Dis. Sys. Des. Assignment 3 Doc

Amir Saboury

For this assignment I removed every CORBA related code and libraries and user Web Services instead. For using Web services I choosed the first option which is using `javax.jws` package.

`javax.jws` is the framework to publish java methods as a web service. For this purpos we can use `wsgen` and `wsimport` commands to generate the WSDL file which contains the description of exposed APIs. Also it generates the corresponding java classes to use in the client. For the server I annotated the class that I wanted to create the web service for, then by using the `wsgen` command, in generated the WSDL file from the annotated source code.

I tested the project for its functionality and atomicity. I tried to create a user in a server and put a book in another server.
Then I reserved the book from the first server. I saw that the book reserved on the other server and it stored on the first server and everything worked fine. The output was the same as the previous assignment.
Here is the log of these scenario:

```
[concordia] (c.user.12) Reserving book
```

```
intro.to.algs:clrs for c.user.12
[concordia] (c.user.12) Authenticating
[concordia] (c.user.12) Book intro.to.algs:clrs not
found
[concordia] (c.user.12) Requesting to mcgill to
reserve intro.to.algs:clrs for c.user.12
[mcgill] (N/A) Recieved InterReserve-
intro.to.algs$clrs@0.8490283499823432
[mcgill] (N/A) Got request for InterReserve for
(intro.to.algs:clrs)
[mcgill] (INTER) Reserving book intro.to.algs:clrs
[mcgill] (INTER) Book intro.to.algs:clrs not found
[concordia] (c.user.12) NOT Found the book on
mcgill!
[concordia] (c.user.12) Requesting to polymtl to
reserve intro.to.algs:clrs for c.user.12
[polymtl] (N/A) Recieved InterReserve-
intro.to.algs$clrs@0.6352347139531923
[polymtl] (N/A) Got request for InterReserve for
(intro.to.algs:clrs)
[polymtl] (INTER) Reserving book intro.to.algs:clrs
[polymtl] (INTER) Book intro.to.algs:clrs is
reserved. Quantity: from 4 TO 3
[concordia] (c.user.12) Found the book on polymtl!
Reserving ...
[concordia] (c.user.12) Added reservation
intro.to.algs:clrs##polymtl@c.user.12 for user
c.user.12
```

I have a LibraryServerMain class which boots up all 3 instances of the Library. It also creates a config file which

contains the names that those libraries are bounded to and also their port numbers.
The config file will be used by libraries themselves to find other libraries' port number. Client and AdminClient will use that file as well. This main, also creates the web services and expose all the required functions for the client.

One object for each library will be created for each school. Each Library will be bounded to different port different port for UDP communications.

After booting up the Libraries, we can run Clients. The class Client contains all the functions that needed to be provided to students.
It has `createAccount`, `reserveBook` and `interReserveBook` functions. It also has an internal function for finding the appropriate remote library object to connect to. There is also a `test` function which creates bunch of users and reservations for them just for test.

AdminClient is also used to connect to remote objects, but it provide the functions that admin has permission to run. It also provide the function for changing duration of a reservation which should be used for debugging purposes. The functions are: `setDuration` and `getNonRetuners`.

Both Client and AdminClient also have a main function that uses a command line interface to call their provided functions with custom arguments.

By invoking `interReserveBook` on the client, the client run this command on the corresponding school that the current user is registered in. The school itself first looks up the requested book in the local repository of books, if it can find that book, the procedure will be the same as `reserveBook`. If not, it will send UDP requests to other schools to see if the book is avaiable there. If the book is available on other schools, one of them will be selected (based on the order that we can define) and the quantity of the book will be decreased and a message regarding the reservation will be sent back to the main school. The main school then create a virtual book reservation record for the student and make the reservation for him/her. If any exceptions occur in evey step, it is going to be catched and cause all the previous actions to roll back.

After invoking a `getNonRetuners` call to a remote library, the library itself gets the list of the books, for each book, first it locks it for parallel access, then it iterate over all students that reserved the book and list all of them which their reservation period is less than specified amount of days.
Then the library sends UDP packets to other libraries to get their list as well. Then all the results will be merged and will be sent to the requester.

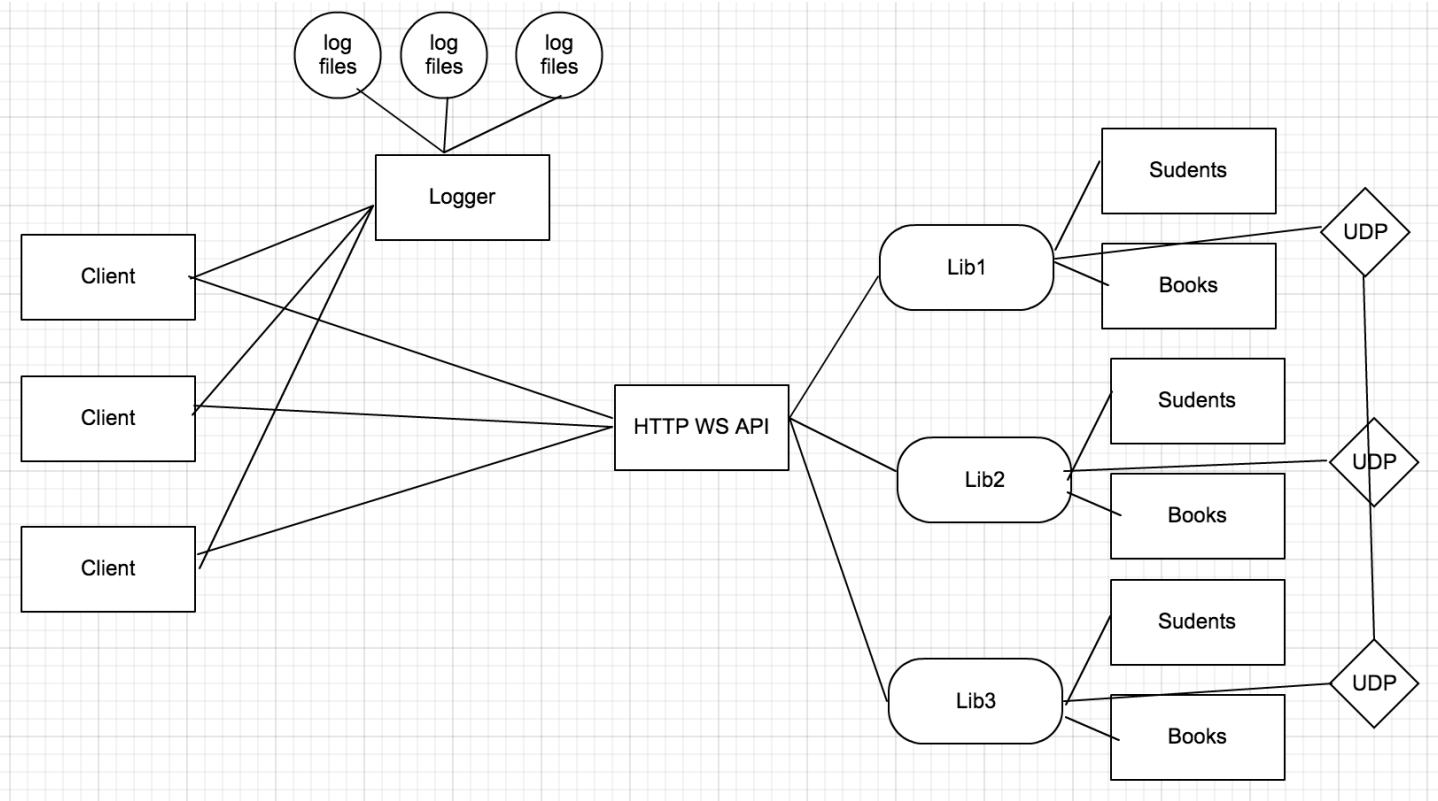
Each library has its own UDP server to answer to other request from other libraries. The server is running on a different thread. If a message come that starts with `nonRetList-` followed by a number, it means the requester needed the list of non-returners for the specified amount

of days, after computing the list, it will be sent back to the server that initiated the request.
If the mssage starts with `InterReserve-` this means another library wants to see if an specific book is available on this library or not. The message will be followed by that specific book's name and author.

All the UDP transportation and servers are encapsulated to the class `UDPTransporter`. This class has two methods, `transport` for sending a UDP packet and `server` to create a UDP server. The server gets a function as a server function which inputs the receiving string and outputs the reply. The server passes every coming message to that function and will send back the result to the requester.

For accessing the data of a student, I used an array of HashMaps: `HashMap<String, Student>[] students = new HashMap[26];`. Each student will have a key, in this case the key of each student is the first character of its username. For accessing a specific student, first we lock the appropriate HashMap in the array by looking its key, then by getting feeding the student username into the hash map, we can have the student.

All the Logging are done by a class called `Logger`. The logging are done by writing the log information into the appropriate file (based on the username of requester's username) and also by printing the log info on the standard output. The name of the Institution or `Client` or `AdminClient` also will be printed with the log info.


log files
log files
log files
Logger
Client
Client
Client
HTTP WS API
Lib1
Lib2
Lib3
Sudents
Books
Sudents
Books
Sudents
Books
UDP
UDP
UDP