University of Cyprus

Computer Science Department

EPL425: Internet Technologies

Fall 2017

Announced Date: Wednesday**,** 27/9/2017
Submission Date: Κυριακή, 8/10/2017 (23:59)

Implementation of a simple Client / Server Program

The goal of this exercise is to write a simple **Java Client / Server program** through which you will obtain hands-on experience in fundamental communication and performance concepts. Specifically, you are to implement a basic information exchange functionality between two remote hosts communicating over the Internet. You are to assume the following scenario:

- A server program waits for any number of clients to connect to it.
- A client program simulates N concurrent users that generate a series of requests (workload) towards the server.
- For each simulated user, the client program establishes a connection to the server. Each request is a simple *HELLO* (handshake) message. The message comprises of:
    1. a "HELLO" string,
    2. the client IP address & port, and
    3. the simulated user-id [1...N].
- Once the server receives the *HELLO* message, it immediately generates a *RESPONSE* message and sends it back to the client. The message comprises of:
    1. a "WELCOME <user-id>" string and
    2. a variable (random) size payload.

The payload size is calculated by considering the current system time on the server, when generating the *RESPONSE* message. The payload size should be in the range of: (300 – 2000) Kb.

The client program should simulate the transmission of at least 300 requests by each user. When the above has been met, the client should gracefully terminate the connection with the server.

Requirements:

1. The server program should run on a virtualized Linux server instance that you will configure and deploy over the AWS EC2 infrastructure.
2. Both client / server programs should simulate concurrency, hence you are required to utilize Thread programming in your Java code.
3. The client program should be executed on your PC / laptop / UCY CS lab machine and execute with at least 10 concurrent simulated users.
4. You are required to test the performance of your server and the inter-communication of hosts by measuring the following metrics:

    i. **Communication Latency**: The round-trip time (RTT), capturing the time between sending a request and receiving the respective response. Latency is measured in units of time (seconds, milli-seconds, nano-seconds, etc.) at the client-side.

ii. **Server Throughput**: The amount of requests a server satisfies in a given time interval (i.e. every second). Throughput is measured at the server-side.

5. Your performance tests must consider the following:

   i. Use of repeated client-runs to obtain more accurate results.
   ii. Utilization of AWS instance types that allow you to run the server program over 1,2 and 4 vCPU's. Take care to use only Free Tier instance types.

6. You are required to generate plots of the above performance measures. These should include:

   i. Latency vs Number of Users (for 1, 2, 4 vCPU)
   ii. Throughput vs Number of Users (for 1, 2, 4 vCPU)
   iii. Throughput vs Average CPU load
   iv. Throughput vs Average Memory utilization

Notes:

1. The clients must know the server's address (IP and port) when they start. Thus the client program must take a command line parameter the server's address (e.g. *java Client 192.168.1.10 9999*)
2. The server will have as an address the IP of the machine it is running on and the port will be given to it as a command line argument (e.g. *java Server 9999*)
3. The clients will use a random port of the machine they reside on. Attention must be paid in the fact that some port may be taken by other programs or clients.
4. The number of repetitions should be given as a parameter to the server through the command line (thus the actual command for starting the server would be: *java Server 9999 1000)*

General Instructions:

1. This is a team assignment. Use the teams assigned during the recent labs. The teams must consist of two persons and can work with the pair programming technique. Effort of individual team members will be identified and evaluated accordingly, influencing the overall team grade. Therefore, make sure you contribute and promote equal team effort.
2. You are required to utilize GitHub for the sharing and revision control of your source code. Use the respective repository assigned to your team, i.e., "*wn16_tmXX*" where XX the ID of your team).
3. Create a folder in your repository named: **Assignment1.** Then create and commit the following sub-folders
   1. **src** : that will contain your Java code
   2. **docs** : that will contain any documentation file
   3. **plots** : that will contain the performance plots with latency and throughput
4. All commits should be accompanied with meaningful messages
5. You are free (urged) to use Branches and Issues of GitHub.
6. Once your code is developed and runs as expected, commit and tag it as "**final**". Send an email to dpasch01@cs.ucy.ac.cy with the link to you Github repository by the 2nd of October 23:59. Any submissions and commits after that will not be graded.
7. Avoid plagiarism. Your programs will be checked by plagiarism detection software.

   Plagiarists will be given the mark of 0 and risk further disciplinary actions.