Laboratory Goals / Objectives

The purpose of this lab is to investigate the aspect of service provisioning in distributed systems. At the end of the lab, an understanding of the service provisioning concept should be demonstrated, by providing a working simulation of the implementation.

Overview

In distributed systems, remote services can stop, resume, exit or simply break down. Therefore, it is very important for the client to learn not only about the availability of a service, but also to be notified if something happens to that service while it is processing its request. If the client is not aware of any incident occurred on the server side, it might wait indefinitely for results that will never be delivered.
The client can subscribe to an event notification service of the service. After subscribing, the client will receive notifications from the server. The server will send notifications only to its clients.

Lab Work

In this lab, you can use the software you developed for the past two labs. The focus should be on the server that will have an event notification service with it.
As a scenario, you can consider that there can be several hardware errors that can occur and that can have an impact on the server running on top. You may consider several events such as disk error, or networking error, or system overloaded that can be notified to the client.
The client logic may ignore these notifications, or take decisions according to their seriousness, up to cancelling its in-process request.

Tasks to Do

1. **New interface**. A service client should be able to subscribe/unsubscribe to notifications from the server. When subscribing, the client needs to send in its call-back reference. What events will be notified to the client? How will the client react to these notifications?

2. **Design**. Outline the new protocols, on the server and on the client side, in pseudo-code. This will be your design.

3. **Programming**. Adapt the protocols above for one computer environment and then provide an implementation of the publisher and client-subscriber (in Java or Python).

4. **Testing**. Test your implementation by simulating the execution of the protocols. Provide screenshots of your simulation.

Return your results to Tasks 1-4 described above, including the pseudo-code, the **code** and **screenshots**, in a **pdf** file on moodle, by the deadline.

Add comments to your code. Place your name and student number at the top of each class file. Place comments at the method level; use one-line comments inside method bodies to describe more complex statements if you feel they are not obvious.

Questions

The following questions are to be filled in individually by each student and the pdf file retuned through Moodle before the deadline.

1. What changes did you operate on the server side and on the client side?

2. Provide the new interfaces.

3. Provide only the new code and its comments. Add execution screenshots.

4. Your additional comments on implementing this service.