Justin Chen, Suyash Amatya

September 9, 2025

CSCI 4220 - Network Programming

Lab 1 Submission

1. What was the port number on the client side?

- 54801

2. What was the port number on the server side?

- 9877

3. Briefly explain how these port numbers were decided for the UDP protocol?

- The sevrer port number is static and was hardcoded as 9877 in the unp.h file.
- The client port is dynamic and temporary. When the client is ran, a random unused port number is assigned by the OS.

4. How large is the UDP header?

- The UDP header is 8 bytes.
    - 2 bytes for source port
    - 2 bytes for destination port
    - 2 bytes for length
    - 2 bytes for checksum

5. How large is the application data? (Answer this for just one of your packets)

- For the packet "hello", the length is 6 bytes ('hello' + '\n')

6. How large are all the headers in one packet? Give just a single total number. (Answer this for any one of your packets)

- 28 bytes for network and transport headers
    - 20 bytes for IP header, containing source and destination IPv4 addresses.
    - 8 bytes for UDP header (calculated previously)
- 48 bytes including applcation data (6 bytes) and linux "pseudo-header" (14 bytes)

7. Find the source IP address and the destination IP address from the IP header. Explain why they are 127.0.0.1?

- 127.0.0.1 is localhost, aka the loopback address. localhost is the address of the machine itself, meaning the source and destination IP addresses are the same. This makes sense because both the udpserver and client are running locally on the terminal.

In your submission, write down how many different protocols are visible with the filter active. Also, write down how many UDP datagrams your program should have sent, and how many it should have received. Make sure to record the input and output from your terminal as well to help the mentors. How many datagrams in Wireshark appear to be from either your udpserv01 or udpcli01 programs? Write down this number as well.

- I see UDP/XML and NTP protocols with the filter active.
- 2 datagrams are exchanged when sending a message through the programs - one from the client and one from the server.
- Since I'm running this on WSL, there's a ton of datagrams between the windows network and wsl network using NTP. There's also some datagrams to a remote IPv6 addres using UDP/XML. They outnumber the datagrams from the UDP prgoram by a lot.

Now sum every 16-bit chunk together. If you have an overflow in the most significant bit, simply drop the carry bit (so that we still have a 16-bit number), and then add 1 to your sum. Do this for each time you cause an overflow - you may add 1 many times!

- IP Pseudo Header:
  - Source Address: 127.0.0.1 -> 0x7f000001
    - 0x7f00 + 0x0001
  - Destination Address: 127.0.0.1 -> 0x7f000001
    - 0x7f00 + 0x0001
  - Protocol: UDP (17) -> 0x0011
  - UDP length: 14 -> 0x000e
- UDP Header:
  - Source Port 54801:
    - 0xd611
  - Destination Port 9877:
    - 0x2695
  - Length 14:
    - 0x000e
  - Checksum 0:
    - 0x0000
- Application Data ("hello\n"):
  - "he": 0x6865
  - "ll": 0x6c6c
  - "o\n": 0x6f0a

Add the pseudo header contents to your sum (using the same approach of adding 16-bit chunks and then dealing with overflow). Let’s call this the pre-checksum. Once you have computed the entire pre-checksum, take the one’s complement (make every 0 a 1 and every 1 and 0), and that’s the checksum.

- Pre-Checksum:
  - 0x7f00 + 0x0001 + 0x7f00 + 0x0001 + 0x0011 + 0x000e + 0xd611 + 0x2695 + 0x000e + 0x0000 + 0x6865 + 0x6c6c + 0x6f0a = 0x3eb3

- Final checksum:
    - pre-checksum: 03eb3
    - binary: 0011 1110 1011 0011
    - flipping: 1100 0001 0100 1100
    - one's complement: 0xc14c

Choose a packet with a short message and compute the checksum by hand (you can use a calculator, but for every addition, write down which two numbers you added). Include your answer and work in your submission PDF. Also write down if it matches the checksum that was in the datagram (it may not, due to a hardware feature called "checksum offloading", or due to arithmetic errors on your part).

- Packet's checksum: 0xfe21
- It does not match the checksum in the datagram

Finally, verify the checksum by adding it to the pre-checksum. What's the result? Keep your work, and write down your answer to this question as well.

- Calculated checksum + pre-checksum:
    - 0xc14c + 0x3eb3 = 0xffff