Universidad de León
Bachelor Degree on Computer Science and Engineering
Course on Distributed Systems

Homework no. 3

Published on: 18th - November- 2020
Submission date: 25th-May-2020
Submit via: Agora virtual campus homework submission
Formats:

- Only **pdf** format is accepted for the writeup; please, include your name and ID in the document.
- The source code should include your name embedded in the comments

--- Study Guide ---

1. At all times, have the textbooks by Kindberg *et al.* and Peterson & Davie at hand as well as the presentations, notes, practice scripts and solved exercises in `paloalto.unileon.es/ds`.

2. Make sure that you richly explain the answers that you provide. The explanations that *we* compose have the greatest value for *our* advancement.

3. Properly cite whatever references you consult in completing this homework submission.

4. Download the companion zip file from:

   `http://paloalto.unileon.es/ds/homework/hw3-2020.zip`

Exercises

1. **Compose a brief discussion on the differences between the Java** Object Model and the Java Distributed Object Model by applying core concepts about Distributed Systems.

2. **How is the C/S model relevant to the 3-tiered architecture discussed** in the lectures?

3. **What's the value of multithreading in C/S computing?**

4. **Does Java RMI support passing objects by value? What about** passing objects by reference?

5. **In the lecture presentation about CS Models**:

   http://paloalto.unileon.es/ds/Lec/DS-Models.pdf

   in slide no. 17 a statement about C/S computing reads: *"This model does not scale well due to the centralized nature of the server"*. Explain some solutions to the lack of scalability of the C/S model based on the concepts explained in the course about DS.

6. **In C/S computing sending data entails turning the data into a bit stream such that its receiver can correctly interpret it.** The formats and techniques that specify how the different data types can be sent over a socket connection are known as marshalling, in general; in particular, in Java it is named serialization. Applying Java serialization one can send all the native Java data types and objects. We have mentioned the concept of marshaling several times along the practices of CN and of DS.

   The simplest form of marshaling happens when sending integers over a socket. The POSIX-compatible native sockets library of Linux offers a number of calls that let one to marshal an integer before transmission and also to de-marshal it upon reception. The htons() turns a 16-bit word from the hardware-specific byte ordering to the network byte ordering known as Network Byte Order. The counterpart to **htons()** on the receiver side is **ntohs().** Skim the slides from the presentation on Marshaling; on slide 2, the **ntohs()** family of functions are explained:

   `http://paloalto.unileon.es/ds/Lec/DS-Marshal.pdf`

   In this exercise we aim to study and experiment with marshaling in its simplest form, to that end, a TCP server is listening for connections on paloalto.unileon.es at port 60002; the server program performs a very limited number of operations ("Send the date"; "Shutdown server" and "Multiply integer by 2"). For the purposes of this exercise, we are interested only in the latter operation ("Multiply integer by 2"), consequently we provide a simple client program that exercises it.

The program clientBase.c sends the operation string to the server; the 4-byte integer that is to be multiplied by the server comes next to the string. The binary representation of the integer is translated to Network Byte Order before it is sent over the socket connection. When the integer (N) is received by the server, it computes 2*N and marshalizes it appropriately, so what the client receives is the Network Byte Order of 2*N.

a. Run Wireshark or tcpdump for identifying and highlighting the operation string and the ensuing 4-byte integer in Network Byte Order.

b. Identify and highlight the response string provided by the server along with the ensuing 4-byte response (2*N) in Network Byte Order.

c. Write the client code that prints the result of 2*N on the client's screen.