# CS380 — Project 3

January 30, 2017

Due: Wednesday, February 8, 2017 (50 points)

## Description

In this project, you will be implementing IPv4. I will have a server running that will accept IPv4 packets, verify the checksum, and reply with (in text) the word "good" if the packet is formatted properly, it can correctly verify the checksum, and the amount of data sent matches the expected size of the packet.

If the server detects a problem with one of your packets, it will respond with a line of text indicating what it thinks the problem is. If you receive an indication from the server that a packet was bad, the server will then close the connection, so don't try to send any more packets. Instead, restart your program and make a new connection for future packets.

In the following table, I will tell you whether or not to correctly implement a header field. If I say do not implement, simply fill that field with all zeros.

| | |
|---|---|
| Version | Implement |
| HLen | Implement |
| TOS | Do not implement |
| Length | Implement |
| Ident | Do not implement |
| Flags | Implement assuming no fragmentation |
| Offset | Do not implement |
| TTL | Implement assuming every packet has a TTL of 50 |
| Protocol | Implement assuming TCP for all packets |
| Checksum | Implement |
| SourceAddr | Implement with an IP address of your choice |
| DestinationAddr | Implement using the IP address of the server |
| Options/Pad | Ignore (do not put in header) |
| Data | Implement using zeros or random data |

Send the packets as sequences of bytes directly through the socket's output stream. The host will be the same as the previous project: `codebank.xyz` on port 38003.

For more information about the implementation of IPv4, see your textbook, the RFC specification at https://tools.ietf.org/html/rfc791, or the Wikipedia article about IPv4: https://en.wikipedia.org/wiki/IPv4.

Your program must send packets in the following way when run: you will send 12 total packets. The data size will start at 2 bytes, then double each time. The contents of the data do not matter. You can either fill the data with zeros or randomize the contents.

## Sample Output

When you run the program, identify each packet being sent either by number (packet 1, packet 2, etc.) or by data length (data length 2, data length 4, etc.) and then display the server's response. Here is one example of how it might look:

```
$ java Ipv4Client
data length: 2
good

data length: 4
good

data length: 8
good

data length: 16
good

data length: 32
good

data length: 64
good

data length: 128
good

data length: 256
good

data length: 512
good

data length: 1024
good

data length: 2048
good

data length: 4096
good
```

## Submission

1. On codebank.xyz, create a project as your user named `CS380-P3`. Follow this naming convention precisely including case.
2. Make sure you add the reference to the remote repository in your local project with:

   ```
   $ git remote add origin https://codebank.xyz/username/CS380-P3.git
   ```

3. Your project should have a main class named `Ipv4Client` in a file named `Ipv4Client.java`. You can have other files or classes, but it should successfully compile and run by simplying using:

```
$ javac Ipv4Client.java
$ java Ipv4Client
```