# PA#4: The Server Moved Out

Due: Friday 6/26/20 at 11:59pm

## Introduction

In this assignment, you are going to write primarily a class called `TCPRequestChannel` based on TCP/IP communication interface with a view to provide client-server communication across the Internet. The communication over a request channel is to be provided by TCP connection(s) that provide reliability guarantee (i.e., no data loss in transit). While such guarantee is trivial within a machine, it is non-trivila problem for programs communicating over the Internet, and hence the widespread use of TCP.

First, you are to modify the server program from PA3 to handle incoming requests over TCP channels while keeping. Make sure to keep a backup of the existing `FIFORequestChannels`-based communication because we will use that as a comparison baseline. The server must be able to handle multiple channels, either from the same client or from different clients, possibly on different machines. You also must modify the client so that it connects to a remote server when requested by the user.

## The Assignment

In this assignment, the client will no longer use `fork()` + `exec()` to run the server, because the server may not even run in the same machine. Therefore, irrespective of whether running inside the same machine or not, you need to use 2 separate terminals - one for the server and the other for the client. Similar to real servers in the Internet, your PA4 server will be persistent and continue to run with or without the client programs. Multiple instances of the client program, either from the same or from different client machines, would connect to the server simultaneously. Here is how you should run your programs. First, you start the server using the following command format:

```
./server    -m <buffer capacity> -r <port no of the process>
```

, where the port number "-r" is needed only if `TCPRequestChannel` is to be used which is indicated by "-i t". For instance, the following command creates a TCP server with buffer capacity of 500 bytes at port 1100:

```
./server    -m 500 -r 1100
```

.

Note that a server process needs a port number because that number distinguishes it from other processes potentially working as other servers in the same physical machine. For instance, the same OS/machine can host a HTTP server at port 80, a DNS server at 53 and your PA4 at another port. The port number for PA4 must be in range $[1024, 65535]$

because it is an unsigned short and the first 1024 ports are reserved by the OS for well-known services.

Once the server is up and running, you need to run the client in the same machine or in a different one as long as the two are "visible" and "reachable" from each other. You can use the `ping` command to ensure bidirectional (i.e., client to server and vice versa) network visibility. Just visibility does not ensure the ability to make TCP connection. For instance, you can `ping` the server `compute.cse.tamu.edu` from `linux2.cse.tamu.edu`, but TCP connections are not allowed between the two by the admin policy. Therefore, after `ping` is successful, you should attempt `telnet`, which sets up TCP connection at a given port between the two machines. Look up `ping` and `telnet` to see how they work.

While TCP connection works between client and server within the same host, you should really test it between separate hosts. For instance, the server could be your laptop and the client could be your desktop machine, where both are under the same home router (i.e., you can both `ping` and `telnet` between these). Another option is launching 2 virtual machines - one for the server and the other for the client. This requires enabling guest-to-host networking from the Virtualbox-¿Properties.

Once you decide where you want to run your client, use the following command format to launch it:

```
./client    -n <#requests/person> -p <# persons>
            -b <bounded buffer size>
            -w <# threads>
            -m <buffer capacity>
            -h <host name or IP address of the server>
            -r <port no of the server process>
```

Note that if the server happens to be running on the same machine, you can use either "localhost" or "127.0.0.1" for the host name. The user can choose to enter the IP address directly or give the host name, which your client program must be able to resolve correctly (i.e., you should be able to refer to your desktop by the host name and the IP address).

# What to Hand In

You are to hand in a .zip file that comprises the following files:

- Code: All necessary code files and the makefile.
- Report: Measure the performance of the system with varying numbers clients and write a report with the runtime numbers presented in graphs like PA3. Make sure to compare against FIFO communication. What is the maximum number TCP connections you can make? How does that number compare against FIFO channels? Try to explain the runtime difference you between TCP and FIFO.
- Video: Include a link in the report to your youtube video where you demo your PA. Do a quick rundown of the code and then demo in the same way you did demo your PA3.