NUCLEARCOAP

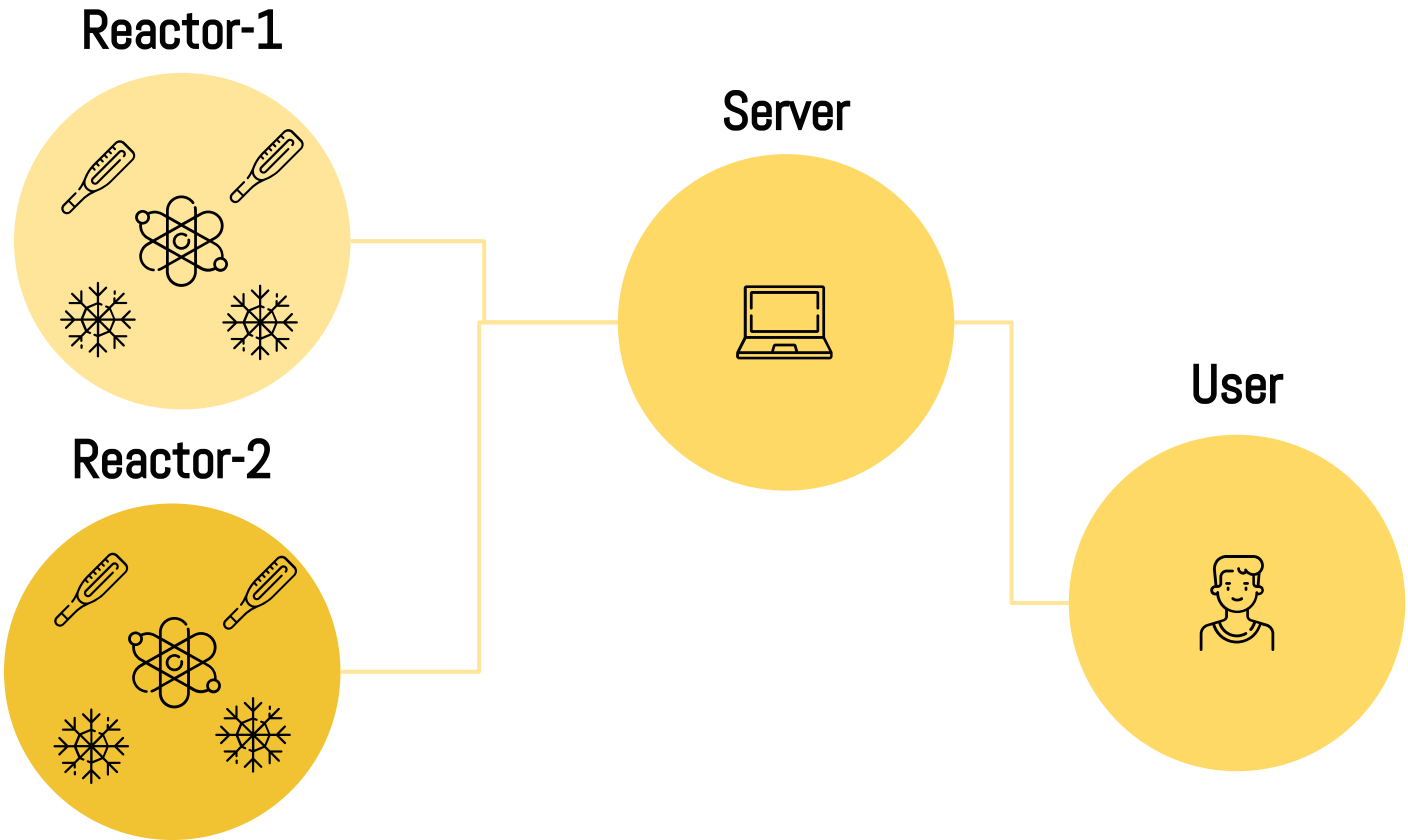
IoT Project

Luca Barsellotti

SCENARIO

- Each **reactor** requires an **internal temperature** under a specific **threshold**.
- Several **sensors** sense the temperature inside each reactor with a predefined period.
- When the average temperature is above the threshold, several **actuators** are activated to cool the reactor.

ARCHITECTURE



COAP RESOURCES

- The server exposes two CoAP **resources**:
  - /register
  - /control
- "**/register**" is used by sensors and actuators to announce their presence.
- "**/control**" is used by users to send commands to the server.
- The sensors expose an observing resource called "**/temperature**" used to retrieve the sensed temperature.
- The actuators expose a resource called "**/cooling**" used to cool down the reactor.

SERVER RESOURCES

REGISTRATION

GET: used by the **actuators** to announce their presence and the reactors to which they belong (passed as payload). They are added to an HashMap structure.
POST: used by the **sensors** to announce their presence and the reactors to which they belong (passed as payload). After the exchange, the server starts observing the registered sensor to store its **measurements**.

CONTROL

GET: used by the **users** to retrieve the lists of reactors and last measurements.
PUT: used by the **users** to activate the actuators in a specified reactor.

DEVICE RESOURCES

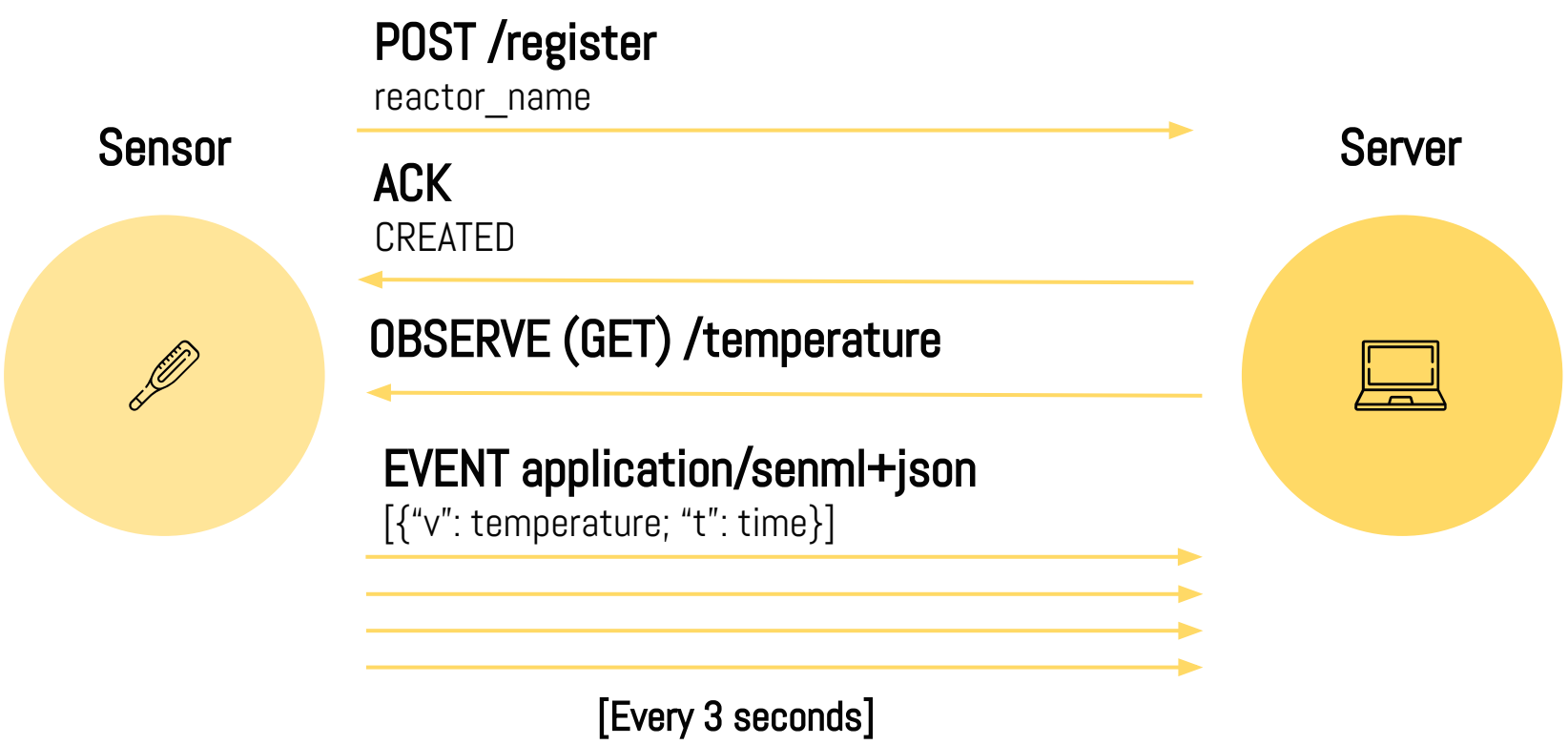
TEMPERATURE (SENSOR)

GET: called by the **observing event handler** to sense the temperature and to send it to the **server**, after the registration.

COOLING (ACTUATOR)

PUT: used by the **server** to activate the actuator **cooling mechanism** when the average temperature in the corresponding reactor is above a certain threshold (850) or when the users require it.

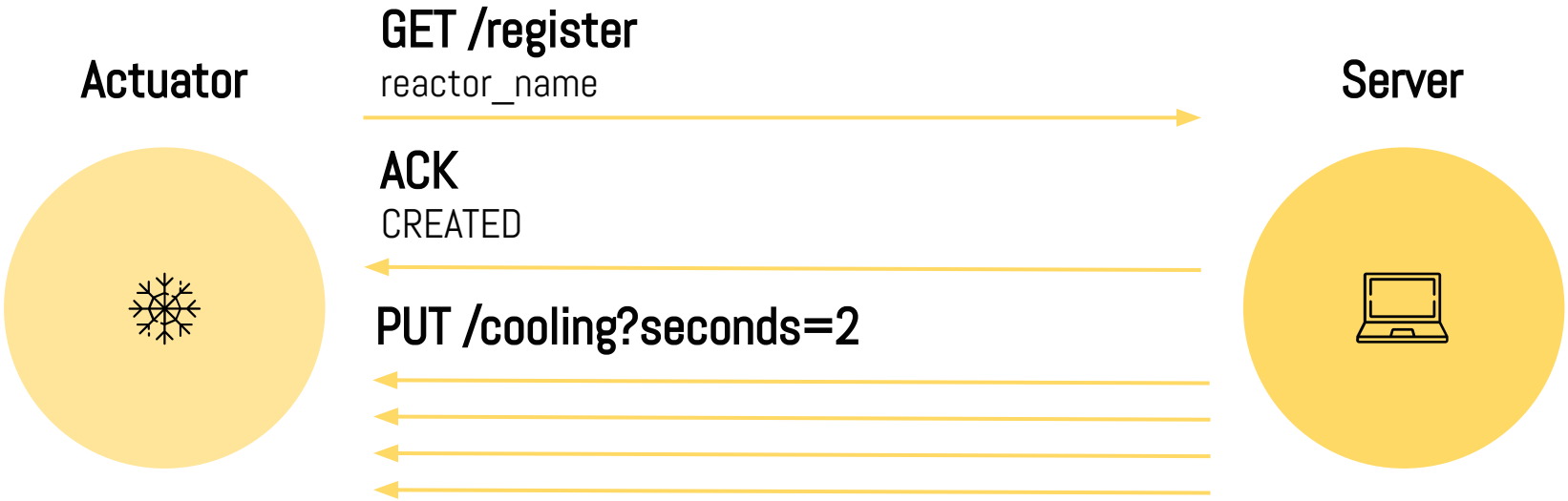
SENSOR-SERVER INTERACTION



MONGODB

- The sensed values of the temperature are stored in a database called **NuclearCoAPdb** on **MongoDB**.
- Each **reactor** has its own collection.
- Each **measurement** is a document stored in the collection relative to its reactor when the observation is received by the server from the sensor.
- The fields are “**date**” and “**value**”.

ACTUATOR-SERVER INTERACTION



[The server checks the average temperature in a reactor each second. If it is above 850 then activates the actuators]

USER COMMANDS

There are 4 commands that the user can send to the server.

- **GET {"o": 1}**
  Show the list of available reactors in the database.
- **GET {"o": 2, "r": reactor, "l": limit}**
  Show the list of the last *"limit"* measurements in the indicated reactor.
- **GET {"o": 3, "r": reactor, "l": limit}**
  Show the average of the last *"limit"* measurements in the indicated reactor.
- **PUT {"s": seconds, "r": reactor}**
  Activates the cooling in the actuators that belong to the indicated reactor. That means that the server sends a PUT request to the actuators.

COOLING TIMER

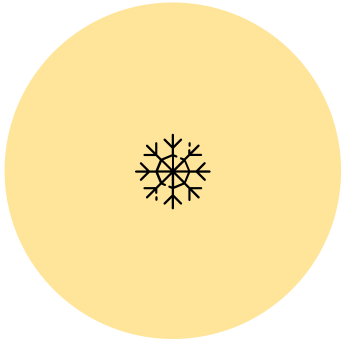
PUT /cooling?seconds=20

[After 10 seconds]

PUT /cooling?seconds=15

[After 3 seconds]

PUT /cooling?seconds=10



THANK YOU!