UNIVERSITÉ Concordia UNIVERSITY

COMP 6231 DISTRIBUTED SYSTEMS DESIGN
SUMMER 2021

DESIGN DOCUMENT FOR ASSIGNMENT 1

TEAM SMARTY PANTS
BINIT KUMAR - 40172005
ABDUL AZIZ RAZA SHAIK - 40182726

CONCORDIA UNIVERSITY
MONTREAL, QUEBEC, CANADA

JUNE 2021

Introduction

Distributed Class Management System (DCMS) using Java RMI :

Distributed Class Management System is a distributed system used by center managers to manage information regarding the teachers and students across different centers.

The three centers locations are:

- Montreal (MTL)
- Laval (LVL)
- Dollard-des-Ormeaux (DDO)
-

The server for each center (called CenterServer) maintains a number of Records. The two types of Records are:

- TeacherRecord
- StudentRecord

A Record a unique RecordID starting with TR - TeacherRecord or SR – StudentRecord.

Fields in TeacherRecord – First Name, Last Name, Address, Phone, Specialization and Location.

Fields in StudentRecord – First Name, Last Name, Courses Registered, Status and Status Date.

Managers with a unique ManagerID can perform operations :

- createTRecord
- createSRecord
- getRecordCounts
- editRecord

This application has a number of CenterServers (one per center) each implementing the above operations for that center and ManagerClients (one per center) invoking the manager's operations at the associated CenterServer as necessary.

It has been developed using JAVA RMI

Description of the technique used:

JAVA Remote Method Invocation (RMI):

DCMS has been designed using Java RMI which is the object oriented equivalent of remote procedure calls (RPC). RMI helps in creating a distributed application by allowing an object to invoke methods running on a separate JVM.

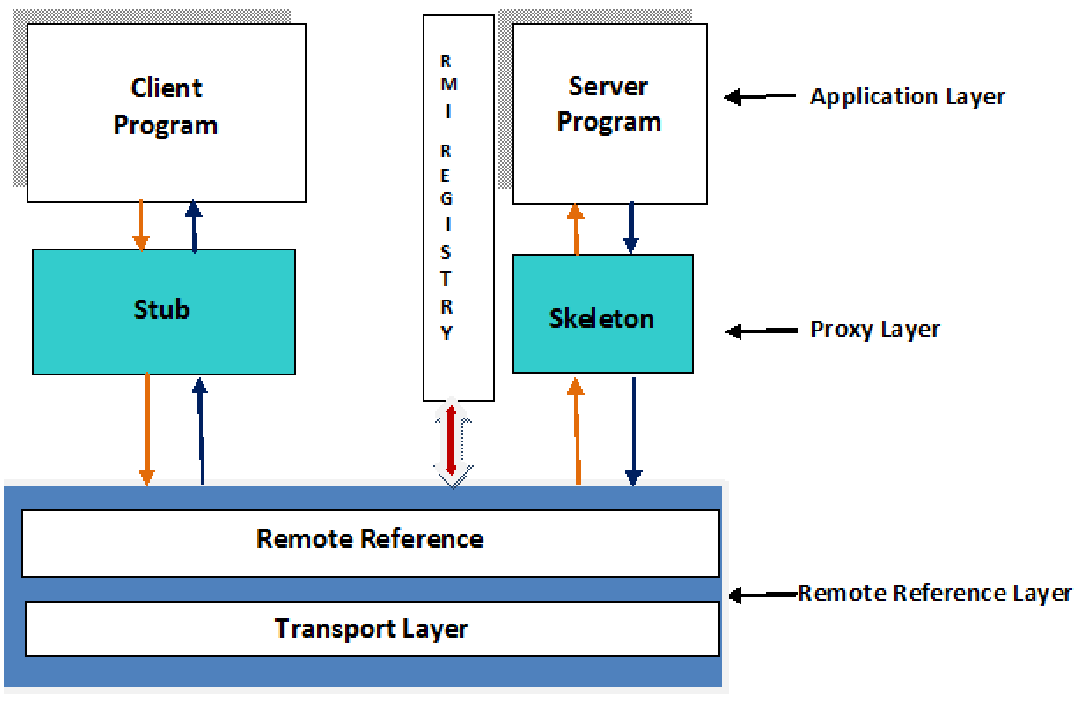


Fig: RMI Architecture

As seen in the figure:

Every RMI has two programs: the client program and the server program.

Client Program: “Stub” acts as a gateway for the client side, requests for the methods running on the server side are routed through this program.

Server Program: “Skeleton” acts as a gateway for the server side, incoming requests from the remote objects in client program for access to its method are handled here and made available to the client using a remote registry by its references.

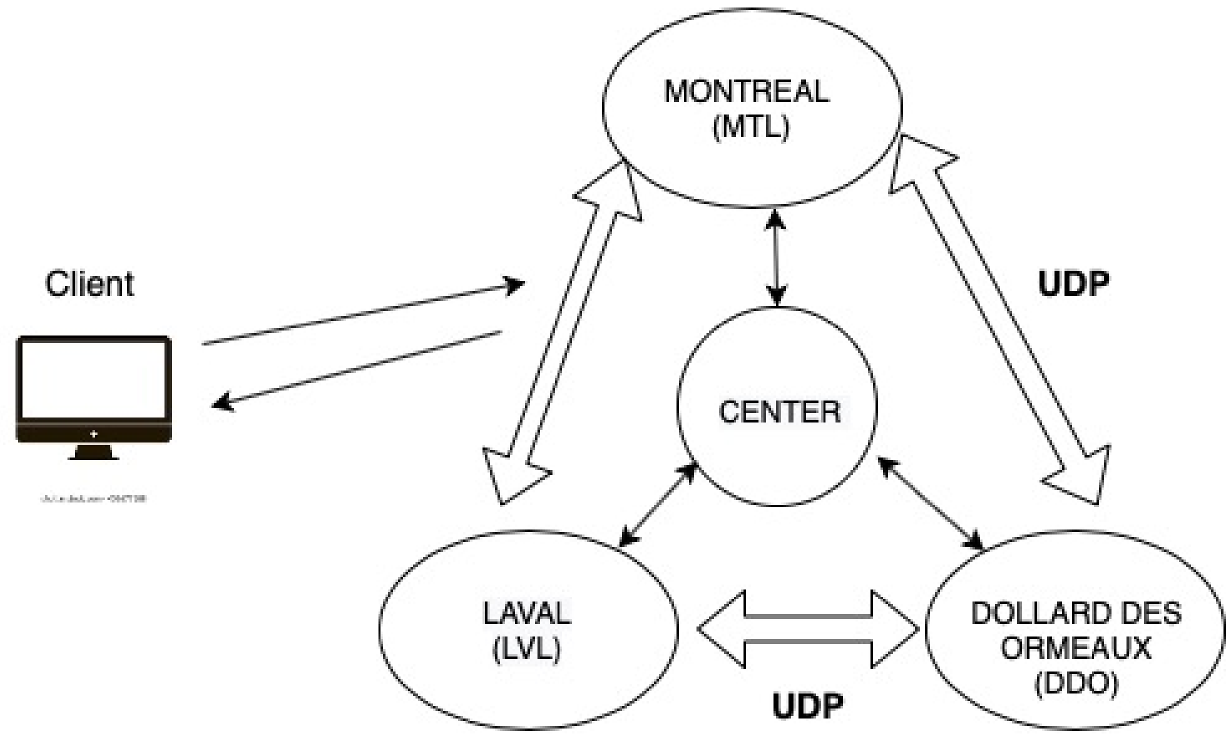
Architecture of the system:

The DCMS architecture has 3 locations (Montreal, Laval and DDO) which are the clients of the system.

The Managers at these locations have access to the database which consists of the records: Student Record and Teacher Record. The managers can perform functions such as createTRecord, createSRecord, getRecordCounts and editRecord as per requirement.

ManagerClient is the client program which handles these requests and attempts to create or edit the appropriate record within the corresponding server associated with the manager in the form of a hashmap and each server also maintains a log containing the history of all the operations that have been performed on that server in a separate text file. By invoking getRecordCounts, the manager of a center can also get the number of record (both teacher and student) present in their respective database.

This system has multiple centers and one CenterServer for each of them which make use of a centralized ManagerClient to invoke the required method from the repository, for each operation it finds the information about the requesting server and invokes the corresponding operation.



TEST CASES EXECUTED:

D. Test Scenarios:

ID	Description	Expectation	Result
1	Manager creating Teacher records	Record should be saved in database	PASS
2	Manager creating Student records	Record should be saved in database	PASS

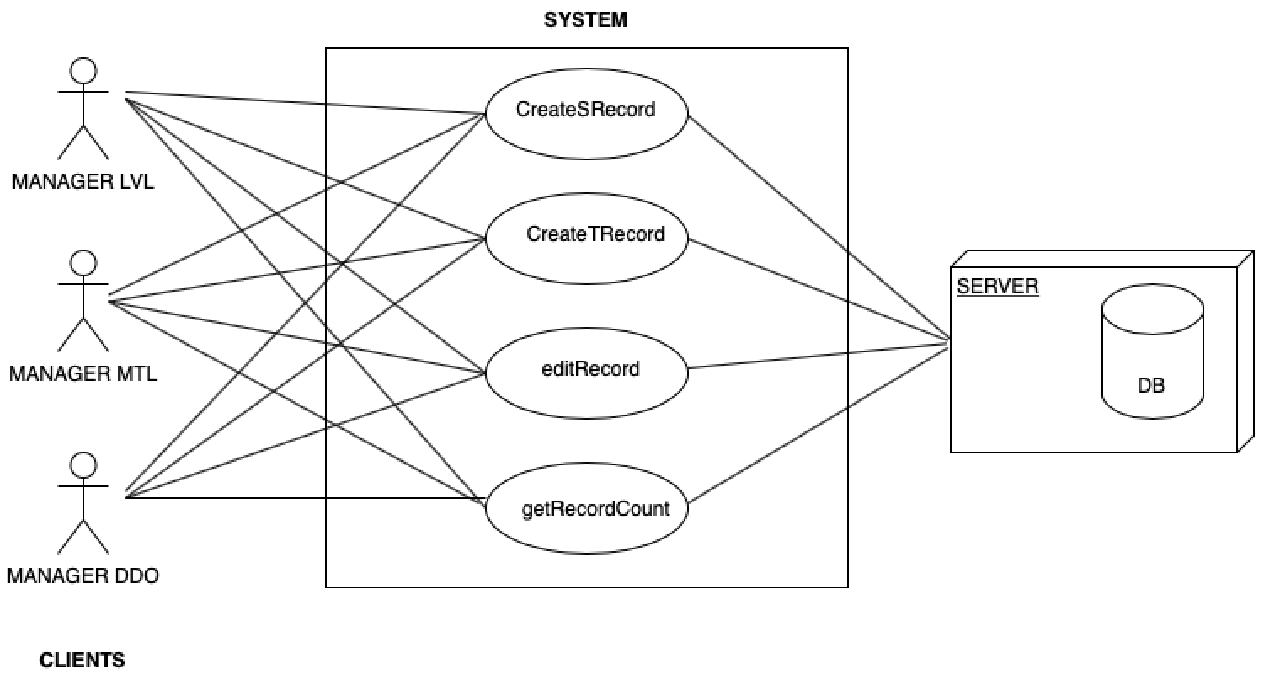
3	Managers requiring valid id to access the server Test Data: (MTLXXXX / LVLXXXX / DDOXXX)	Managers should be able to connect to the right server	PASS.
4	Managers not able to connect with invalid id	Invalid managers id will get an error message	PASS
5	Managers able to get all the record counts	The total count of all the records should be printed for each of the servers	PASS
6	Manager should be able to edit records	Manager should be able to modify records from the record id	PASS
7	Phone number validation in Teacher Records	Only numeric values in Phone number	PASS
8	Status Field validation in student record	Only Active and IActive status	PASS
9	Managers should not be allowed to edit any fields other than Address, Phone and Location in the Teacher Record.	Any fields other than Address, Phone and Location in the Teacher Record are not modified	PASS.
10	Managers should not be allowed to edit any fields other than Course Registered, Status and Status Date in the Student Record.	Any fields other than Course Registered, Status and Status Date in the Student Record are not modified	PASS

CHALLENGES FACED:

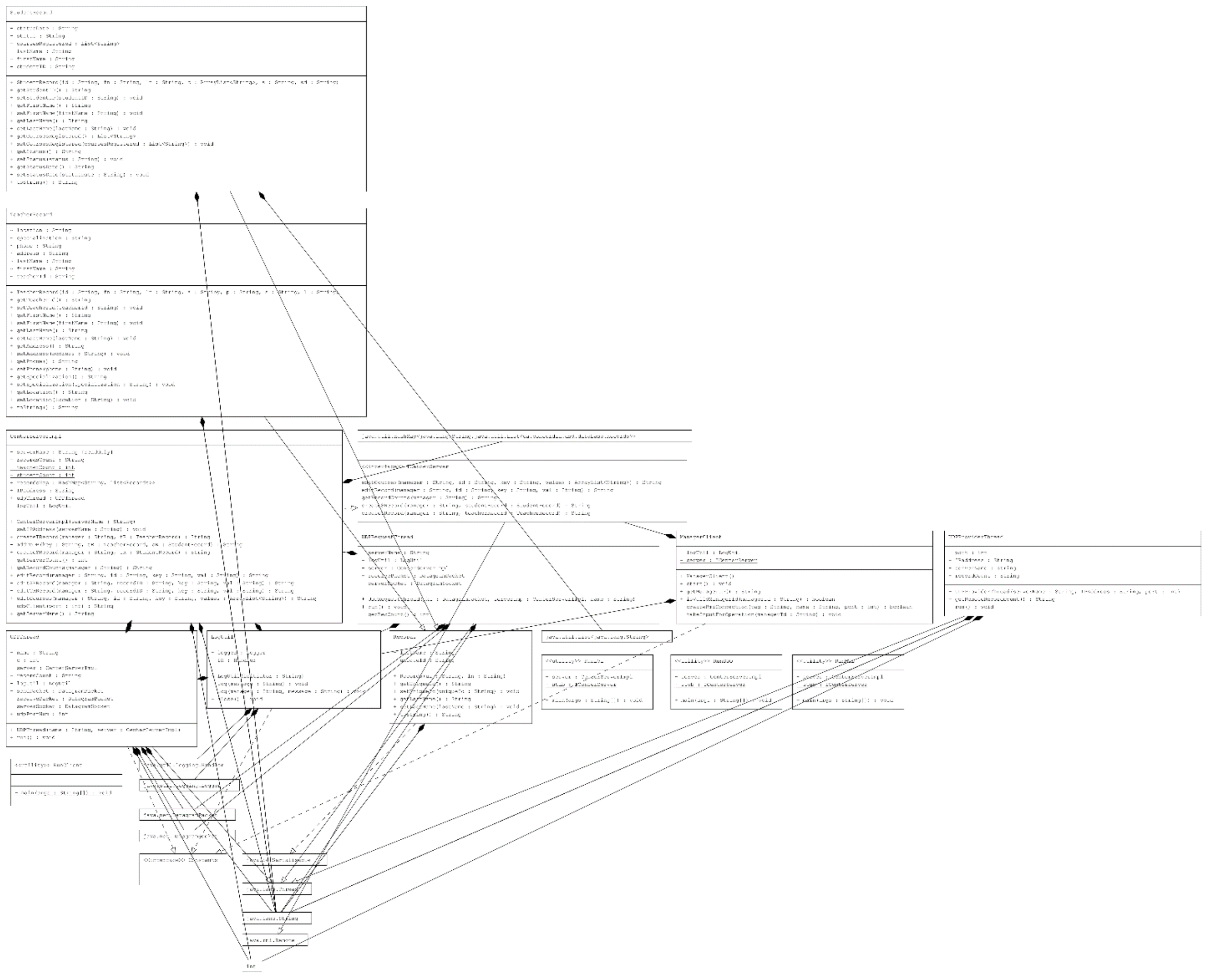
- The most difficult and important part of the design and execution was creating the correct client-server architecture which would work exactly as given in the problem statement.
- Another challenge faced by us was to understand the specifics and limitations of the RMI system and use it to implement our proposed architecture without any shortcomings in the required features.
- Third difficulty we faced was to establish the User Datagram Protocol (UDP) and IP sockets for the communication and data exchange between the center managers and the central server which contains the records.
- The logging of each execution into a text file also was a challenging part.
- Getting the whole distributed system to run on a single system using different ports and sockets for the development purpose was also a tough task.

UML Diagrams:

Use case diagram:



Class diagram



References:

- George Coulouris, Jean Dollimore, Tim Kindberg and Gordon Blair, "*Distributed Systems Concepts and Design"*, Fifth Edition, Addison-Wesley, 2012, ISBN: 0-13-214301-1.
- M. L. Liu, "*Distributed Computing: Principles and Applications*", Pearson Addison-Wesley 2004, ISBN 0-201-79644-9.
- https://www.javatpoint.com/RMI
- https://en.wikipedia.org/wiki/Java_remote_method_invocation
- https://docs.oracle.com/javase/tutorial/rmi/index.html
- https://www.codejava.net/java-se/networking/java-udp-client-server-program-example