Universität Zürich[UZH]

Institut für Informatik

Lecture **Distributed Systems**
Fall 2010

Prof. Abraham Bernstein, Ph.D.

ASSIGNMENT III - MESSAGING SYSTEMS

(10 POINTS)

Rules

- Assumed programming language is *Java*.
- Code that is handed in and does not compile will NOT be graded. So please make sure to test your implementation properly.
- Assignments have to be solved individually. It is ok and also desired to discuss problems with peers, whereas copying code is not. As a result, plagiarism will lead to 0 points for the particular assignment for both parties.
- The due date is a hard deadline. E-mails that arrive after this deadline will be discarded and therefore the contained solution not graded.

All the above rules are final and no matter for further discussions!

Due date: Dec. 15, 2010, 14:00 (CET)

The purpose of this assignment is to get familiar and gain practical experience with Message Queue Systems.

Your tasks are the following:

1. A simple chat system

- Server specification: From command line accept "L" key press - list all connected users (clients involved in chat).

- Client specification: Each interaction consists of 2 phases: 1) select user (type in name, Enter), 2) send message (type message, Enter).

a. Implement using TCP sockets a message queue system (both server and client) which will enable the described scenario. The server must be multithreaded and must handle the queue of messages coming from clients. (**4 Points**)

i. server specification: class name=uzh.ifi.ddis.ds3.mql.Server

ii. client specification: class name=uzh.ifi.ddis.ds3.mq1.Client

b. Create a document explaining the following: (**2 Points**)

i. system architecture

ii. system flow

iii. message format

2. The "Project Solutions" company has a central office and more local offices. It can handle problems regarding 3 domains: IT, economic and human resources (HR). To obtain the best client satisfaction all the client requests are published by the central office to the local offices. There are 2 local offices for IT, 1 for economic and 1 for HR. A project request consists of a project name, project description, a request date, maximum payment and the domain. Once they receive the project requests each local office will reply back to the central office with a message stating the time and costs needed to finish the proposed project. For simplicity each local office randomly generates a price and an end date (greater than the current date). The central office will process each offer and select the best proposal (based on the end date and cost). The local offices should receive the messages even if they are not connected to the topic(durable subscriptions).

a. Implement the system using the JMS technology in Java and the HornetMQ standalone server. (**3 Points**)

i. Use JMS publisher subscriber pattern to publish the requests from the central office to the local office

ii. Use point to point communication to process client requests and local office bidding

iii. Provide a class for testing purposes which allows you to create a central office, the local offices and several clients to prove your solution. Also please provide all the needed resources to automatically deploy and install the resources needed on the HornetQ (queues, topics).

b. Create a document describing your implementation (somebody who has not seen your code before should be able to understand it by reading your documentation). (**1 Point**)

Note:

Follow the naming conventions specified. If any external resource is used (i.e. Files, Ports, Ip addresses) they must be specify-able in command line as an option - document the option (do not hardcode it!)

Grading:

Grading will be based on:

a) the correctness of your code, i.e. does it solve the given task? and appropriate error handling.

b) readability/structure of your code (including appropriate comments).

c) clarity of your documentation, i.e. does it really describe what you implemented and how well can it be understood by somebody who has not written or read the code.

What to hand in and how:

- Create a zip file named <your_student_id>_<first name>_<last name>_A3.zip (e.g. 1234567_John_Doe_A3.zip).

- This zip file should contain the structure defined below:

- The structure of the zip file:

    - /part_1
        - /bin **(compiled code)**
        - /src **(the source code)**
        - doc_a3_1.pdf
    - /part_2
        - /bin
        - /src
        - doc_a3_2.pdf
    - README <= optional see below

The limit of the zip file should not exceed 500 KB. If it does please contact: Floarea Serban (*serban|at|ifi.uzh.ch*) *or* Cosmin Basca (*basca|at|ifi.uzh.ch*).

In case your code requires any special treatment to compile, you have to enclose a README describing the necessary steps.

Send this zip archive on time via email to:

Floarea Serban (*serban|at|ifi.uzh.ch*)

Cosmin Basca (*basca|at|ifi.uzh.ch*)

The email subject should start with ***[DS 2010]*** .

Helpful reading

Message queue

http://en.wikipedia.org/wiki/Message_queue

HornetQ

http://www.jboss.org/hornetq

http://www.jboss.org/hornetq/docs.html

http://hornetq.sourceforge.net/docs/hornetq-2.1.2.Final/user-manual/en/html/using-jms.html#d0e1186

http://hornetq.sourceforge.net/docs/hornetq-2.1.2.Final/user-manual/en/html/using-server.html#using-server.configuration

JMS

http://en.wikipedia.org/wiki/Java_Message_Service

http://java.sun.com/developer/technicalArticles/Ecommerce/jms/index.html