ECE435: Network Engineering – Homework 5
UDP

Due: Friday, 6 March 2026, 5:00pm

For the first half of this homework (short answers) put your answers in the README file and it will be included when you submit your code.

1. **UDP**

   (a) The `tcpdump` program can record network packets. The following packet was gathered using the command
   `sudo tcpdump udp -XX -i eth0`

   The first lines show a summary of the packet. The rest is a hexdump of the packet. The leftmost column is the offset in hex. The next 8 columns are the hex representation of the bytes (in 16-bit chunks). The far right is the contents of the packet in ASCII (unprintable characters are shown as '.').

```
22:20:59.106555 IP macbook-air.43424 >
 google-public-dns-a.google.com.domain:
 57673+ A? www.adafruit.com. (34)
 0x0000:  0013 3b10 667f 0050 b647 1cde 0800 4500  ..;.f..P.G....E.
 0x0010:  003e e1ea 4000 4011 7fe6 c0a8 0826 0808  .>..@.@......&..
 0x0020:  0808 a9a0 0035 002a 9299 e149 0100 0001  .....5.*...I....
 0x0030:  0000 0000 0000 0377 7777 0861 6461 6672  .......www.adafr
 0x0040:  7569 7403 636f 6d00 0001 0001            uit.com.....
```

   The first part of the packet includes Ethernet and IPv4 headers that we don't know about yet. The UDP fields start at offset 0x22:

```
 0x0020:       a9a0 0035 002a 9299 e149 0100 0001  .....5.*...I....
 0x0030:  0000 0000 0000 0377 7777 0861 6461 6672  .......www.adafr
 0x0040:  7569 7403 636f 6d00 0001 0001            uit.com.....
```

   Using what you know about the layout of the UDP header answer the following questions:

   i. What is the source port (in decimal)?
   ii. What is the destination port (in decimal)?
   iii. What is the size of the UDP packet (in decimal)?
   iv. Are checksums enabled? How can you tell?
   v. What type of protocol is this / what is the packet doing? (note, the answer to this question is *not* UDP)

   (b) What is one reason to use UDP over TCP?

2. **UDP Client/Server Coding**

(a) Download the code from:
`https://web.eece.maine.edu/~vweaver/classes/ece435/ece435_hw5_code.tar.gz`

(b) Unpack the files:
`tar -xzvf ece435_hw5_code.tar.gz`

(c) Build the C files:
```
cd ece435_hw5_code
make
```

(d) Try running the `udp_client` and `udp_server`.

- The client sends a UDP message you type to the server over UDP (DGRAM socket).
- The client then waits for a response from the server, and if it gets none within 5 seconds it gives up and prompts for another message.
  It uses the `select()` system call to wait for data with timeout.
- The server receives the incoming UDP packet from the client using the `read()` system call and prints it to the screen. Since UDP is connectionless, it cannot reply using `write`!

(e) Modify the `udp_server` code to use the `recvfrom()` system call instead of `read()`.

i. A `recvfrom()` call looks something like below. The call receives the IP address and port number as part of the sockaddr structure which can be used to send a reply back to the client.

```
n = recvfrom(socket_fd,buffer,(BUFFER_SIZE-1),0,
                        (struct sockaddr *) &client_addr,
                        &client_len);

// structure definition of struct sockaddr * for reference
//struct sockaddr_in {
//    short           sin_family;   // e.g. AF_INET, AF_INET6
//    unsigned short  sin_port;     // port (remember in network order)
//    struct in_addr  sin_addr;     // struct holding the address
//    char            sin_zero[8];  // zero padding
//};
```

ii. Have the server print out the host name, host address, and source port of the incoming connection. (HINT: You can use the following code to get a string version of the hostname and address).

```
hostp = gethostbyaddr(
                (const char *)&client_addr.sin_addr.s_addr,
                sizeof(client_addr.sin_addr.s_addr), AF_INET);
hostaddrp = inet_ntoa(client_addr.sin_addr);
```

iii. You can use `sendto()` to send a response. `client_addr` will already be set from the incoming `recvfrom()` call.

iv. As with HW#1, uppercase the message before sending it back.

v. Also, as with HW#1, close the server and client once the message "bye" is sent

```
n = sendto(socket_fd,buffer,strlen(buffer),0,
                (struct sockaddr *)&client_addr, client_len);
```

3. **Extra Credit**

   - See if you can do your own capture of a UDP packet on your local network.
   - On Linux you can install the tool tcpdump to do this
   - On Windows/Linux you can install the wireshark tool which might have an easier to use GUI interface
   - Please use tools like this responsibly, though these days it's harder to spy on people than it used to be
   - Just include in your README a dump of the UDP packet you captured
   - One good source of UDP packets is your code from this assignment. In that case you probably want to specify the loopback device ("lo") to listen on rather than "eth0" (the first ethernet card) from the example in the first problem.

4. **Submit your work**

   - Please edit the README file to include your name.
     Also put your answers to the questions there.
   - Run `make submit` which will create a `hw5_submit.tar.gz` file containing the `README` and the udp code.
     You can verify the contents with `tar -tzvf hw5_submit.tar.gz`
   - e-mail the `hw5_submit.tar.gz` file to me (vincent.weaver@maine.edu) by the homework deadline. Be sure to send the proper file!