2023/2024

USTHB University
Computer science faculty
Software Engineering

CAGL project

Important:

Practice is very important to learning the software engineering skills relating to requirements specification, design, testing, etc. Copying is considered serious offense. Both the person copying and the one supplying the copied material will be penalized.

Tools:

- JRE System library [Java SE].
- Eclipse IDE for Enterprise Java and Web Developers (J2EE) – version: 2023
- WildFly 27.0 Runtime Server.
- TCP/IP Network.

Objective:

The goal of this work is to implement applications that are developed and executed over a network using EJB3 technology, TCP sockets, UDP sockets, and RMI

Architecture:

We consider the architecture of a distributed application which is composed of (see Figure 1):

- Five servers (***Serv$_i$***). Each server must handle a group of service (S1, S2….Sn).
- Six clients (***Client$_i$***). Each client has a reference chain that represents a service requests.
- A central server (***Central Server***) with a database that hosts services.
- An intermediate server (***Inter***) that redirects client requests to the corresponding servers (***Serv$_i$***).

Process:

Services are stored in a database. The database contains various services for all clients. Services are identified by a *number, name* and *description*. Before a server (***Serv$_i$***) start communicating, it must register with the central server by providing the necessary information (IP address, server name, communication port).

Each client sends its various requests to the intermediate server (***Inter***) which redirects these requests to the associated server (***Serv$_i$***). Each server (***Serv$_i$***) calls the ***Central Server*** which performs service search and retrieval from the database. Information regarding the requested service must be sent back to the concerned client (Since each client has a chain of references, a query is sent by reference).

To send their requests, all clients use a sequential method using a rotation token. Clients are initially organized into a unidirectional logical ring. Only clients that receive the token can send its next request. It is important to monitor the token (for example, if it is lost due to a network interruption or station failure). To do this, assume that each station has a ***TIMER*** that is reset when a token passes through it. When the station stops sending token, the ***TIMER*** will reach the end. The first station whose ***TIMER*** expires launches the algorithm to select a new candidate. It sends deletion information that moves around the ring and issues a new token.

Work

To show how it works, we consider the following client reference chains.

Client$_i$	Reference chain
01	S7, S10, S1, S2, S4, S3, S0, S6, S12, S11, S8, S5, S2, S1, S0, S5, S1, S7, S9, S2.FIN
02	S15, S1, S4, S2, S3, S6, S7, S10, S1, S13, S6, S14, S2, S0, S1, S3, S2, S7, S0, S1.FIN
03	S10, S3, S6, S4, S2, S7, S0, S3, S12, S8, S7, S0, S6, S5, S3, S2, S1, S7, S0, S4. FIN

04	S13, S2, S1, S7, S0, S4, S6, S5, S11, S0, S14, S3, S9, S1, S0, S3, S6, S7, S12, S11, FIN
05	S2, S14, S1, S7, S6, S3, S0, S5, S2, S15, S7, S0, S4, S12, S3, S6, S7, S2, S1, S13. FIN
06	S5, S4, S1, S3, S10, S7, S6, S0, S13, S2, S1, S7, S6, S3, S15, S2, S1, S14, S5, S6. FIN

Server$_i$	**Service processing**
01	S0, S1 S2 S3
02	S4 S5 S6
03	S7 S8 S9
04	S10 S11 S12
05	S13 S14 S15

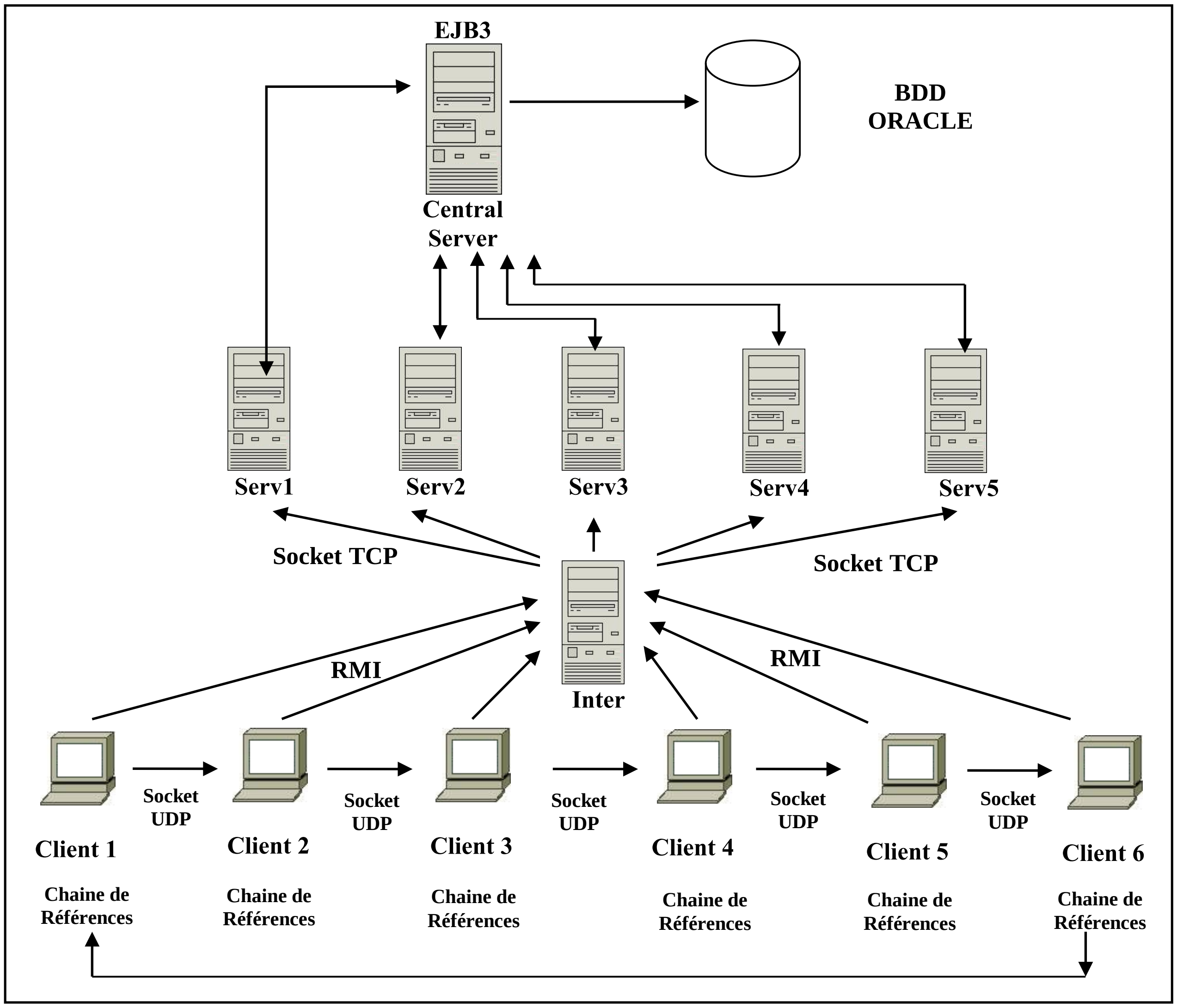

Figure 1 : Application architecture.

The work consists of implementing the system described above.