Contents

1. Requirements

Suppose that the clients have intelligent home appliances that can be controlled remotely using remote procedure call (RPC). Each such device can communicate with the server that will compute the time when the device will be started for an optimal energy consumption.

Develop a client-side application (either a desktop application or a web application based on a JavaScript framework running from the browser) for the smart appliance associated to a client that:

i) gets the client hourly historical energy consumption over d days in the past ($E_{client}^{d}(h)$);

ii) gets the averaged energy consumption for the client over the past week (e.g. client baseline);

$$Baseline(h) = \frac{1}{7}\sum_{d=1}^{7} E_{client}^{d}(h)\,,\forall h \in \{1..24\},$$

$$where\ E_{client}^{d}(h)\ is\ the\ client\ energy\ consumption\ for\ day\ d\ in\ the\ past\ and\ hour\ h\ from\ day\ d$$

iii) allows the selection of a program with a duration in hours (select a duration D of a program);

iv) gets the best time to be started considering the baseline and the program duration to avoid energy peaks from the client (e.g. to minimize the maximum energy consumption for every hour of the day)

$$Compute\ t_s, t_e\ such\ that\ Min\big(Max(Baseline(h) + E_{Device})\big), \forall h \in [t_s, t_e], t_e = t_s + D$$

1.1. Functional requirements:

- The client application displays a chart with the client historical energy consumption over d days in the past (default d = 7)
- The client application displays the client baseline as a reference consumption for the next day
- The client application allows the selection of a program with a duration (either from a list of programs or by entering directly the duration)

- The client application asks the server for the best start time in the next day to minimize the peaks of energy consumption. It displays the new chart of estimated consumption as the baseline summed with the device max consumption.

1.2. Implementation technologies:

- Choose one of the following technologies: **any RPC framework** (for instance: gRPC, Apache Dubbo, JAVA RMI, .NET Remoting, JSON-RPC 2.0, etc.)

***NOTE: Before choosing a technology check if it is feasible to deploy on HEROKU cloud. For instance, HEROKU cloud does not support HTTP/2, thus it requires other libraries to perform a successful deploy. Other details are provided in the laboratory presentation.**

2. Deliverables

- A solution description document (about 4 pages, Times New Roman, 10pt, Single Spacing) containing:
  a) Conceptual architecture of the distributed system.
  b) UML Deployment diagram.
  c) Readme file containing build and execution considerations.
- Source files. The source files and the database dump will be uploaded on the personal GitLab account created at the Lab resources laboratory work, following the steps:
  - Create a repository on GitLab with the exact name: DS2021_Group_LastName_FirstName_Assignment_3
  - Push the source code and the documentation (push the code not an archive with the code or war files)
  - Share the repository with the user utcn_dsrl

3. Evaluation

3.1. Assignment Related Basic Questions:

During project evaluation and grading you will be asked details about the following topics:

- Distributed objects middleware components: Stub, Skeleton, Dispatcher, etc.
- gRPC, protobuf files
- HTTP/2 key differences to HTTP/1.X
- Distributed objects vs Local objects
- Distributed objects problems: security, latency, life-cycle, etc.

3.2. Grading

The assignment will be graded as follows:

Points	Requirements
5 p	**Minimum to pass** • Client – Server application RPC with one distributed object and. display client historical consumption on chart • Simple GUI (Desktop form or Web)

	• Documentation • Correct answers to 3.1 questions
2 p	Client baseline feature
1 p	Program duration selection feature
2 p	Compute start time feature and show chart with estimated consumption if device is run