ECE 528- Application Software Design

Alan Palayil

Due Date: March 30th, 2023

Question 1:

Consider the RESTful interface design of a service that allows clients to query power consumptions of plugs during certain time period.

i. To specify the path in @GetMapping and the parameters with annotations:
   - @GetMapping("/api/power/{plugName}")
   - synchronized public Object GetPlugPower(@PathVariable("plugName") String plugName)

ii. To add support for the "begin" query parameter while still supporting the conditions from (i).
   - @GetMapping("/api/power/{plugName}")
   - synchronized public Object GetPlugPower(@PathVariable("plugName") String plugName, @RequestsParam(value="begin",required=false) String begin)

iii. To add support for the "end" query parameter while still supporting the conditions from (i) and (ii).
   - @GetMapping("/api/power/{plugName}")
   - synchronized public Object GetPlugPower(@PathVariable("plugName") String plugName, @RequestsParam(value="begin",required=false) String begin, @RequestParam(value="end", required=false) String end))

iv. An example of a response body in JSON format for the above RESTful requests.
   - The power key should be mapped to a list containing either changes in the plug's power or the plug's power measured at predetermined time increments over the specified time interval.
   - Example:{ "plugName": "examplePlug", "power": {"2022-03-05T23:34:55": 1500.5, "2022-03-06T00:30:00": 1800.25, "2022-03-06T01:15:30": 1200.75}}

Question 2:

When designing a system where plug states come from an MQTT broker and clients can also control the plugs via RESTful services, you have two main options regarding updating the plug state when a RESTful request is received:

i. Update the state of the plug immediately upon receiving the RESTful request.
   - Pros:
     - Faster response times for clients, as they receive feedback about the state change immediately.

        - Simplified logic in the service, since it doesn't have to wait for confirmation from the MQTT broker.
    - Cons:
        - There might be a discrepancy between the reported plug state and the actual plug state if the MQTT broker fails to process the request or takes longer than expected to process it.
        - Potential for race conditions between RESTful requests and MQTT messages, where the system might have an outdated state when processing a new request.

ii. Update the state of the plug after receiving confirmation from the MQTT broker:
    - Pros:
        - More accurate state representation, as the service only updates the state after receiving confirmation from the MQTT broker.
        - Reduced risk of race conditions, as the state is updated in response to MQTT messages, ensuring the system is in sync with the actual state of the plug.
    - Cons:
        - Slower response times for clients, as they need to wait for confirmation from the MQTT broker before receiving feedback about the state change.
        - Increased complexity in the service, since it must handle confirmation messages from the MQTT broker and potentially handle errors or timeouts.

In conclusion, deciding whether to update a plug's state immediately upon receiving a RESTful request depends on the system's priorities. Faster response times and simplicity favor Option (i), while accuracy and reducing race conditions favor option. Often, Option (ii) is preferred due to its consistency and reliability, despite the potential increase in complexity.