Internet Programming (Assignment 3)

Unmesh Joshi (VUnetID: uji300) and Koustubha Bhat (VUnetId: kbt350)

14th October 2012

Chapter 1

Assignment Documentation

1.1 Platforms

1 C Programs:

- gcc version 4.7.0 20120507 (Red Hat 4.7.0-5) (GCC)
- Thread model: posix
- Target: x86_64-redhat-linux

2 Java Programs:

- java version 1.6.0-24
- OpenJDK Runtime Environment (IcedTea6 1.11.4)
- OpenJDK Server VM (build 20.0-b12, mixed mode)
- Target: x86 64-redhat-linux and i386 Debian (Ubuntu 12.04)

1.2 What to expect after *make*

A directory structure filled with executables at right place(s) as posted on the URL.

1.3 Explanations of programs

1 Paper Storage Server:
Requirements:

1. paperserver stores papers in main memory which paperclient will query.
2. paperclient has to call RPC procedure of servers to get its job done.
3. File size of papers can be of arbitrarily of any length.
4. Number of papers are unlimited.

Explanation: Paper server will maintain linked list of papers so that, clients can add any number of papers (only limited by main memory of server). Server is made stateless because subsequent "server procedure calls " from different clients should not affect each other's output.

We ensured following things:

- Server returns result using static variable.
- No other static variable is used so as to make server stateless.

Paper details' structures are stored in a linked list (heap memory). When server ends, this memory will be freed automatically (as process ended). FreeList need not be called.

2 Hotel Reservation Server: Requirements:

1. hotelserver should maintain records of the room bookings done in main memory.
2. Dates are out of scope.
3. There are three kinds of rooms - 1, 2 and 3.
4. Guest names can have spaces in them.
5. The client program does an RMI call to the server interface object to get its job done.

Explanation: IReserver interface is defined that outlines the functionality of the RMI server. ReservationHandler class does the actual job of maintaining the booking record. Reserver class invokes an object of ReservationHandler to implement the functionality of the IReserver interface.

The IReserver interface .class file is shared with the client program which uses this interface to make RMI calls to the remote object.

3 Hotel Reservation Gateway:
Requirements:

1. Communicate with clients written in any language on behalf of the RMI server.

Explanation: We used bare sockets to have communication between 'C' client and Java Gateway server. Gateway server will spawn new thread for each client request which is transmitted as a sequence of bytes over a socket. Our Gateway will validate the command so that invalid command will not take round trip to RMI server. Gateway could be implemented as thinner interface, forwarding client requests as is to RMI server. Advantage in latter case is that RMI server can be modified without having to change the gateway.

hotelgwclient is a simple 'C' client, which creates socket, connects to gateway. Then it writes the command (received from terminal) to the socket and waits for the response from gateway.

1.4 Answers to the questions

1 Answer to the Question A:
We can send *any number* of paper details to client in following way:

- Server will maintain a linked list of nodes (of PAPER details)
- Client will call the server procedure in a loop each time asking for the next record
- Server will fetch the record requested by client and will return it to client

Note that this will work for concurrent requests by any number of clients. (As the server procedure is *re-entrant*. It means that server is stateless however, client always asks server which node it requires.)

2 Answer to the Question B:
It is difficult to transfer strings of arbitrary size using Sun-RPC, because we cannot pass pointers as function arguments, we cannot share file descriptors. Basically servers and clients do not share any memory.

How can we solve ? We can implement internal data structures at server which clients will govern, so that we can call same RPC server procedure multiple times to send one file (in multiple chunks) or to fetch one file in multiple chunks.

We implemented following protocol: Client will send server following things:

- Author Names, Paper title
- File Size, Chunk Index
- Paper Number = -1 (to indicate that it's first chunk

Server will then

- Generate new paper number
- Create a new NODE for linked list
- Allocate memory of "File Size " bytes
- Copy the first chunk in this node
- Return generated paper index to client

Client is sending the data chunk by chunk. Server will each time fetch the correct node from its linked list (maintained transparently from client) and will copy the chunk data at the appropriate position as specified by client in ChunkIndex field.

3 Answer to the Question C:
We used *One-thread-per-client* server structure for gateway. Because, multiple clients should be able to use server functionality at the same time as RMI server is Multi-threaded. That is why gateway will spawn separate threads for each client-request and will send the commands to RMI server through sockets.

4 Answer to the Question D:
Client-server protocol can be as simple as follows:

- Gateway server will create a socket and wait for client(s).
- ANY client can *connect* to this socket and send the command string.
- Gateway will validate the command and will call Remote Method on RMI server.
- Gateway will return reply from RMI server "as is " to the client.

5 Answer to the Question E:
Files to be given to the corresponding users are:

- hotelserver: Class files of RMI server program, including the *server interface class* file.
- hotelgw: *server interface class* file and its own class files.
- hotelclient: *server interface class* file and its own class files.

— end —