# Distributed Systems and Algorithms CSCI-4510/6510 - Fall 2018
# Project 0 - Distributed Echo Service

This project is for self-evaluation of your knowledge of socket and multi-threaded programming. It will not be graded.

## Project Specification

For this project, you will implement a distributed echo service. The service consists of 3 servers. A `hosts.txt` file specifies the IP address and port number for each server. The contents of hosts.txt will look like (replaced with the appropriate IP addresses and ports for your machines):

```
127.0.0.1 7000
127.0.0.1 7001
127.0.0.1 7002
```

Each server is initialized with the location of the `hosts.txt` file and its own ID (position in the list of addresses).

After a server is started, it allows the user to enter text at the command prompt. When the user hits enter, the server sends an echo request message, containing the entered text, to every other server.

When a server receives an echo request message, it creates a new echo response message, containing the received text, and it sends this message (the echo) to the sender of the echo request.

When a server receives an echo response message, it displays the sender ID and contents of the message in the console.

An example execution of one echo at Server 1 looks like:

```
% Enter text:  Hello, world!
% Server 3 reply: Hello, world!
% Server 2 reply: Hello, world!
```

## Implementation Details

Your application should consist of a single executable that is run on each “server”. Every server should run the exact same code. You should test your service on at least two different computers (e.g., one server on machine 1, and two servers on machine 2).

When a server sends echo request messages to the other two servers, it should send these messages in parallel, and it should handle the echo response messages as soon as they are received. In other words, a server should not perform the echo request/response phase at the first other server, and then perform the echo request/response phase at the second server.

Advanced features:

1. Server failure: If you terminate one server (using Ctrl-C), the remaining two servers should still be able execute the echo protocol.
2. Server recovery: When you restart a failed server, the servers should be able to again execute the echo protocol using all three servers.
3. Use UDP sockets instead of TCP sockets. Since UDP messaging is not reliable, a server may not receive a response to its echo request. The server should not block indefinitely if this happens.