Internet of Things
2021-2022

Final Project

May 13, 2022

After completing the "Hands on `CoAP`" tutorial on `Java` and Californium framework, you are able to build both `CoAP` servers and clients, all running on a single PC, as well as running on different machines, addressable through IP addresses on a common IPv4/IPv6 network. The final project is the following and is related to the traffic management in a smart city crossed by a river.

1 Connected City

Consider a city in which people and vehicles moving inside its borders shall be able to exchange information with the urban mobility infrastructure. The city is on the shores of an important river, crossed by multiple bridges since ancient times.

In detail, each bridge $B_i, i \in \{1, \ldots, N_{\text{BRIDGE}}\}$ is monitored by a CoAP-like process P_i that handles the traffic crossing B_i through two traffic lights at the two bridge sides, denoted as $L_{i,1}$ and $L_{i,2}$. At every instant, each bridge allows one-way traffic: the direction of the traffic is controlled by $L_{i,1}$ and $L_{i,2}$. Assuming that a **green** traffic light is associated with "`1`" and a **red** traffic light is associated with "`0`," at each time $\{L_{i,1} = 1, L_{i,2} = 0\}$ or $\{L_{i,1} = 0, L_{i,2} = 1\}$. Each process P_i, every T seconds, "swaps" the status of the traffic lights at the borders, so that when one traffic light allows vehicles to cross the bridge in one direction, the other one prevents vehicles at the other extreme of the bridge from crossing the bridge. So as, these traffic lights should listen for the decisions taken by P_i and react consequently (e.g., showing a message in the console, etc.).

Each bridge B_i is then equipped with a vehicles counter C_i, which reports the total amount of vehicles that have crossed the bridge so far.

Finally, these data (traffic lights status and counters) may be of interest for citizens, who can request the status of the traffic lights for a specific bridge in advance (in order to take decisions on the best route to be followed to go from one side of the city to their destination), and for the local police, who may be interested in knowing (i) how many vehicles crossed all the bridges in the city and (ii) if there was any vehicle which crossed a bridge with a red traffic light—this is possible since each vehicle, once reached a bridge, should send their own identifier V_{ID} before requiring for the bridge's status.

2 Repository Entity

In the considered scenario, in case there is the need to share some information among the different entities mentioned in the previous section or as will be described in the following, it is possible to adopt the following strategy. Create a "fake" repository (logically similar to a database), represented by a `Java` class, in which it is possible to maintain some static objects, as follows:

```
public class Repository {
    private static Repository instance = null;
    public static Repository instance() {
        if (instance == null) {
            instance = new Repository();
        }
        return instance;
    }

    public YourObjectClass var = new YourObjectClass();
}
```

In this way, the required information is available everywhere, and one can refer to this object with `Repository.instance().var.<method>()`.

Note #1: the `Repository` class has to be used ONLY if it is strictly necessary; otherwise, it is recommended to exchange data ONLY exploiting the `CoAP` protocol.

Note #2: the content of the `Repository` class will exist only at run time, i.e., it is not physically stored on disk. Please, be aware of this aspect.

In general, it is suggested to follow the Plain Old Java Object (POJO) paradigm: keep internal variables related to the information proper of your classes as private variables; implement `get/set` methods for each of these private variables.

Finally, it is up to you to define the exchange *schema* to be adopted for representing the information to be sent through `CoAP POST` requests. As mentioned before, one possibility is represented by the adoption of the `JSON` format.