Distributed Systems
Project assignment - Building a middleware

Hadrien Croubois
hadrien.croubois@ens-lyon.fr

All documents are available on my website: `http://hadriencroubois.com/#Teaching`

This poject is due for april 17, 2015, 23h59, Lyon's time (UTC+2).

Part 1

Introduction

Middlewares are pieces of software which are used to enable various components of a distributed system to communicate and manage data. In our case we will focus on a middleware that provides an interface to remotely deploy job according to the RPC[1] paradigm.

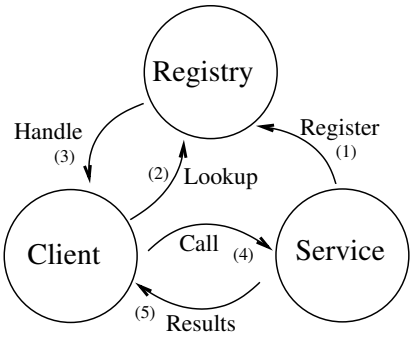

Figure 1: The GridRPC paradigm

In this paradigm, different computing instances providing services will register themselves to a registry. Later on, when a client wants a computation to be performed, he contacts the registry that then directs him to the adequate computing instance.
This paradigm is used by many middlewares such as DIET, NINF or GRIDSOLVE.

Part 2

Assignment

Now about the assignment. The objective here is to build a simple middleware architecture and you will not have to go through all the steps involved in the GridRPC model. Also, note that fulfilling all the proposed features, however interresting it is, is not required to have a good mark. There are many things to do, so if you fill uncomfortable with some feature, fell free to skip them and implement others. Don't be stressed if don't answer all the question, rather choose the one you feel interresting and capable of doing in order to build your own solution at an objectively very complex problem.

Agent deployment

The first job in deploying a middleware is to design agents that will run on the different machine and communicate with one another. Once one or multiple agents are deployed, you should be able to add new nodes or remove existing nodes while maintaining a coherent topology.
Here are some feature that you could want to implement. Most of those are voluntarily vague, which means that you will have to take decisions based on what you feel is appropriate and what you feel confident to implement.

[1] Remote Procedure Call

Question 1

a) [**Simple**] Spawn a agent that wait for remote command;

b) [**Simple**] Have this agent be part of a coherent topology that allows command to pass through;

c) [**Simple**] Be able to have a new agent join an existing topology;

d) [**Medium**] Have the possibility to remotely kill an agent while maintaining the coherent topology among the remaining agents;

e) [**Hard**] Have the possibility to detect and react to a agent dying without prior notice.

Using the deployed agent for remote job execution

Middleware agent are interresting in that they can be used has a middleman to deploy jobs to remote machines. Now that you have a set of agents, we will use it to distribute work.

Question 2

a) [**Simple**] Have your agents be able to receive job request and send back the results to the client once the job has been executed;

b) [**Simple**] Have your agents keep a waiting list of jobs to be executed so that no more than one job is performed simultaneously on this specific node;

c) [**Medium**] Have a voting mechanism that selects where to execute the job depending on the current status of the nodes waiting lists;

d) [**Medium**] If you implemented the possibility to remotely kill nodes, all jobs in their waiting list gets redistributed to the other nodes in the topology;

e) [**Hard**] Have a mechanism that makes free node fetch jobs from other nodes waiting lists.

Agent monitoring

In interresting tool to have is a monitoring tool that is able to interact with a topology of agent. This agent could:

Question 3

a) [**Simple**] Given an hostname, remotely spawn an agent on that node and have it join the topology;

b) [**Simple**] Query the topology for statistics;

c) [**Simple**] Submit job to a specific node in the topology and get the result value returned once the job is complete;

d) [**Medium**] Submit job to be executed on any node in the topology and get the result value returned once the job is complete;

e) [**Medium**] Build a script that, given a set of nodes, automatically deploy the platform so that it is ready to accept new jobs.

Going further

Here are some further idea:

Question 4

a) [**Simple**] Have multiple monitors connect to the same agent topology and submit tasks;

b) [**Simple**] Give priority to tasks so that more critical work can bypass part of the waiting list;

c) [**Hard**] Have a deadline mechanism that automatically kills jobs if they take to long to compute;

d) [**Extra**] Feel free to discuss and implement any other feature that you might find interresting.

Using your middleware

A middleware is just a tool !

Question 5

a) **[Extra]** Propose an usage for the features you just developed and implement it. Be creative !

Part 3

Handover details

Apart from the code itself, you are encouraged to to provide a `.pdf` report detailing the choice you made. I'd recommend this report to be written using LaTeX, but you are free to use applications like Microsoft Word or Open Office (as long as you export it as a PDF). This report, and all your code must be provided put in a `.tar.gz` archive named `name.tar.gz` in which you must have a folder `name` (replace `name` by your name !). You are also encouraged to provided detailed information on how to deploy and test you code, or even better, have a script do it automatically !

Your code is your property, protect it with a license ! Also, when sending your archive don't forget to provide us with a `md5` checksum so we can verify its integrity.