Joel Sivanish

Professor Thurston

Lab 5

11/7/2025

1) TCP Ports: 43060, 5432 (P. #1). Port 5432 is the default port for PostgreSQL [1]

IP Address: 127.0.0.1 (P. #1), 127.0.0.5 (P. #13), 127.2.0.1 (P. #22), 127.8.0.1 (P. #35)

2) The Client: 127.0.0.1:43060 The Server: 127.0.0.1:5432. We know this because packet 1 is the beginning of the 3-way handshake with SYN (Client to Server), progressing to packet 2 for the SYN-ACK (Server to Client), and then packet 6 for the ACK (Client to Server).

3) All packets in this capture use the 127.0.0.0/8 loopback range, meaning the traffic never left the host system. This shows the capture was taken on the loopback interface rather than any physical Ethernet or Wi-Fi adapter. Loopback addresses (127.0.0.0–127.255.255.255) are managed by the local TCP/IP stack, allowing a device to send and receive its own packets internally without putting them on the network [2]. The large MSS value (~65,495 bytes) further supports that this was local communication.

4) The Client and Server MSS are both 65,495 bytes as shown in the SYN and SYN-ACK of the 3-Way handshake. Client MSS: 65,495 (P. #1), Server MSS: 65,495 (P. #2). The MSS defines the largest TCP payload (excluding headers) that a device is willing to receive. It's determined by the MTU minus the TCP and IP header sizes [3]. Because this traffic stayed on the loopback interface, there were no hardware limits like Ethernet's 1500-byte MTU, allowing a much higher MSS than the usual 1460 bytes [3].

5) No. With MSS = 65,495 bytes, there are no TCP segments whose payload (tcp.len) exceeds MSS. In Wireshark, the display filter tcp.len > 65495 returns 0 packets [4].

6) Retransmissions occur when a sender does not receive an acknowledgment (ACK) for a segment before the retransmission timeout (RTO) expires. TCP starts this timer as soon as a segment is sent, and if the ACK doesn't arrive in time, it resends the segment [5]. Common causes include packet loss, congestion, incorrect flow control settings, duplex mismatches, or network hardware issues [5].

7) Wireshark marks retransmissions at packets #23, #24, #25, #33, and #36, all resending the same sequence numbers and payloads as earlier data. The most likely cause is artificial network delay or packet loss simulated with the Linux tc netem utility. tc (traffic control) allows users to emulate packet loss, delay, duplication, or corruption to test network behavior [6].

8) Wireshark flags these as retransmissions: #3, #4, #5, #23, #24, #25, #32, #33 (and #36 as spurious). Total TCP frames = 39.

Overall TCP frames (excluding spurious): $8/39 = \mathbf{20.5}\%$

Overall TCP frames (including spurious): $9/39 = \mathbf{23.1}\%$

These rates are far above common guidance (internet-facing traffic typically shouldn't exceed ~2% retransmissions). Elevated rates can impact service responsiveness and user experience [7].

9) Yes. The Linux tc (traffic control) utility can deliberately introduce packet delay, loss, duplication, or corruption to test how TCP reacts. The retransmissions in this capture were most likely triggered by tc netem rules that emulate packet loss or delay on the loopback interface [6]. A command like this could have been used to create the same behavior seen in the pcap: sudo tc qdisc add dev lo root netem loss 10% delay 100ms 20ms

10) The display filter tcp.analysis.retransmission shows only the retransmitted TCP segments [4]. If you also want to include fast retransmissions, you can use: tcp.analysis.retransmission || tcp.analysis.fast_retransmission.

11) Yes. The retransmissions in this capture clearly slow down the data flow. After each retransmission event (P. #23–25, #32–33, and #36), there's a noticeable pause in traffic before the connection resumes. These gaps line up with the sender waiting for its retransmission timeout (RTO) to expire before trying again.

12) Yes. Following the TCP stream from the client to the server shows data that begins with the ASCII header %PDF-1.5, which identifies a PDF document. This signature appears near the start of the first data segment (P. #8) and continues through the large data payloads.

13) The data can be reconstructed in Wireshark by right-clicking a packet in the TCP stream and selecting "Follow → TCP Stream." This combines all TCP payloads from the client to the server into one continuous file using Wireshark's built-in TCP reassembly feature, which pieces together data spread across multiple packets [8]. I exported the stream as raw data and manually changed the file extension to ".pdf." After renaming it, the file opened successfully and contained the instructions for Lab1 confirming that the transmitted data was a PDF document.

14) No. There are no DNS packets in this capture. All traffic occurs between 127.x.x.x loopback addresses, meaning the communication stayed entirely on the local machine. Because no domain names needed to be resolved, there was no need for a DNS lookup.

Work Cited

[1] Station X Networks. “Common Ports Cheat Sheet.” Station X – Cyber Security Training and Resources, 2024. https://www.stationx.net/common-ports-cheat-sheet/

[2] GeeksforGeeks. “What Is a Loopback Address?” Computer Networks, 23 July 2025. https://www.geeksforgeeks.org/computer-networks/what-is-a-loopback-address/

[3] Cloudflare. “What Is Maximum Segment Size (MSS)?” Cloudflare Learning Center, 2024. https://www.cloudflare.com/learning/network-layer/what-is-mss/

[4] Wireshark Foundation. “TCP – Wireshark Display Filter Reference.” Wireshark Docs, 2025. https://www.wireshark.org/docs/dfref/t/tcp.html

[5] ExtraHop Networks. “Understanding TCP Retransmission Causes and Solutions.” ExtraHop Blog, 2024. https://www.extrahop.com/blog/understanding-tcp-retransmission-causes-and-solutions-extrahop/

[6] Vouzis, Panagiotis. “How to Use the Linux Traffic Control.” NetBeez Blog, 29 November 2017. https://netbeez.net/blog/how-to-use-the-linux-traffic-control/

[7] Wallerstorfer, Dirk. “Detecting Network Errors and Their Impact on Services.” Dynatrace Blog, 14 Dec 2022 (updated 22 Apr 2024). https://www.dynatrace.com/news/blog/detecting-network-errors-impact-on-services/

[8] Wireshark Foundation. “Packet Reassembly.” Wireshark User’s Guide, Section 7.8, 2024. https://www.wireshark.org/docs/wsug_html_chunked/ChAdvReassemblySection.html