KOÇ UNIVERSITY
COMP 416 - Computer Networks, Fall 2020

Project 2

Berkay Barlas, 0054512

December 9, 2020

Contents

1 Part 1 - SSL Implementation and Experiments

The maximum possible port number is 65535 (max 16-bit unsigned integer) therefore choosing $54512 + 05 = 54517$ as port number will not be a problem.

1.1 Questions

1. Locate the SSL Server IP address and port number, client IP address and port number through which these agents are communicating by using Wireshark.

   In this I used 4444 port as default for TCP connections and the port used for SSL connections is 54517. Both client and server IP addresses are 127.0.0.1 because it's the dedicated localhost IP.

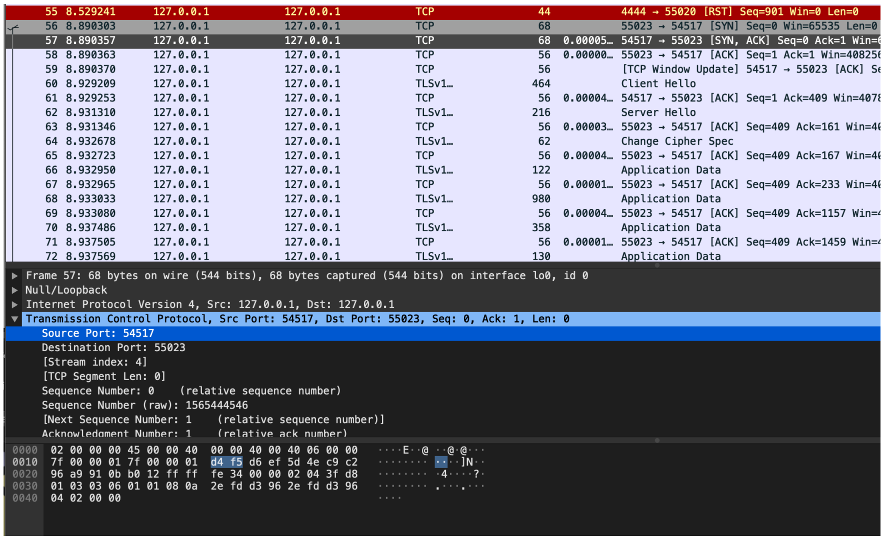

Figure 1: Wireshark in Loopback mode. client port is 55023

2. Locate the data containing TCP segments. What is written in the data field? Compare it with the data you exchanged between the client and server. Why do you think is this the case?

   In the data field I can clearly see the messages between client and server in TCP segments.

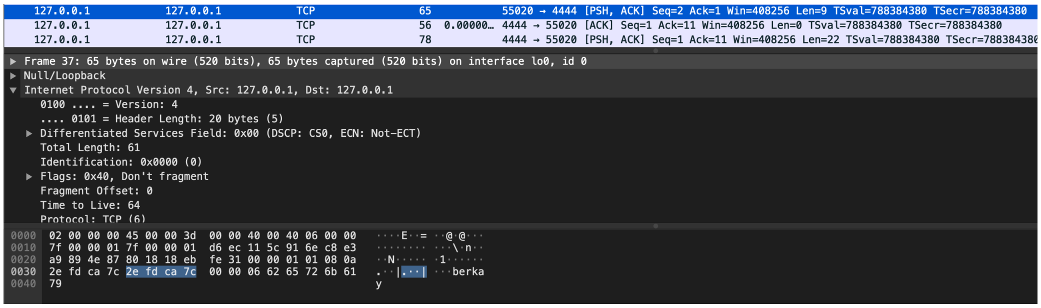

Figure 2: Wireshark Data Field for Username Message

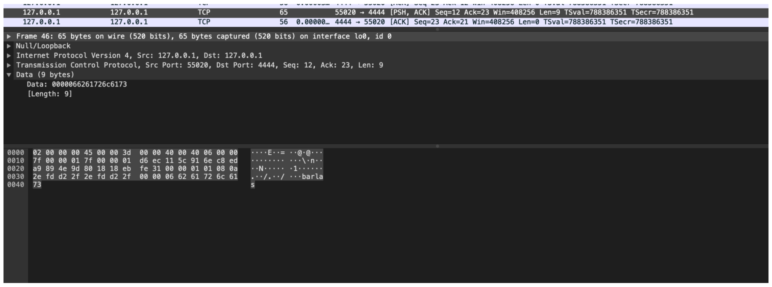

Figure 3: Wireshark Data Field for Password Message

3. How many TCP segments are transmitted in total while your KUSIS username + KUSIS ID is exchanged one by one with non-persistent connections?

   If investigate from the Wireshark filtering with ssl port I used, 54517, we can see that 285 TCP segments are transmitted during SSL connection for ACK messages.

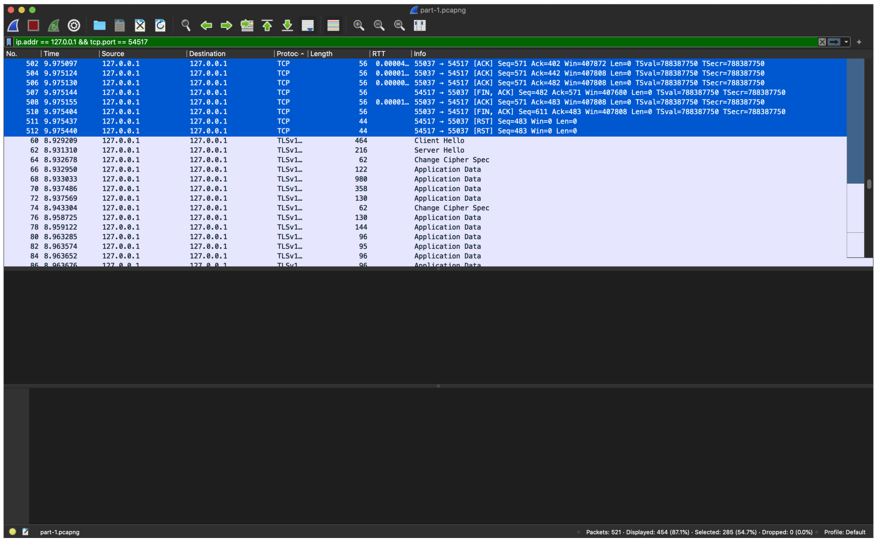

Figure 4: Wireshark Number of TCP segments During SSL connection

4. What difference did you see in the payload of SSL and TCP? Can you locate the login name and password entered by the user? Can you locate the email information?

   I can't locate email information because it's not used in this communication. The payloads in TCP are visible therefore I can locate my kusis information however SSL payload is encrypted.

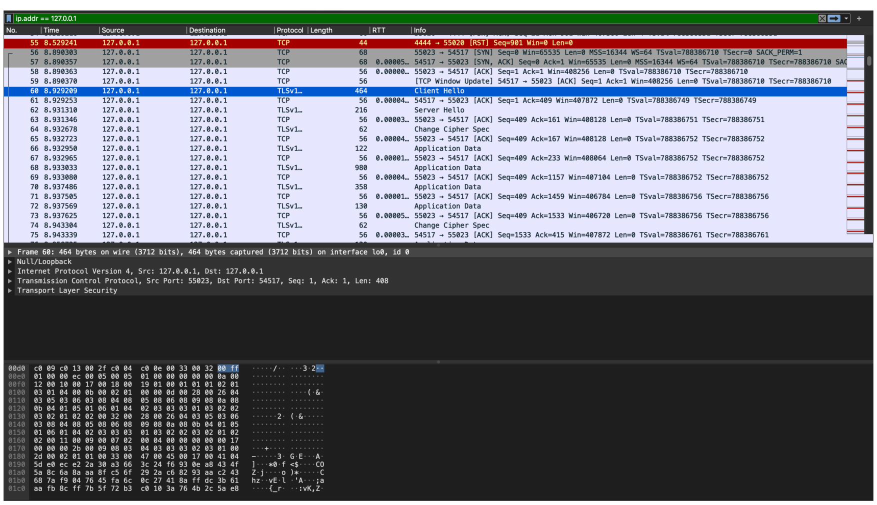

Figure 5: Wireshark SSL Payload

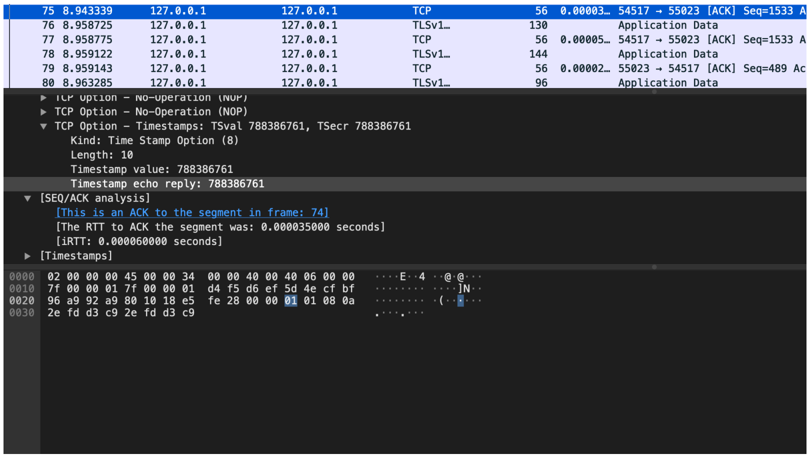

Figure 6: Wireshark SSL Payload

2 Part 2 - TCP Experiments

2.1 Questions

5. What is the significance of the various IP addresses shown in the Flow Graph?

   There are multiple IP addresses in the flow that shows the source and the destination of packets. This IP addresses identifies the different source and network connections.

6. What are the sequence numbers (which appear in the Wireshark program) of the segments used for the 3-way handshake protocol that initiates the first TCP connection? Sequence number for first ACK from the client and SYN/ACK from the server in 3-way handshake protocol are 0. The

   The raws values: 302580735,1535182375, 302580736

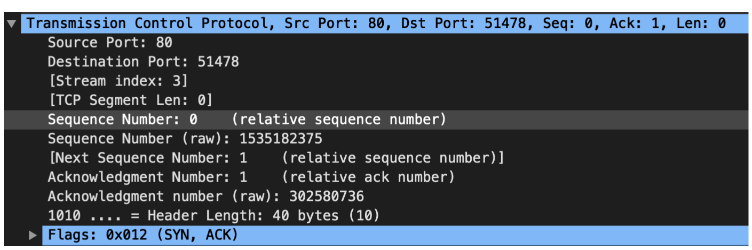

Figure 7: SYNACK segment

7. What is the value of the Acknowledgement field in the SYNACK segment? How did gaia.cs.umass.edu determine that value? What is it in the segment that identifies the segment as a SYNACK segment?

   As in the previous figure, the value of the ack field is 1. This acknowledgement value is equals to the sequence number of the arriving package incremented by one. The SYN and ACK nonzero flags in the segment identifies as SYNACK segment.

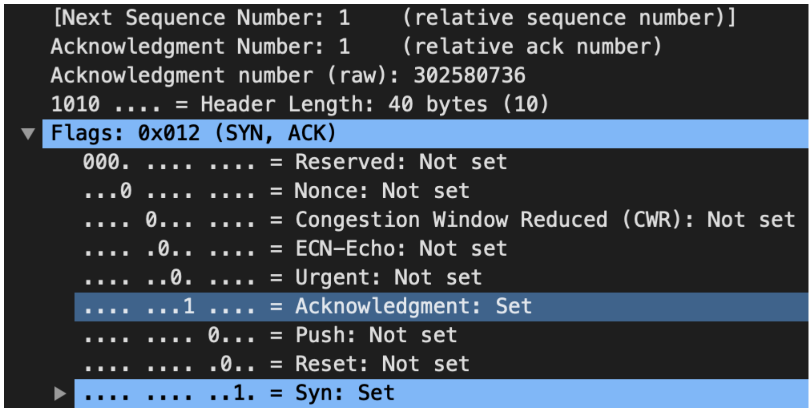

Figure 8: SYNACK segment flags

8. What is the sequence number of the SYNACK segment sent by gaia.cs.umass.edu to the client computer in reply to the SYN?

   The relative sequence number of the SYNACK is 0. Raw sequence number is 1535182375. Also

9. What is the sequence number of the TCP segment containing the HTTP POST command?

   The (relative) sequence number of the TCP segment is 1 and the raw sequence is 3694263361

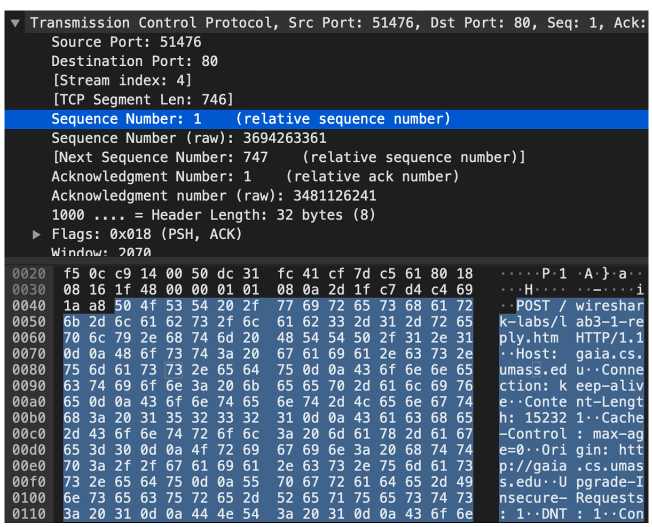

Figure 9: TCP segment containing the HTTP POST

10. Consider the TCP segments containing the HTTP POST as the last segment in the TCP connection. What are the sequence numbers of the first six segments in the TCP connection? At what time was each segment sent? When was the ACK for each segment received? Given the difference between when each TCP segment was sent, and when its acknowledgement was received, what is the RTT value for each of the six segments? What is the EstimatedRTT value (see Section 3.5.3 in the textbook) after the receipt of each ACK Assume that the value of the EstimatedRTT is equal to the measured RTT for the first segment, and then is computed using the EstimatedRTT equation (Section 3.5.3 in the textbook) for all subsequent segments.

    EstimatedRTT values are calculated in a similar way to moving average procedure.

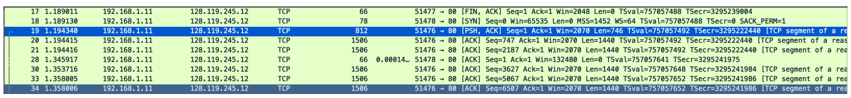

Figure 10: First six packages

3 Part 3 - UDP Experiments

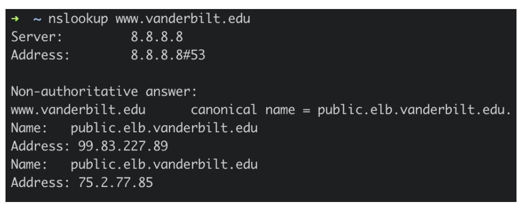

Figure 11: NS lookup command for www.vanderbilt.edu

3.1 Questions

11. What display filter did you apply in order to see appropriate packets?

    To see the packet for nslookup for www.vanderbilt.edu, dns.qry.name == ”www.vanderbilt.edu” filter can be used.

Figure 12: Wireshark filter for www.vanderbilt.edu

12. Which application layer and transport layer protocol do nslookup work on? What is the reason that transport layer protocol is chosen?

    As shown in figure from Wireshark application, nslookup works on DNS Application layer and UDP Transport layer protocols. UDP is used because it’s faster than TCP. TCP has 3-way handshake and congestion control methods that makes it slower compared to connectionless UDP protocol.

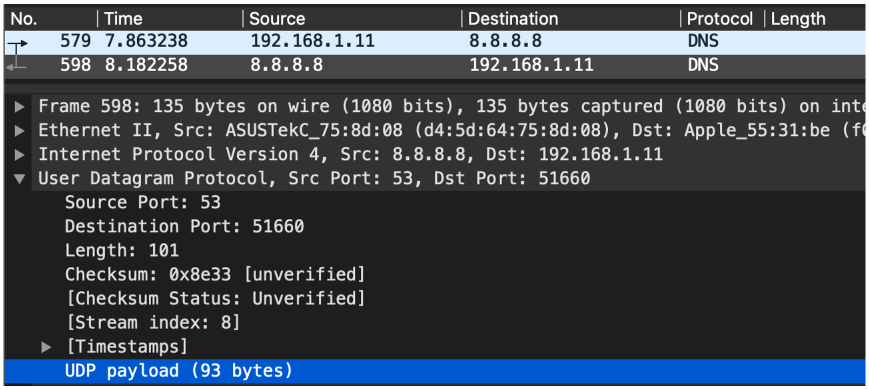

Figure 13: nslookup protocols

13. Can you derive the local DNS server you connected work in iterative or recursive manner? If you can or cannot please provide a detailed explanation. Please also briefly explain the advantages and disadvantages of iterative and recursive approach over each other.

    Yes, we can determine that using Wireshark. The flag in Domain Name System states that the local DNS that is for "www.vanderbilt.edu" works in recursive manner as shown in the figure.

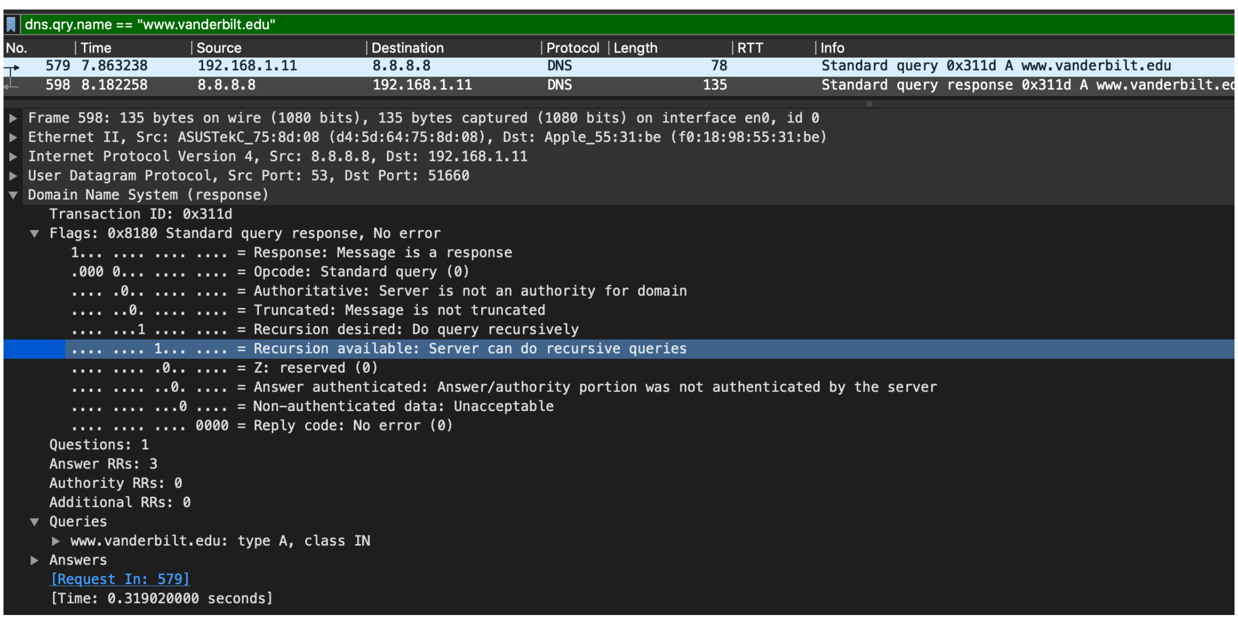

Figure 14: DNS Flags

14. What are the header lengths of application layer protocol and transport layer protocol

that nslookup works on?

The application layer, DNS, has 12 byte length header which includes transaction ID (2 bytes), flags (2 bytes), number of queries (questions, 2 bytes), number of answers (2 bytes), number of authoritative records (2 bytes), number of additional records in packet (bytes) fields.

The transport layer, UDP, has 8 byte length header which includes source(2 bytes) and destination(2 bytes) ports, length(2 bytes) and checksum(2 bytes) fields.

15. How many checksums does an UDP segment have in the checksum field? Why?

    UDP itself has one 2 byte checksum for checking payload. There is also another 2 byte checksum for header for checking the received datagram.

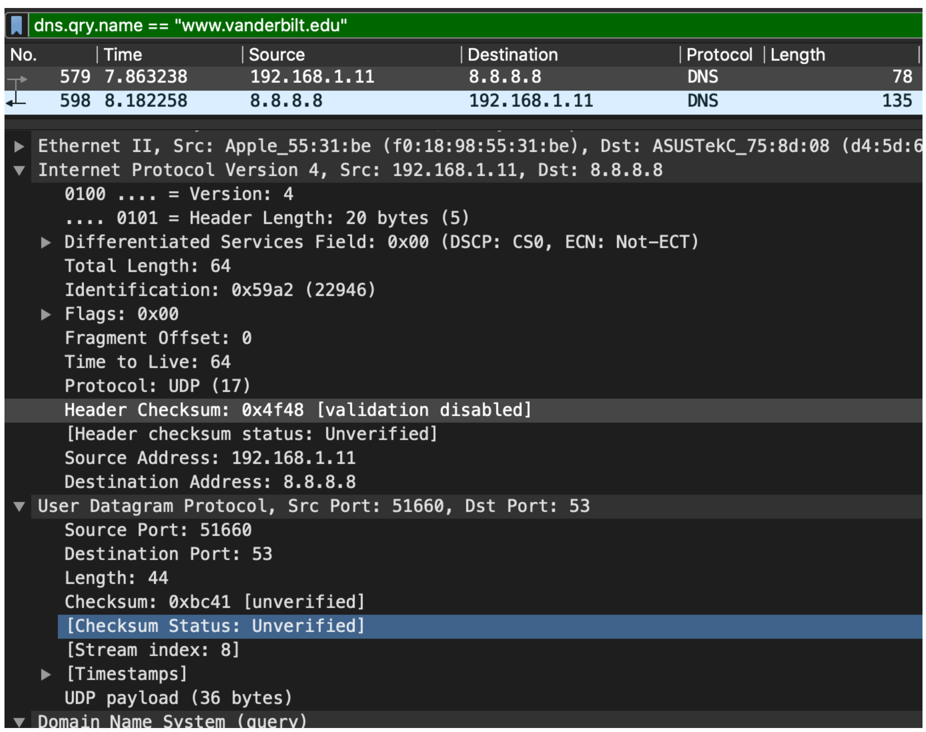

Figure 15: UDP DNS segment fields

4 Part 4 - Stop-and-Wait ARQ Protocol

4.1 Part 4.b - Analyzing the Implementation

16. How many retransmissions have occurred from the sender to the receiver? Explain.

    According to Wireshark, there were 50 packages send from sender(port 8000) to receiver(port 8001), 32 packages(ACKs) from receiver to sender. This means there is at least 50-32=18 re-transmissions occurred during this process.

17. How long did it take to transfer all the packets? Does this result make sense? Why or why not?

    Reducing the timeout in sender side reducing the transfer time significantly, however, it might cause unnecessary re-transmission.

    With 500ms sender side timeout, the time for total transmission took roughly 24s. This doesn't make sense and it is indicates the packet loss. Because 18 re-transsmission means it should roughly take 9s assuming that the local transmission time for successful packages are in milliseconds. The results from Wireshark indicates that there were around 48 timeouts in sender side. These losses not visible through Wireshark due to underlying implementation.

ip.dst == 192.168.1.11

No.	Time	Source	Destination	Protocol	Length	RTT	Info
25	0.800584	192.168.1.11	192.168.1.11	UDP	187		8001 → 8000 Len=155
26	0.803383	192.168.1.11	192.168.1.11	UDP	182		8000 → 8001 Len=150

Figure 16: Stop-and-Wait ARQ Protocol Analysis start

112	[illegible]	192.168.1.11	192.168.1.11	UDP	182	8000 → 8001 Len=150
113	25.010894	192.168.1.11	192.168.1.11	UDP	185	8001 → 8000 Len=153
114	25.011461	192.168.1.11	192.168.1.11	UDP	182	8000 → 8001 Len=150

Figure 17: Stop-and-Wait ARQ Protocol Analysis end

```java
public Packet[] receiveWithARQ() {

    List<Packet> packets = new ArrayList<>();
    int lastSequence = -1;
    while(true) {
        Packet p = receivePacket(15000);
        // Stop receiving either on timeout or when we received the last
            message packet.
        if(p.timedOut) break;

        sendAck(p.sequenceNumber % 2, p.lastPacket);

        if(lastSequence == p.sequenceNumber) continue;
        lastSequence = p.sequenceNumber;
        packets.add(p);

        if(p.lastPacket) break;

    }
    Packet[] pks = new Packet[packets.size()];
    for(int i = 0; i < pks.length; i++) pks[i] = packets.get(i);
    return pks;
}
```

```
public void sendWithARQ(Packet[] packets) {
    int currentPacket = 0;
    while (currentPacket < packets.length) {
        sendMsgPacket(currentPacket % 2, currentPacket == packets.length-1,
            packets[currentPacket]);
        Packet p = receivePacket(500);
        if(!p.timedOut) {
            currentPacket++;
        }
    }
}
```