# CSCI 4220 Lab 1

## Lab 1: Getting Started / UDP Exploration

In this lab you will examine UDP packets in more depth and learn about the "Internet checksum", which is used by UDP.

### Getting Set Up

First, install Wireshark. You do not have to worry about USB captures, but you will need to allow installation of some sort of pcap library. For Windows users (including WSL users), follow the Windows instructions. If you are using Linux, check your distribution's package manager before resorting to building from source. In Linux you may have permissions issues - see this article for one possible fix. You should **not** run wireshark with sudo. If you are prompted to install Ncap do so. You do not need USBPcap. If Ncap installation asks which features to enable, make sure that "Support loopback traffic" is checked.

Note: If you are running Cygwin on Windows you will probably need to use something else - either WSL, a virtual machine running a Linux distribution of your choice, or a dual-boot into Linux.

Now get the source code that comes with the book. You can get this from the book's website or the Submitty Course Materials section. Make sure to extract it to a location you'll be able to get to easily, we'll use it many times throughout the semester. To install:
1. Run `./configure` inside the top-level source directory
2. Edit the `Make.defines` file and add `-fPIC` to the end of the `CFLAGS =` line
3. Run `make` in the `lib/` and `libfree/` directories.
4. In the `libfree/` directory you will probably get a compiler error or two. If you get an error about `inet_ntop.c:60:9: error: argument 'size' doesn't match prototype`, change the type from `size_t` to `socklen_t`
5. If you changed anything in step 4, rerun `make` in the `libfree/` directory
6. Navigate to the `udpcliserv/` directory and run `make` to build the UDP client server examples.

If you forget the `-fPIC` flag (i.e. forget step #2) you may have trouble with all string libraries, and will probably see some strange warnings.

### Initial Capture

Now start Wireshark and attach it to whichever interface you're currently using for internet access. This is done by clicking the interface under "Capture" (the bottom half of the welcome screen), and then going to the Capture menu and clicking "Start". Alternately, just double click the interface. In your terminal, run `./udpserv01 &` and then run `./udpcli01 127.0.0.1`. Send a few messages and then kill the client by pressing **Ctrl+C**. Run `fg` to recover the server and kill it as well by using **Ctrl+C**.

Stop the capture in Wirehsark. There's likely a lot of data - modern applications are very "chatty" - so we want to filter the data. Near the top is a long search box which currently says "Apply a display filter". Inside this box type udp and press Enter. Write down how many different protocols are visible with the filter active. Also write down how many UDP datagrams should your program have sent, and how many should it 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.

## Switching to Loopback

Since we're sending data over a local address (127.0.0.1), it never needed to go out to the Internet. Your computer makes a decision at the network layer about where to forward a packet, and in this case, the answer was simply to pass it from one UDP socket with a port number to another UDP socket with a different port number on the same host (machine). Instead, we need to be listening on a Loopback device. In Linux systems this is usually "lo", in Windows this usually "Ncap Loopback Adapter".

Repeat the capture process from before, sending the same messages, but this time choose the loopback adapter. Once data collection is done, stop the capture. Again, filter by udp. You'll *might* still see a surprising amount of other loopback traffic is happining on your computer. To narrow things down, change the filter text to `udp.port==9877` and press Enter. Now you should just see your packets. Do the numbers match up? Either way, record the number of packets you see.

Some packets may be labeled strangely with a protocol like `openSAFETY over UDP`. This is a problem with dissectors - ultimately data is just data and bits are just bits, so we might match the pattern for a protocol we don't actually intend to use (in this case, UDP on port 9877). To fix this, go to the Analyze menu and choose Enabled Protocols. Scroll down to `openSAFETY ov. UDP`, uncheck it, and click OK. You may see a different protocol ; but unless another program was already using port 9877 (in which case our `bind()` call should have failed), all traffic we captured on that port should just be ordinary UDP datagrams.

## Examining Packet Contents

Now that we have captured the packets (using the term "packet" loosely, since we're looking at a lot of layers), we can actually examine some of the details. Click on the packet corresponding to `udpcli01` sending the first message. Your Wireshark view should be split into roughly thirds, with a timeline and summary of packets in the top third, details about the packet in the middle third, and the raw data in the bottom third. You should be able to see an ASCII representation of your message, but the rest of the plaintext probably is unprintable or looks like garbage - because it's not meant to be read as plaintext. Keep in mind that the values to the left of the plaintext are in hexadecimal, and the addresses in the left margin are also hexadecimal.

The dissectors (the middle third of the screen) show information for each layer. If you expand the UDP dissection, you can see UDP details. If you click on the UDP summary (the line where you can expand/collapse the dissector information) it will highlight the raw hex in the bottom third that corresponds to the UDP header.

Write down answers to the following questions, using the information in Wireshark:

1. What was the port number on the client side? 9877
2. What was the port number on the server side? 55189
3. How large is the UDP header? 8 bytes
4. How large is the application data? (Answer this for just one of your packets) 6 bytes
5. How large are all the headers in one packet? Give just a single total number. (Answer this for any one of your packets) 34-6=28 bytes