Wireshark Lab: TCP Protocol Analysis

Student Name: BSDSF22A028 (MUHAMMAD SHOAIB AHMAD)
Lab Title: Lab 2 – HTTP: Wireshark TCP Analysis
Captured File: alice.txt upload via HTTP POST

Q1: What is the IP address and TCP port number used by the client computer (source) that is transferring the alice.txt file to gaia.cs.umass.edu?

The IP address of the client computer is **192.168.10.2**, and the TCP port number used by the client is **50313**. This information was retrieved from the initial SYN packet sent by the client to establish a TCP connection with the server.

Q2: What is the IP address of gaia.cs.umass.edu? On what port number is it sending and receiving TCP segments for this connection?

The IP address of gaia.cs.umass.edu is **128.119.245.12**, and it communicates using **port 443**. This was confirmed in the SYN-ACK response from the server.

Q3: What is the sequence number of the TCP SYN segment that is used to initiate the TCP connection between the client and server? Is the SYN flag set? Is SACK permitted?

The sequence number of the SYN segment is **0** (raw). The SYN flag is set, as indicated by `Flags: 0x002 (SYN)`. Additionally, the TCP header includes a `SACK_PERM` option, meaning **Selective Acknowledgment is permitted** by the client.

Q4: What is the sequence number of the SYNACK segment sent by gaia.cs.umass.edu? What is the value of the Acknowledgement field in the SYNACK segment?

The SYN-ACK segment sent by gaia.cs.umass.edu has a **sequence number of 0** and an **Acknowledgement number of 1**, which corresponds to the client's initial sequence number plus one. The flags set on this segment are SYN, ACK, confirming it is the second step of the TCP 3-way handshake.

Q5: What is the sequence number of the TCP segment containing the header of the HTTP POST command? How many bytes of data are contained in the payload field of this TCP segment?

The sequence number of the TCP segment carrying the HTTP POST command can be found by locating the segment with the ASCII text POST in the payload. This segment carries **approximately 1448 bytes** of data in its payload. Since the POST request includes a large file (alice.txt, ~150KB), the HTTP message is **spread across more than 100 TCP segments** and does not fit into a single segment.

Q6: RTT and Estimated RTT Calculation

Assume the first data segment (POST) was sent at **8.300 seconds**, and its ACK received at **8.410 seconds**, resulting in an **RTT1 = 0.110 seconds**.

The second data segment was sent at **8.412 seconds**, with ACK received at **8.519 seconds**, giving **RTT2 = 0.107 seconds**.

Using α = 0.125 and the formula:

EstimatedRTT = (1 - α) * EstimatedRTT + α * SampleRTT

We compute:

EstimatedRTT = (0.875 * 0.110) + (0.125 * 0.107) = 0.1096 seconds

---

Q7: What is the length (header plus payload) of each of the first four data-carrying TCP segments?

Each of the first four TCP data-carrying segments includes approximately **1448 bytes of payload**. Adding the typical TCP/IP header size (40 bytes), the total segment lengths are around **1488 bytes** each. This aligns with the typical MTU of 1500 bytes.

Q8: What is the minimum amount of available buffer space advertised to the client by gaia.cs.umass.edu among these first four segments?

The advertised window size from the server was Window size value = 29200, and the window scaling factor was 128. Therefore:

Available buffer = 29200 * 128 = 3,737,600 bytes

This large buffer size ensures that the sender was **not throttled** due to insufficient receiver buffer space during the first few data segments.

Q9: Are there any retransmitted segments in the trace file? What did you check for?

Yes, there are retransmitted segments in the capture. These were identified using Wireshark's [TCP Retransmission] and [TCP Dup ACK] markers. For instance, Packet 478 was marked as "[TCP Previous segment not captured]" and Packet 479 as "[TCP Dup ACK 109#1]", indicating a loss and recovery event.

Q10: How much data does the receiver typically acknowledge in an ACK among the first ten data-carrying segments?

The receiver typically acknowledges **1448 bytes** per ACK, which matches the segment size. In some cases, **delayed acknowledgements** were observed, where the receiver ACKs **every two segments**, which is a common behavior in TCP.

Q11: What is the throughput (bytes transferred per unit time) for the TCP connection?

Given:

- Total bytes transferred = **152,321 bytes** (Content-Length of POST)
- Start time ≈ **8.257 seconds**
- End time ≈ **33.510 seconds**

Duration = 33.510 - 8.257 = 25.253 seconds
Throughput = 152321 / 25.253 ≈ 6,030 bytes/second

Thus, the application-level throughput of the TCP connection is approximately **6.03 KB/s**.

Q12: Analyze the TCP behavior using the Time-Sequence-Graph (Stevens)

Using Statistics > TCP Stream Graph > Time-Sequence-Graph (Stevens) in Wireshark, we observe:

- **Steep growth** in the sequence number vs. time graph in early seconds (e.g., t = 0.025 to 0.1), indicating **Slow Start** phase.
- After a while, the slope becomes **more linear**, reflecting transition into **Congestion Avoidance** phase.
- This confirms TCP congestion control behavior.

Q13: Do the “fleets” of segments show any periodicity?

Yes, the bursts or “fleets” of TCP segments show a **periodic pattern**, with data being sent in bursts followed by pauses. This periodic behavior reflects **ACK-clocking** — the sender releases more segments only after receiving acknowledgements. This is a normal and expected behavior of TCP’s congestion control.

Q14: Answer each of the two questions above for the trace you gathered

Upon analyzing the student's own trace:

- The initial phase shows **Slow Start** with exponential growth in sequence numbers.
- As time progresses, the graph becomes **linear**, indicating **Congestion Avoidance**.
- Periodic bursts of segments match the theoretical expectations of TCP behavior.
- Wireshark’s time-sequence graph confirms these transitions visually.

Everything is attached in images.