# CS 118 - Project 2

Alex Crosthwaite – Jacob Nisnevich – Jason Yang

June 6, 2016

## 1 Design

As a group, we implemented two classes to create our TCP packet abstraction, `Packet` and `Packet_info`. We also made the decision to *not* implement classes for the client and server for this project. Rather, we decided to simply execute the client and server code line-by-line in `client.cpp` and `server.cpp` respectively.

### 1.1 Packet

In our project, we used the `Packet` class for as our TCP packet abstraction. The class contained `SYN`, `ACK`, and `FIN` flags, in addtion to member variables for the sequence number, acknowledgment number, recieve window, and the data contained in the packet. The class could either be initialized with each of these values or with default values.

### 1.2 Packet_info

We also implemented a `Packet_info` class that we used to keep track of data length, the time the packet was sent, and the timeout time for the packet.

### 1.3 Client

### 1.4 Server

## 2 Problems and Solutions

### 2.1 Binary File Transfer Issues

### 2.2 Bit Vector Issues

### 2.3 Congestion Control Issues

### 2.4 Debugging Issues

### 2.5 Extra Bits at the End of Packets

## 3 Extra Credit

For the extra credit portion of the project, our group chose to implement TCP Reno fast retransmission and fast recovery as well as adaptive RTO.

## 4 Build Instructions

For Project 2 we did not modify the Makefile or Vagrantfile and just used the suggested virtual machine and g++ version. To build the project, simply initiate the two Ubuntu-based virtual machines with:

```
vagrant up
```

Then, `ssh` into both the client and server machines with:

```
vagrant ssh client
vagrant ssh server
```

Finally, build the project with:

```
make client
make server
```

# 5 Test Cases

## 5.1

# 6 Contributions

## 6.1 Alex Crosthwaite

## 6.2 Jacob Nisnevich

## 6.3 Jason Yang