Distributed Network Application Development

Ștefan Cristian Mladin Popescu Răzvan

June 22, 2015

Professor: Costin Bădică
When: Tuesday 8:00
Web: Distributed Network Application Development

Abstract

This document describes the most important features of our proect. It contains information about architecture, implementation. It also contains some of the methods specified in pseudocode.

1 Statement

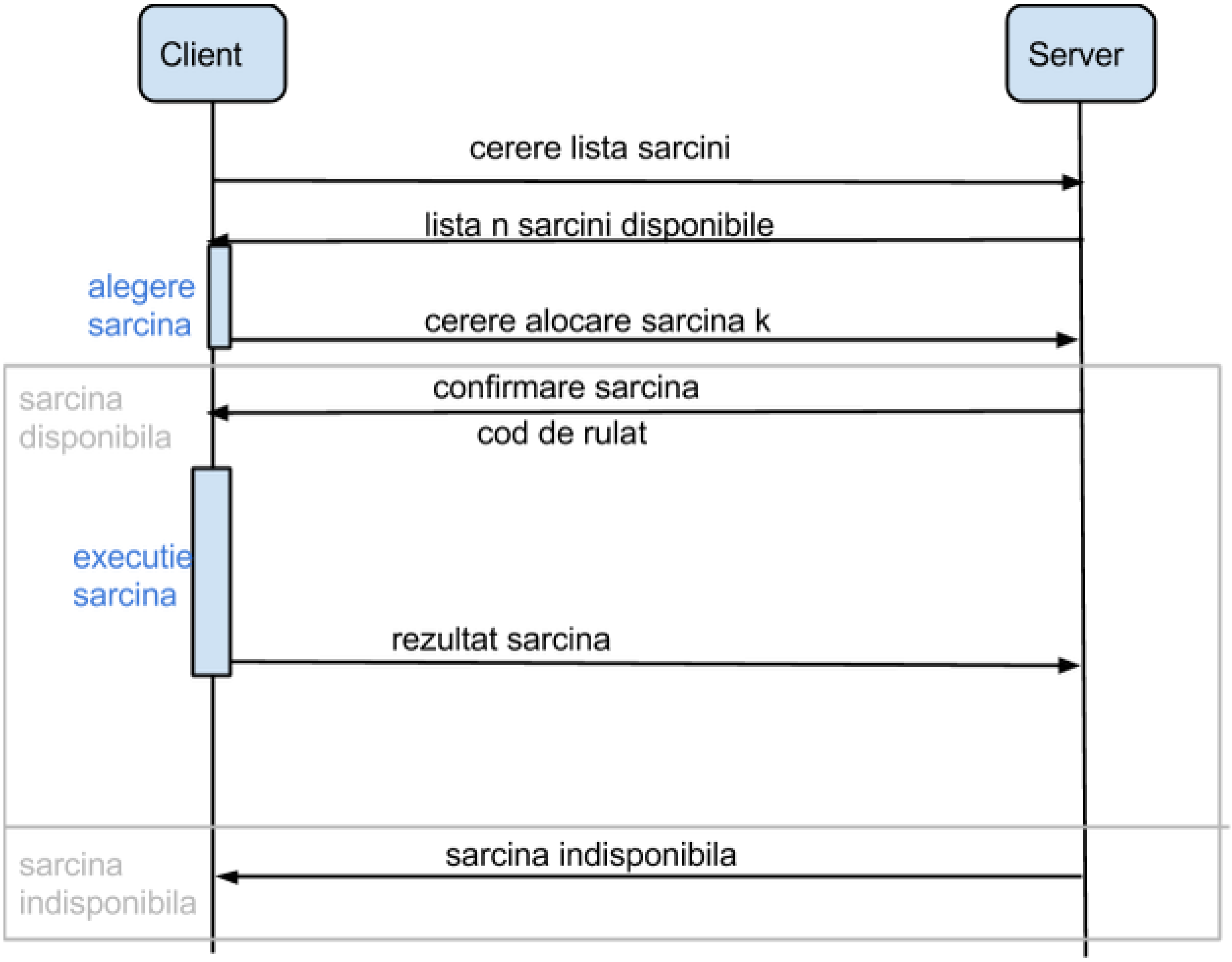

Figure 1: Required client server communication protocol.

The project is, in essence, a client – server application that is meant to create the possiblity of using the clients' resources to execute tasks. It is considered the situation of a research group in which there is needed more computing power and the team is willing to use their home computers in order to achieve that. Considering this, the client application can be installed on any computer and, after it is connected to the server it will receive a task, execute it and then send the result back to the server. The server application will have at a list of tasks and when the clients asks for one, it will send the most suitable one to him as a java file that the client will then execute on it's machine. A more detailed explanation of the communication protocol between the server and client is shown in figure 1.

From this specification we have extracted the following use cases for the application. They are represented in figure 1.

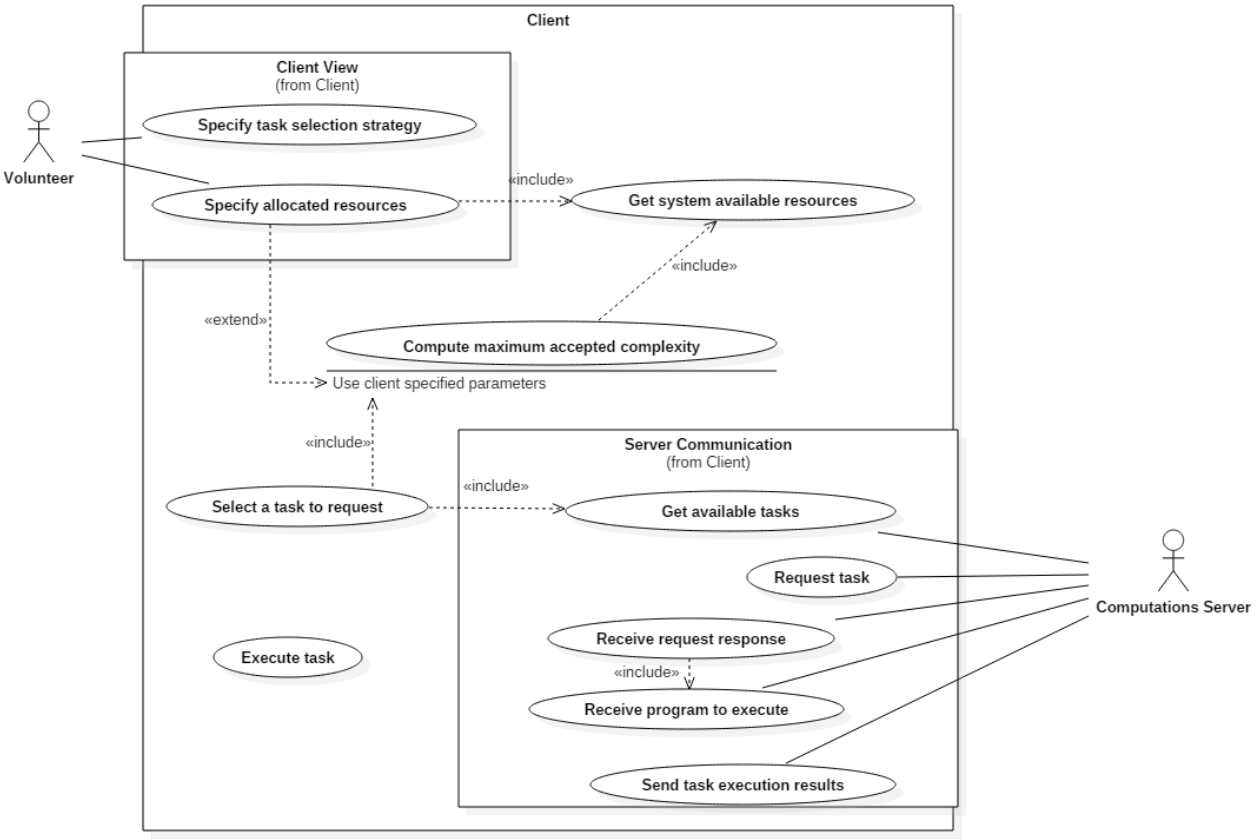

Figure 2: Use cases.

2 Algorithm Description

```
Data: selection strategy
initialization;
establish connection to server;
send request for list of tasks;
wait for list of tasks;
send a task assignment request to the server;
send the result to the server and a finished request;
```

Algorithm 1: Client algorithm.

```
Data: request, task
initialization;
while TRUE do
    if request is REQUEST_TASK_LIST then
        send task list;
    end
    if request is REQUEST_TASK_ASSIGNMENT then
        if task is assigned then
            send UNAVAILABLE signal to client;
        else
            assign task;
        end
    end
    if request is FINISHED then
        close connection with client;
    end
end
```

Algorithm 2: Server Response Algorithm

2.1 Application design

2.2 Logical Model

The entities of our application are the Task, Tasks and RequestType classes(represented in figure 2.2)

The RequestType is used to identify request messages that are passed between the server and clients(see 2).

The Task encapsulates information about the task(e.g. date, complexity) as well as means to set them and access them The Tasks class provides means of getting the tasks.

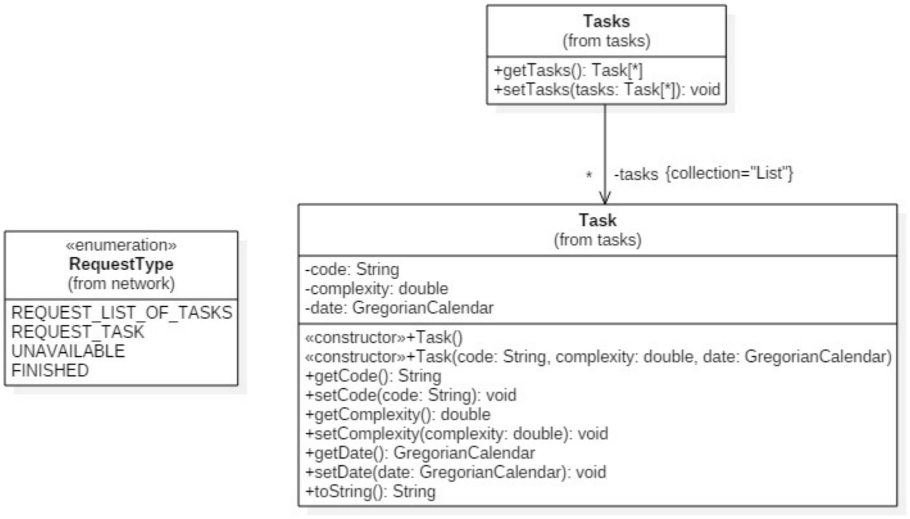

Figure 3: Use cases.

2.3 Client and Server Component Classes

We have specified two components for our project: the client and the server, both implemented by the classes shown at fig.2.3. The components will be deployed on the client and server nodes respectively.

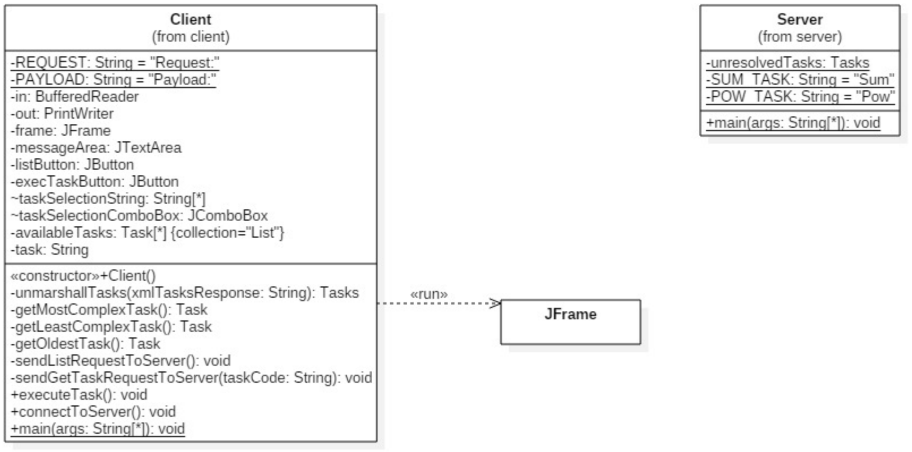
Client
(from client)
-REQUEST: String = "Request:"
-PAYLOAD: String = "Payload:"
-in: BufferedReader
-out: PrintWriter
-frame: JFrame
-messageArea: JTextArea
-listButton: JButton
-execTaskButton: JButton
~taskSelectionString: String[*]
~taskSelectionComboBox: JComboBox
-availableTasks: Task[*] {collection="List"}
-task: String
«constructor»+Client()
-unmarshallTasks(xmlTasksResponse: String): Tasks
-getMostComplexTask(): Task
-getLeastComplexTask(): Task
-getOldestTask(): Task
-sendListRequestToServer(): void
-sendGetTaskRequestToServer(taskCode: String): void
+executeTask(): void
+connectToServer(): void
+main(args: String[*]): void
«run»
JFrame
Server
(from server)
-unresolvedTasks: Tasks
-SUM_TASK: String = "Sum"
-POW_TASK: String = "Pow"
+main(args: String[*]): void

Figure 4: Use cases.

The client is responsible to providing the visualization of the application, where the user selects the strategy, views tasks and volunteers to offer his resources for the server.

The client selects the task according to the specified strategy and available tasks. It then requests the task from the server and handles the unavailable response(the task is not available) or the approval response, where the task is assigned to that particular client

The client also handles building the java files and executing them. The client then sends the output back to the server.

The server handles assignation of tasks to several clients. It is able to handle requests from multiple clients, such as: task listing requests, task assignation requests, task finalization notifications.

References

[1] Costin Bădică *Distributed Network Applications Course Material.*

[2] Sorin Ilie, *Distributed Network Applications Project Specification.* https://docs.google.com/document/d/13OOOryRr-1HhFIO_v1s18dhf5tRSjeMVkgWo2Ae4vsA/edit?pli=1, accessed in June 2015.

[3] George Coulouris, Jean Dollimore, Tim Kindberg and Gordon Blair, *Distributed Systems Concepts and Design* , 5 th ed. , Addison Wesley, 2011

[4] Leslie Lamport, *LaTeX: A Document Preparation System.* Addison Wesley, Massachusetts, 2nd Edition, 1994.