Machine Problem 5: The Data Server Moved Out!

Introduction

In this machine problem we improve the request channel to provide communication across the network. Specifically, we allow the the client-side end of a request channel to reside on one machine, and the server-side end of the channel on another machine. The communication over a request channel is to be provided by a single TCP connection.

In order to establish request channels over the network, the interface of the request channel must be modified somewhat. Below is a definition of the `NetworkRequestChannel` class that supports request channels across machine boundaries:

```
NetworkRequestChannel(const string _server_host_name, const unsigned short _port_no);
/* Creates a CLIENT-SIDE local copy of the channel. The channel is connected
   to the given port number at the given server host.
   THIS CONSTRUCTOR IS CALLED BY THE CLIENT. */

NetworkRequestChannel(const unsigned short _port_no,
                      void * (*connection_handler) (int *));
/* Creates a SERVER-SIDE local copy of the channel that is accepting connections
   at the given port number.
   NOTE that multiple clients can be connected to the same server-side end of the
   request channel. Whenever a new connection comes in, it is accepted by the
   the server, and the given connection handler is invoked. The parameter to
   the connection handler is the file descriptor of the slave socket returned
   by the accept call.
   NOTE that the connection handler does not want to deal with
   closing the socket. You will have to close the socket once the
   connection handler is done. */

~NetworkRequestChannel();
/* Destructor of the local copy of the channel. */

string send_request(string _request);
/* Send a string over the channel and wait for a reply. */

string cread();
/* Blocking read of data from the channel. Returns a string of characters
   read from the channel. Returns NULL if read failed. */

int cwrite(string _msg);
/* Write the data to the channel. The function returns the number of
   characters written to the channel. */
```

You are to modify the data server program from MP3/MP4 to handle incoming requests over *network request channels* instead of *request channels*. The data server must be able to handle

multiple request channels, either from the same client or from different clients, possibly on different machines.

You also have to modify the client from MP4 to send requests over network request channels.[1] Use the same source code of the data server as in MP3 or MP4 (in file `dataserver.C`) to compile and then to execute as part of your program (i.e. in a separate process).

The Assignment

You are to write a program (call it `client.C`) that consist of a number of *request threads*, one for each person, a number of *worker threads*, and a number of *statistics threads*, one for each person. The number of persons is fixed to three in this MP (Joe Smith, Jane Smith, and John Doe). The number of data requests per person and the number of worker threads are to be passed as arguments to the invocation of the client program. As explained earlier, the request threads generate the requests and deposit them into a bounded buffer. The size of this buffer is passed as an argument to the client program.

Design your dataserver so that multiple instances of the client program, either from the same or from different client machines, can connect to the dataserver simultaneously.
The client program is to be called in the following form:

```
client -n <number of data requests per person>
       -b <size of bounded buffer in requests>
       -w <number of request channels>
       -h <name of server host>
       -p <port number of server host>
```

The data server is to be called in the following form:

```
dataserver -p <port number for data server>
           -b <backlog of the server socket>
```

What to Hand In

- You are to hand in one ZIP file, called `<FIRSTNAME_LASTNAME_MP#>.zip`, which contains a directory called `<FIRSTNAME_LASTNAME_MP#>`.

  For this machine problem, Jane Student would submit one ZIP file, called `JANE_STUDENT_MP5.zip`, which contains a folder called `JANE_STUDENT_MP5`.

  This directory contains all needed files for the TA to compile and run your program. (The expectation is that the TA, after typing `make`, will get two fully operational programs, one called `client` and the other `dataserver`, which then can be tested.) The directory should contain at least the following files:

  - Your implementation of the Network Request Channel, to be submitted in two files: `NetworkRequestChannel.H` and `NetworkRequestChannel.C`.
  - The updated client, with name `client.C`.
  - The updated data server, with name `dataserver.C`.

[1] You can use the client code from MP3 if you did not get MP4 to run. Clearly state this in your documentation and your report! Otherwise the TA will hate you.

  - Any other file that is needed to compile and test your solution, including the updated `makefile`.

- Submit a report, called `report.pdf`, which you include as part of the submission directory. In this report you present a brief performance evaluation of the system with varying numbers clients and sizes of the backlog buffer on the server.