 37 38 39 30 31  32 33 34 35 36 37 38 39   45678901 23456789
00e0  30 31 32 33 34 35 36 37  38 39 30 31 32 33 34 35   01234567 89012345
00f0  36 37 38 39 30 31 32 33  34 35 36 37 38 39 30 31   67890123 45678901
0100  32 33 34 35 36 37 38 39  30 31 32 33 34 35 36 37   23456789 01234567
0110  38 39 30 31 32 33 34 35  36 37 38 39 30 31 32 33   89012345 67890123
0120  34 35 36 37 38 39 30 31  32 33 34 35 36 37 38 39   45678901 23456789
0130  30 31 32 33 34 35 36 37  38 39 30 31 32 33 34 35   01234567 89012345
```

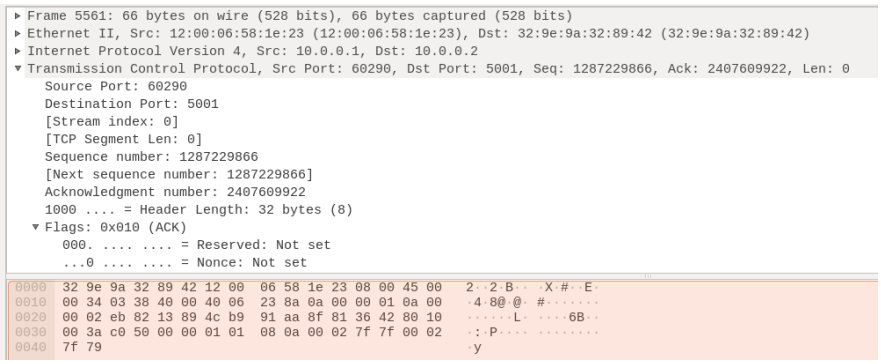
第一次挥手

源端口号：60290								目的端口号：5001
序号：1287227185								
确认号：2407609921								
首部长度	保留位用	URG	ACK	PSH	RST	SYN	FIN	



第二、三次挥手

源端口号：5001								目的端口号：60290
序号：2407609921								
确认号：1287229866								
首部长度	保留位用	URG	ACK	PSH	RST	SYN	FIN	



第四次挥手

源端口号：60290								目的端口号：5001
序号：1287229866								
确认号：2407609922								
首部长度	保留位用	URG	ACK	PSH	RST	SYN	FIN	

四、TCP 可靠传输

1.

197	1.092206	10.0.0.2	10.0.0.1	TCP	68 [TCP Dup ACK 196#1] 5001 → 60452 [ACK] Seq=3361608390 Ack=3731978847 Win=107520 Len=0 TSval=2744712 TSecr=2744707
198	1.103005	10.0.0.1	10.0.0.2	TCP	2964 60452 → 5001 [ACK] Seq=3731984639 Ack=3361608390 Win=29696 Len=2896 TSval=2744717 TSecr=2744712
199	1.103022	10.0.0.1	10.0.0.2	TCP	2964 60452 → 5001 [ACK] Seq=3731987535 Ack=3361608390 Win=29696 Len=2896 TSval=2744717 TSecr=2744712
200	1.104047	10.0.0.1	10.0.0.2	TCP	2964 60452 → 5001 [ACK] Seq=3731990431 Ack=3361608390 Win=29696 Len=2896 TSval=2744717 TSecr=2744712
201	1.104065	10.0.0.1	10.0.0.2	TCP	2964 60452 → 5001 [ACK] Seq=3731993327 Ack=3361608390 Win=29696 Len=2896 TSval=2744717 TSecr=2744712
202	1.105180	10.0.0.1	10.0.0.2	TCP	2964 60452 → 5001 [ACK] Seq=3731996223 Ack=3361608390 Win=29696 Len=2896 TSval=2744717 TSecr=2744712
203	1.108174	fe80::f4b9:40ff:fec…	ff02::fb	MDNS	329 Standard query response 0x0000 TXT, cache flush AAAA, cache flush fe80::f4b9:40ff:fec1:a18e PTR, cache flush cn-virtual…
204	1.113161	10.0.0.1	10.0.0.2	TCP	2964 [TCP Out-Of-Order] 60452 → 5001 [ACK] Seq=3731987535 Ack=3361608390 Win=29696 Len=2896 TSval=2744717 TSecr=2744712
205	1.121376	10.0.0.1	10.0.0.2	TCP	2964 [TCP Out-Of-Order] 60452 → 5001 [ACK] Seq=3731990431 Ack=3361608390 Win=29696 Len=2896 TSval=2744717 TSecr=2744712
206	1.128383	10.0.0.1	10.0.0.2	ICMP	100 Echo (ping) request id=0x1e5e, seq=12/3072, ttl=64 (no response found!)
207	1.132346	10.0.0.2	10.0.0.1	TCP	80 [TCP Window Update] 5001 → 60452 [ACK] Seq=3361608390 Ack=3731978847 Win=118784 Len=0 TSval=2744724 TSecr=2744707 SLE=3…
208	1.137886	10.0.0.1	10.0.0.2	TCP	2964 [TCP Retransmission] 60452 → 5001 [ACK] Seq=3731993327 Ack=3361608390 Win=29696 Len=2896 TSval=2744717 TSecr=2744712
209	1.143304	10.0.0.2	10.0.0.1	TCP	80 [TCP Dup ACK 196#2] 5001 → 60452 [ACK] Seq=3361608390 Ack=3731978847 Win=118784 Len=0 TSval=2744724 TSecr=2744707 SLE=3…
210	1.148341	10.0.0.2	10.0.0.1	TCP	80 [TCP Window Update] 5001 → 60452 [ACK] Seq=3361608390 Ack=3731978847 Win=124928 Len=0 TSval=2744728 TSecr=2744707 SLE=3…
211	1.153304	10.0.0.1	10.0.0.2	TCP	2964 [TCP Retransmission] 60452 → 5001 [ACK] Seq=3731996223 Ack=3361608390 Win=29696 Len=2896 TSval=2744717 TSecr=2744712
212	1.153344	10.0.0.1	10.0.0.2	TCP	1516 [TCP Fast Retransmission] 60452 → 5001 [ACK] Seq=3731978847 Ack=3361608390 Win=29696 Len=1448 TSval=2744730 TSecr=27447…
213	1.153345	10.0.0.1	10.0.0.2	TCP	1516 [TCP Retransmission] 60452 → 5001 [PSH, ACK] Seq=3731980295 Ack=3361608390 Win=29696 Len=1448 TSval=2744730 TSecr=27447…
214	1.154318	10.0.0.1	10.0.0.2	TCP	1516 [TCP Retransmission] 60452 → 5001 [ACK] Seq=3731981743 Ack=3361608390 Win=29696 Len=1448 TSval=2744730 TSecr=2744724
215	1.158410	10.0.0.2	10.0.0.1	TCP	80 [TCP Dup ACK 196#3] 5001 → 60452 [ACK] Seq=3361608390 Ack=3731978847 Win=124928 Len=0 TSval=2744728 TSecr=2744707 SLE=3…
216	1.168478	10.0.0.1	10.0.0.2	TCP	1516 [TCP Retransmission] 60452 → 5001 [ACK] Seq=3731983191 Ack=3361608390 Win=29696 Len=1448 TSval=2744734 TSecr=2744728

重传原因：数据包有丢失，且在超时间隔 RTO 内发送端接收到了 h2 发送的三次 TCP Dup ACK 196

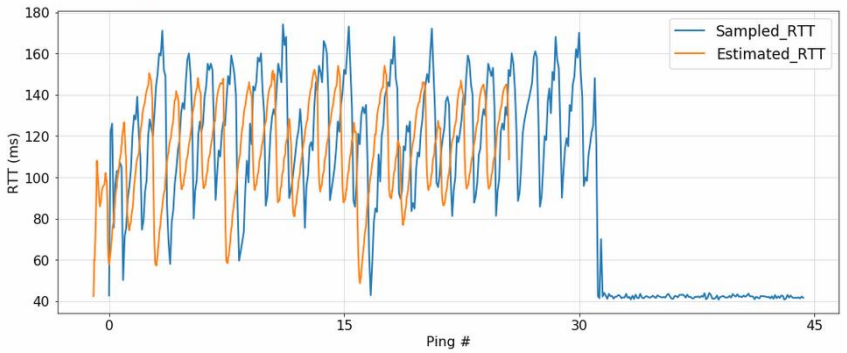
```
▸ Frame 212: 1516 bytes on wire (12128 bits), 1516 bytes captured (12128 bits)
▸ Linux cooked capture
▸ Internet Protocol Version 4, Src: 10.0.0.1, Dst: 10.0.0.2
▾ Transmission Control Protocol, Src Port: 60452, Dst Port: 5001, Seq: 3731978847, Ack: 3361608390, Len: 1448
    Source Port: 60452
    Destination Port: 5001
    [Stream index: 0]
    [TCP Segment Len: 1448]
    Sequence number: 3731978847
    [Next sequence number: 3731980295]
    Acknowledgment number: 3361608390
    1000 .... = Header Length: 32 bytes (8)
  ▸ Flags: 0x010 (ACK)
    Window size value: 58
    [Calculated window size: 29696]
    [Window size scaling factor: 512]
    Checksum: 0x19d1 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  ▸ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  ▾ [SEQ/ACK analysis]
      [iRTT: 0.032573000 seconds]
      [Bytes in flight: 20272]
      [Bytes sent since last PSH flag: 37648]
    ▾ [TCP Analysis Flags]
      ▸ [Expert Info (Note/Sequence): This frame is a (suspected) fast retransmission]
      ▸ [Expert Info (Note/Sequence): This frame is a (suspected) retransmission]
  ▸ [Timestamps]
    TCP payload (1448 bytes)
▸ Data (1448 bytes)
```

```
0000  00 03 00 01 00 06 8a 7d  38 b6 55 95 00 00 08 00   ·····} 8·U·····
0010  45 00 05 dc 02 29 40 00  40 06 1e f1 0a 00 00 01   E····)@· @·····
0020  0a 00 00 02 ec 24 13 89  de 71 7a 5f c8 5e 12 c6   ·····$·· ·qz_·^··
0030  80 10 00 3a 19 d1 00 00  01 01 08 0a 00 29 e1 9a   ···:···· ·····)··
0040  00 29 e1 94 36 37 38 39  30 31 32 33 34 35 36 37   ·)··6789 01234567
0050  38 39 30 31 32 33 34 35  36 37 38 39 30 31 32 33   89012345 67890123
0060  34 35 36 37 38 39 30 31  32 33 34 35 36 37 38 39   45678901 23456789
0070  30 31 32 33 34 35 36 37  38 39 30 31 32 33 34 35   01234567 89012345
0080  36 37 38 39 30 31 32 33  34 35 36 37 38 39 30 31   67890123 45678901
0090  32 33 34 35 36 37 38 39  30 31 32 33 34 35 36 37   23456789 01234567
00a0  38 39 30 31 32 33 34 35  36 37 38 39 30 31 32 33   89012345 67890123
```

数据包内容

2.

```
In [5]: from plot_ertt import plot_srtt_ertt
        plot_srtt_ertt('test_10')
```



橘色线为 ping 采样 RTT、蓝色线为基于 tcpprobe 的估计 RTT。从图中可以看出，估计的 RTT 变化没有采样得到的 RTT 变化剧烈，峰值低于采样而谷值高于采样。可能是因为采样得到的 RTT 具有一定的随机性，因此变化剧烈；而 tcpprobe 估计的 RTT 是对整个过程的估计，平滑性更好。

五、TCP 流量控制

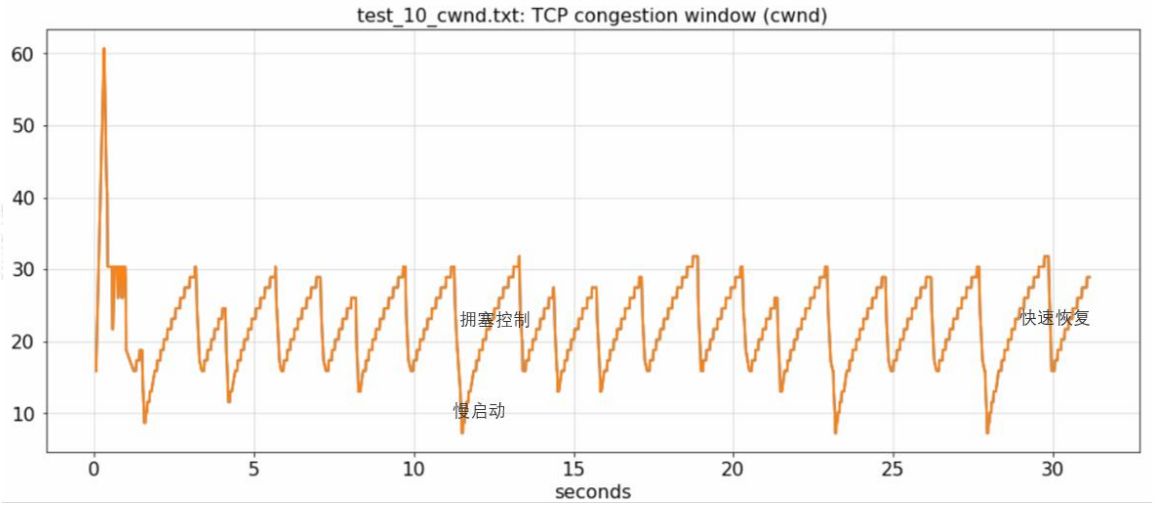
```
 85 0.418065    10.0.0.2    10.0.0.1    TCP     94 [TCP Window Update] 5001 → 33074 [ACK] Seq=2738894137 Ack=843222229 Win=159744
 87 0.434141    10.0.0.2    10.0.0.1    TCP     94 5001 → 33074 [ACK] Seq=2738894137 Ack=843223677 Win=162304 Len=0 TSval=8236239
 88 0.441125    10.0.0.2    10.0.0.1    TCP     94 5001 → 33074 [ACK] Seq=2738894137 Ack=843230917 Win=165376 Len=0 TSval=8236241
 90 0.450167    10.0.0.2    10.0.0.1    ICMP    98 Echo (ping) reply    id=0x30ad, seq=4/1024, ttl=64 (request in 62)
 91 0.450187    10.0.0.2    10.0.0.1    TCP     94 [TCP Window Update] 5001 → 33074 [ACK] Seq=2738894137 Ack=843230917 Win=168448
 93 0.459243    10.0.0.2    10.0.0.1    TCP     94 [TCP Window Update] 5001 → 33074 [ACK] Seq=2738894137 Ack=843230917 Win=171008
 95 0.466686    10.0.0.2    10.0.0.1    TCP     94 [TCP Window Update] 5001 → 33074 [ACK] Seq=2738894137 Ack=843230917 Win=174080
 97 0.474534    10.0.0.2    10.0.0.1    TCP     94 [TCP Window Update] 5001 → 33074 [ACK] Seq=2738894137 Ack=843230917 Win=177152
 99 0.482959    10.0.0.2    10.0.0.1    TCP     94 [TCP Window Update] 5001 → 33074 [ACK] Seq=2738894137 Ack=843230917 Win=179712
101 0.490547    10.0.0.2    10.0.0.1    TCP     94 [TCP Window Update] 5001 → 33074 [ACK] Seq=2738894137 Ack=843230917 Win=182784
103 0.498704    10.0.0.2    10.0.0.1    TCP     94 [TCP Window Update] 5001 → 33074 [ACK] Seq=2738894137 Ack=843230917 Win=185344
105 0.506375    10.0.0.2    10.0.0.1    TCP     94 [TCP Window Update] 5001 → 33074 [ACK] Seq=2738894137 Ack=843230917 Win=188416
108 0.515692    10.0.0.2    10.0.0.1    TCP     94 [TCP Window Update] 5001 → 33074 [ACK] Seq=2738894137 Ack=843230917 Win=194048
116 0.530736    10.0.0.2    10.0.0.1    TCP     94 [TCP Window Update] 5001 → 33074 [ACK] Seq=2738894137 Ack=843230917 Win=197120
117 0.540019    10.0.0.2    10.0.0.1    TCP     94 [TCP Window Update] 5001 → 33074 [ACK] Seq=2738894137 Ack=843230917 Win=200192
120 0.547378    10.0.0.2    10.0.0.1    TCP     94 [TCP Window Update] 5001 → 33074 [ACK] Seq=2738894137 Ack=843230917 Win=202752
122 0.555421    10.0.0.2    10.0.0.1    TCP     94 [TCP Window Update] 5001 → 33074 [ACK] Seq=2738894137 Ack=843230917 Win=205824
125 0.563008    10.0.0.2    10.0.0.1    TCP     94 [TCP Window Update] 5001 → 33074 [ACK] Seq=2738894137 Ack=843230917 Win=208896
127 0.571560    10.0.0.2    10.0.0.1    TCP     94 [TCP Window Update] 5001 → 33074 [ACK] Seq=2738894137 Ack=843230917 Win=211456
130 0.580811    10.0.0.2    10.0.0.1    TCP     94 [TCP Window Update] 5001 → 33074 [ACK] Seq=2738894137 Ack=843230917 Win=214528
132 0.587789    10.0.0.2    10.0.0.1    TCP     94 [TCP Window Update] 5001 → 33074 [ACK] Seq=2738894137 Ack=843230917 Win=217600
134 0.594918    10.0.0.2    10.0.0.1    TCP     94 [TCP Window Update] 5001 → 33074 [ACK] Seq=2738894137 Ack=843230917 Win=220160
136 0.603488    10.0.0.2    10.0.0.1    ICMP    98 Echo (ping) reply    id=0x30ad, seq=6/1536, ttl=64 (request in 107)
137 0.605069    10.0.0.2    10.0.0.1    TCP     94 [TCP Window Update] 5001 → 33074 [ACK] Seq=2738894137 Ack=843230917 Win=223232
139 0.611246    10.0.0.2    10.0.0.1    TCP     94 5001 → 33074 [ACK] Seq=2738894137 Ack=843248293 Win=226304 Len=0 TSval=8236284
142 0.620293    10.0.0.2    10.0.0.1    TCP     86 5001 → 33074 [ACK] Seq=2738894137 Ack=843259877 Win=228864 Len=0 TSval=8236286
144 0.629920    10.0.0.2    10.0.0.1    TCP     86 [TCP Window Update] 5001 → 33074 [ACK] Seq=2738894137 Ack=843259877 Win=231936
146 0.636145    10.0.0.2    10.0.0.1    TCP     94 [TCP Window Update] 5001 → 33074 [ACK] Seq=2738894137 Ack=843259877 Win=235008
148 0.644827    10.0.0.2    10.0.0.1    TCP     94 [TCP Window Update] 5001 → 33074 [ACK] Seq=2738894137 Ack=843259877 Win=237568
151 0.653521    10.0.0.2    10.0.0.1    TCP     94 [TCP Window Update] 5001 → 33074 [ACK] Seq=2738894137 Ack=843259877 Win=240640
154 0.660652    10.0.0.2    10.0.0.1    TCP     94 [TCP Window Update] 5001 → 33074 [ACK] Seq=2738894137 Ack=843259877 Win=243712
```

时间	0.450187	0.459243	0.466686	0.474534	0.482959	0.490547	0.498704	0.506375
接收窗口	168448	171008	174080	177152	179712	182784	185344	188416
时间	0.515692	0.530736	0.540019	0.547378	0.555421	0.563008	0.571560	0.580811
接收窗口	194048	197120	200192	202752	205824	208896	211456	214528
时间	0.587789	0.594918						
接收窗口	217600	220160						

接收窗口总体上是在上涨，但大小被 RcvBuffer 限制，因此在 cwnd 足够大的情况下，rwnd 大导致接收缓存区空余空间变小，会使得下一次的 rwnd 减小，而不能一直增大。

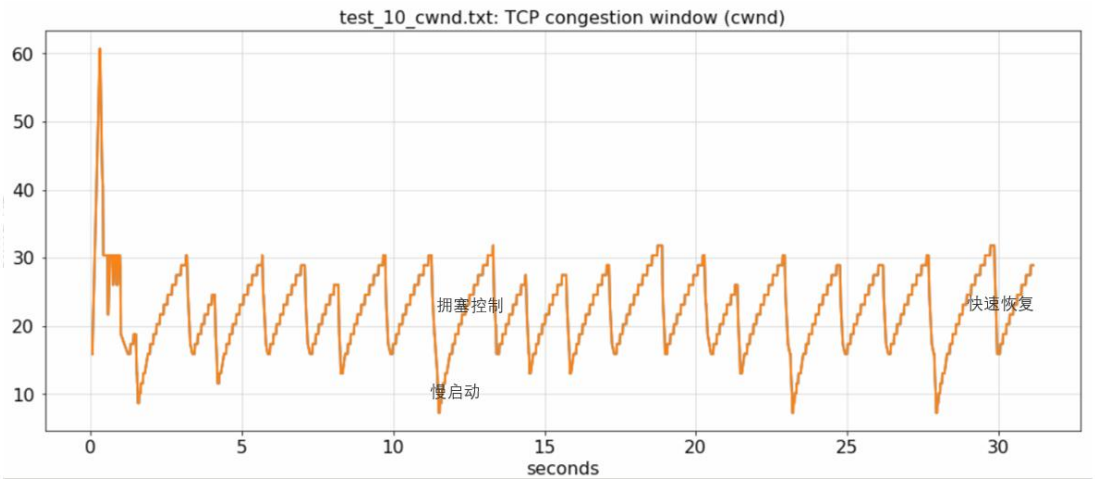
六、TCP 拥塞控制

1.

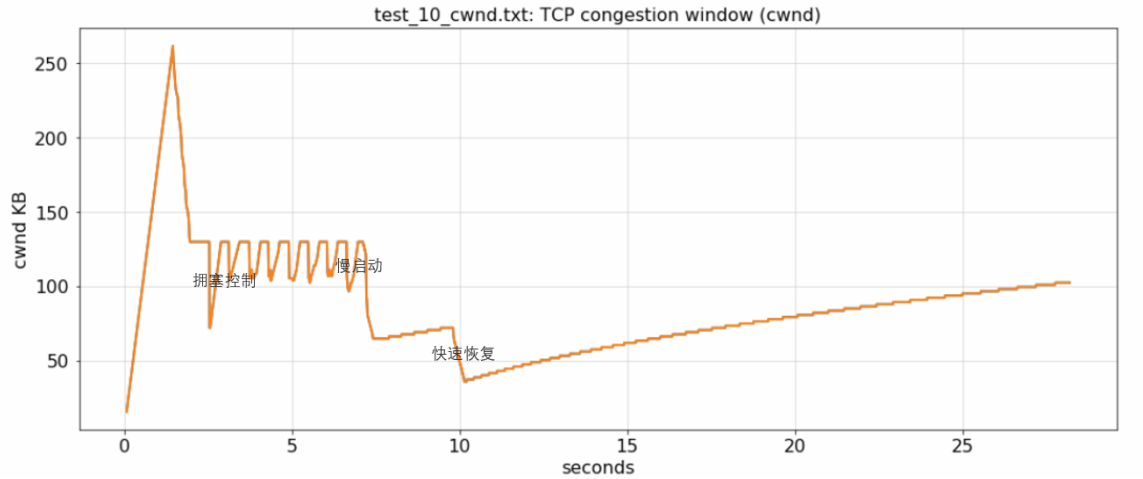


拥塞控制是线性增长段，前面一小段指数增长段是慢启动，cwnd 减半对应快速恢复

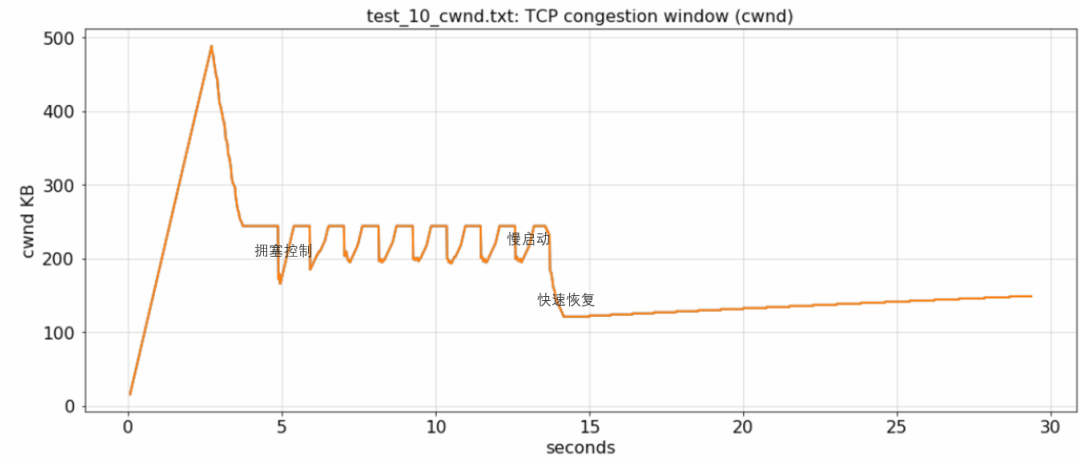
2.



最大队列长度=10



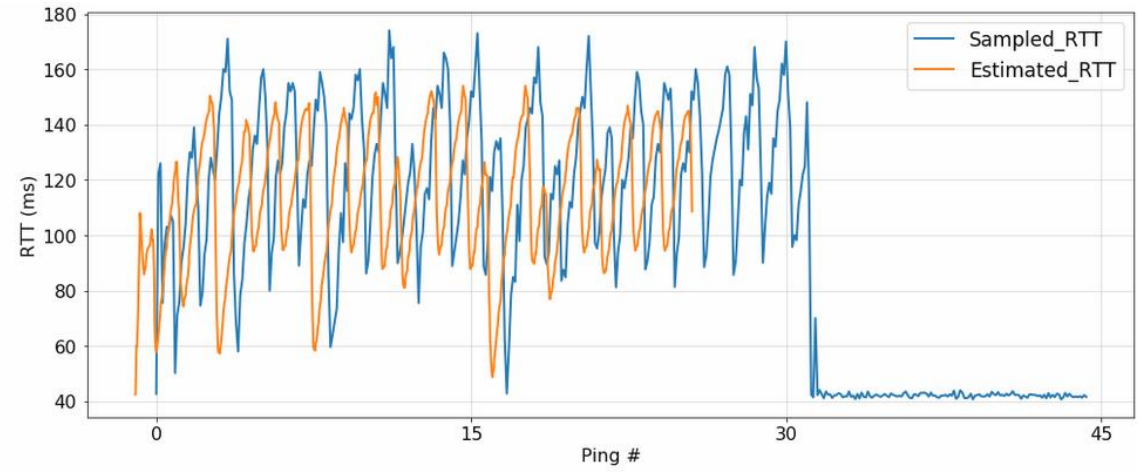
最大队列长度=50



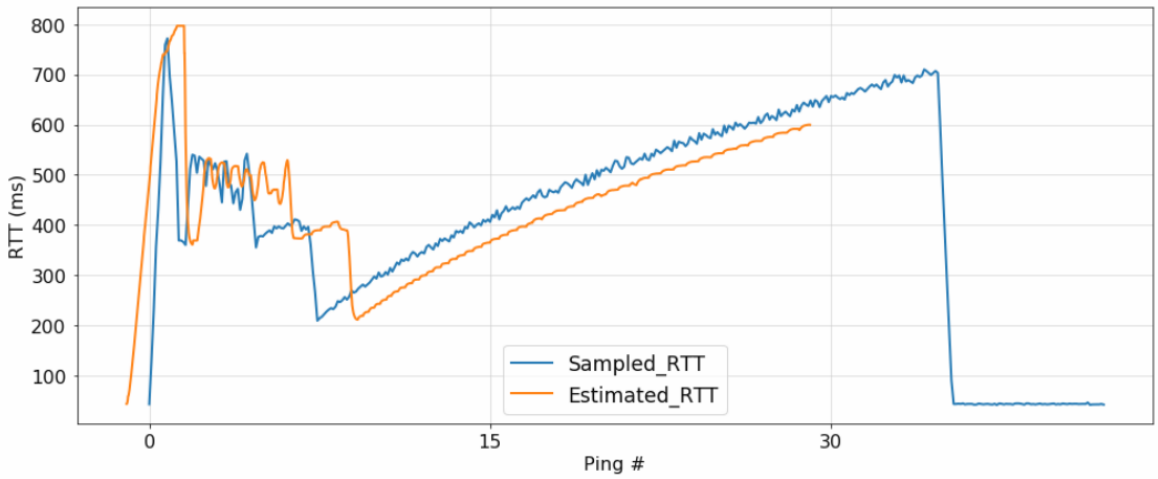
最大队列长度=100

三张图中随着路由器缓存大小的增加，cwnd 的均值变大，慢启动过程变长，整体上和均值的偏差变得不那么剧烈

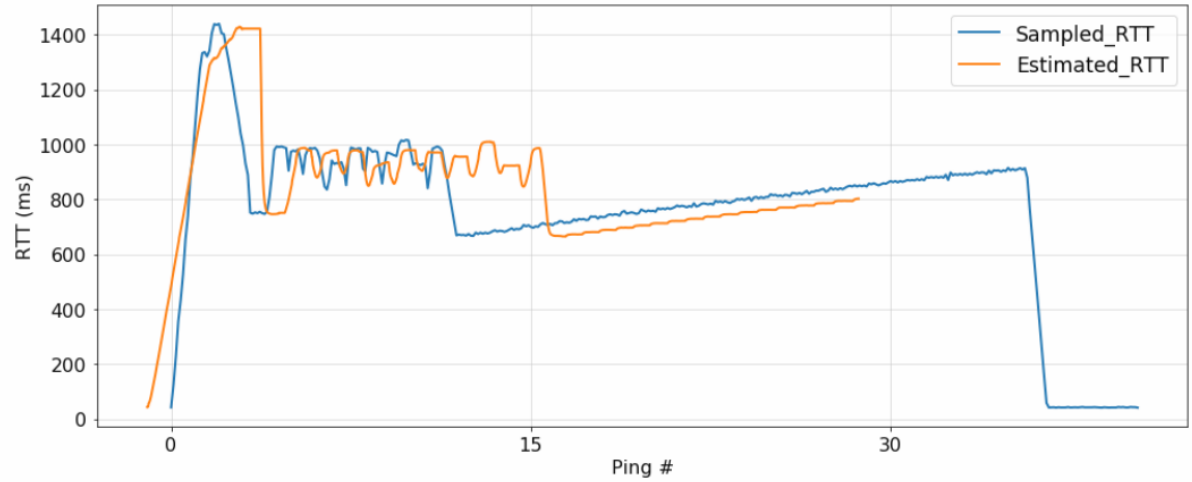
3.



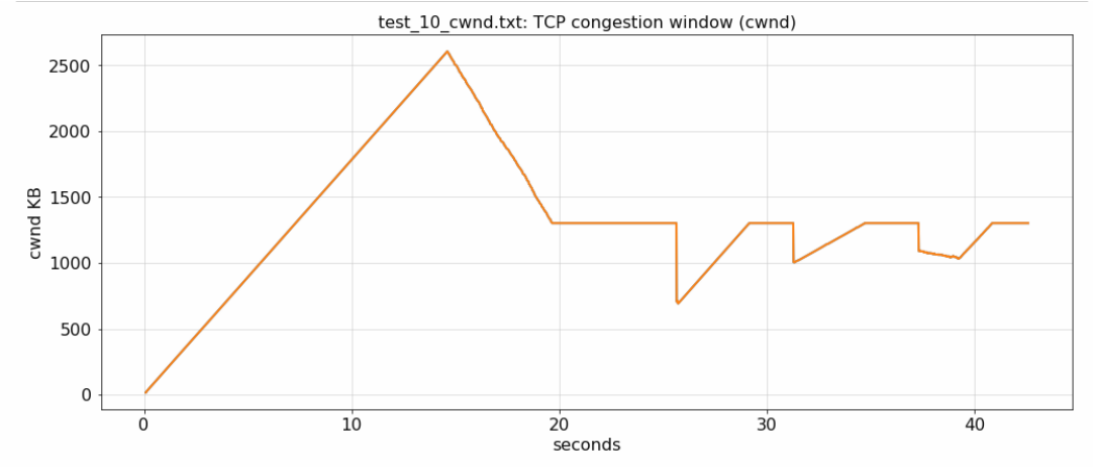
最大队列长度=10 的 RTT



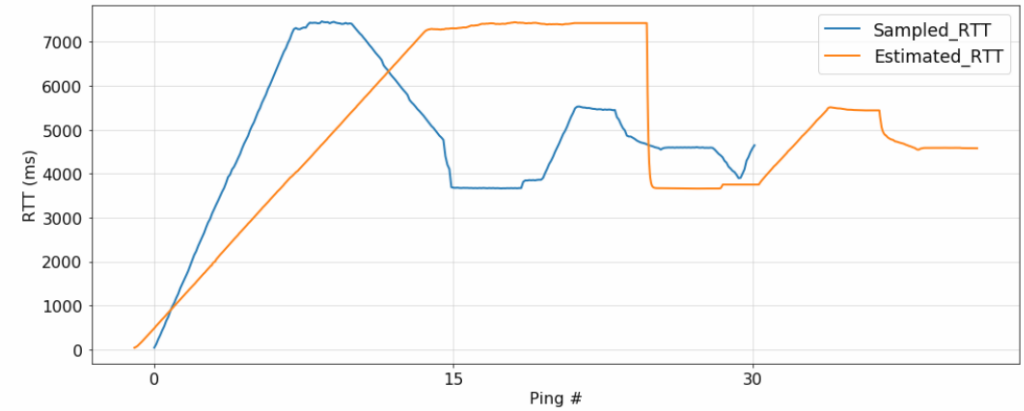
最大队列长度=50 的 RTT

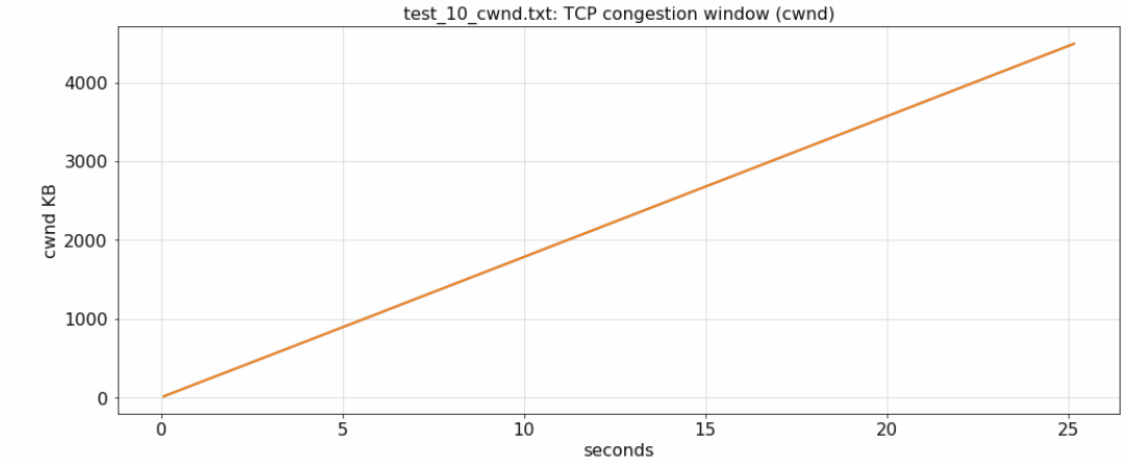


最大队列长度=100 的 RTT

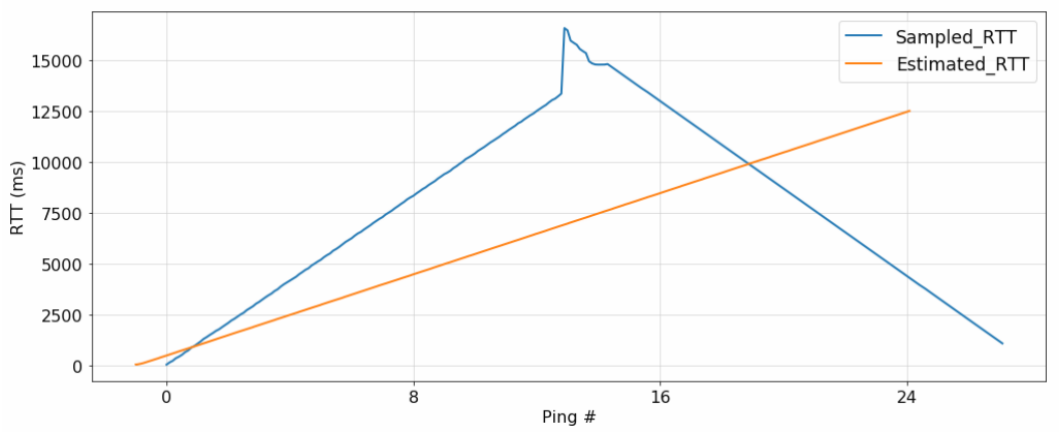


最大队列长度=500





最大队列长度=1000

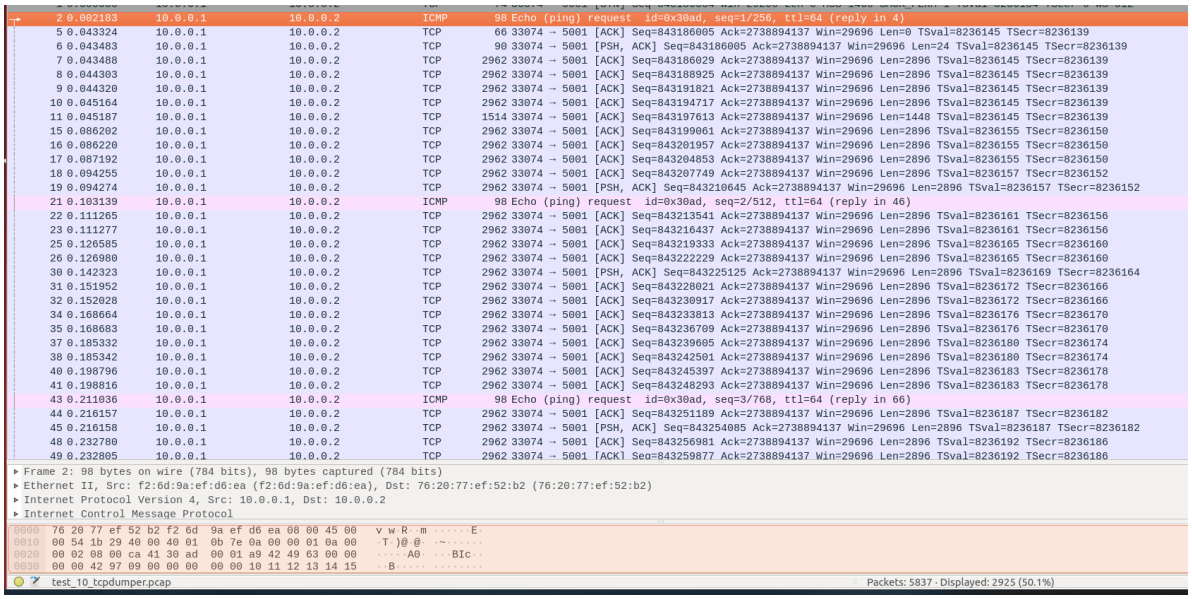


缓存大小为 10/50/100 和 500/1000 情况下 cwnd 显著增大，缓存大小为 100 时 cwnd 在 200KB 左右变化；缓存大小为 500 时 cwnd 在 1500KB 左右变化，缓存大小为 1000 时曲线一直在慢启动阶段，增长到了 4000KB。路由器缓冲足够大时，cwnd 可以保持一个相对大的值，使得很多数据包发出后在路由器缓存里很长时间里而不能很快传递给接收方，接收方发送 ACK 也要经过路由器缓存，其他条件不变，缓存越大数据包从路由器接收到路由器发出时间越长，RTT 也就越长。

七、思考题

（1） 本实验中，基于 Wireshark 和抓到的数据包，分析 h1 的接收窗口变化情况，请解释产生这样现象的原因。

答：



观察到 h1 的接收窗口大小一直是 29696,实际应用中家用电脑只和路由器连接,并不像路由器一样有多个连接,需要根据缓存空间剩余量决定接收窗口大小,因此 h1 的接收窗口大小使用定值即可满足需求。

（2） 本实验利用 Wireshark 分析抓包过程中，除了本实验重点分析的 TCP 协议数据包，还存在哪些其他类型数据包？通过进一步 Baidu 或查阅资料确定这些数据包对应于网络哪一层。

答：还存在 ICMP 数据包,根据网上资料它用于在 IP 主机、路由器之间传递控制消息,属于网络层协议。