Distributed Systems (521290S)   
Course Project Report

<Team Tesla>

|  |  |  |
| --- | --- | --- |
| Group Member | Student ID | Email |
| Md Mobusshar Islam  Muhammad Ahmed  Muhammad Talha Arshad | 2305578  2304796  2304797 | [mislam23@student.oulu.fi](mailto:mislam23@student.oulu.fi)  [mahmed23@student.oulu.fi](mailto:mahmed23@student.oulu.fi)  [Muhammad.arshad@student.oulu.fi](mailto:Muhammad.arshad@student.oulu.fi) |
|  |  |  |
|  |  |  |

# *Instructions:*

***1. DEADLINE (31.Jan.2024): Registration of topic, i.e. the Section “Course project overview” described.***

***2. DEADLINE (12.Feb.2024): Submission of project design. Includes all the red parts marked ”Design:” in the beginning of each section below. At this point, you don’t need to provide full detailed solution(s) description.***

***3. DEADLINE (24.Feb.2024): Final submission of this report, including full detailed description of the project.***

*After each deadline, you can modify your design, implementation and this report based on feedback.*

*Hints:*

* *Remember take a look at the course project slideset and video recording in Moodle.*
* *The way of writing is free (e.g. text / bullet points / figures), only English please.*
* *A picture is worth thousand words.. recommended length* ***3-4 pages*** *(including the title page).*
* *Instead of MS Word, you can use another format (e.g.* ***latex****, RTF), but follow exactly the same document structure.*
* *Submission must be a* ***PDF*** *format*
* *Regarding sections 1-8 below, if some sections are not applicable to your project (e.g. you did not consider fault tolerance) just write “Not applicable, because <short one sentence explanation>”.*
* *In your initial idea registration, already think of architecture, organization and communication pattern in a general level. Just saying its client-server is not enough.*
* *Course book has lots of protocols and solutions described, so saves your workload if you take a look. A bonus is that you can find code or a library for the solutions (e.g. P2P / AMQP / MQTT in python).*

*To save space, for the final submission remove the above instructions and replace with overview section.* ***Submit your final report as PDF.***

# Course project overview

A brief description (abstract) of the project, including 1) the application and 2) the software and system architecture, and 3) the distributed systems topic that your project addresses.

**Overview**

The project aims to process the request for IoT devices based on certain factors like preference, urgency, etc, and respond to the requests accordingly.

The following are the components of the system.

* **Smart IoT Devices:**

Mocking the IoT devices with the code script that will serve as different nodes of the distributed system.

Each IoT device will be monitoring the environment, and it will be requesting the server based on different conditions like temperature, humidity etc

* **Central Server:**

Manages communication with all connected devices. Queues request from devices and prioritize them based on need. Respond to device requests with appropriate actions. Stores historical data for analysis and future optimizations.

# **Request Queuing and Prioritization:**

# Implement a queuing system to manage incoming requests. Assign priority levels to requests based on factors like soil moisture levels, weather conditions, and plant types. High-priority requests get processed first.

**Distributed System Topics Covered**

* **Message Queuing System**:
* Round Robin Queuing:

Distributes incoming requests evenly among available server resources in a circular manner.

* Priority Queue:

Prioritizes requests based on urgency or importance, ensuring that high-priority requests are processed first.

* **Load Balancing:**
* Round Robin Load Balancing:

Distributes incoming network traffic or requests evenly across multiple servers.

* Weighted Round Robin:

Assigns different weights to servers based on their capacities, allowing for more efficient resource utilization.

* **Priority Queue Management:**
* Heap-based Priority Queue:

Utilizes a heap data structure to efficiently manage and extract the highest-priority requests.

* Weighted Priority Queue:

Assigns different weights to requests based on their criticality, influencing the order in which they are processed.

* **Resource Management:**
* Token Bucket Algorithm:

Controls the rate at which requests are processed to prevent resource exhaustion.

* Leaky Bucket Algorithm:

Smooth outbursts of incoming requests, preventing sudden spikes in resource usage.

Fault Tolerance and Redundancy:

In the above-mentioned basic topics covered, we will be trying different approaches to find out best possible solutions for a specific problem

**Architecture Diagram**

A diagram of a diagram

Description automatically generated with medium confidence

# 1. Architecture

Design: Describe the software architecture of your application and the distributed system.

1. Drawing (preferred, but text description also ok) of the system architecture, including:

* Outline of the software architecture: architectural style and organization
* Roles of the system components in your application: client / server / (super)peer / dispatcher / broker / etc
* Services that are either provided by servers or used by the clients / peers / etc
* Interactions / logical connections between components: e.g. directed arrows with short text explaining what is the purpose of each interaction
* Middleware or external services (if applicable)
* Data: e.g. file / database / external data source and its contents

2. Description of how your project demonstrates the distributed systems topic you selected?

3. How does your design support / implement the evaluation you planned for the project? Its important think about the evaluation (see section below), since it can affect your design a lot!

Advice: Use the terminology from the course book, the design may not be accepted if the used terms are not clear. The level of detail is *subsection title*. For example, just saying “its P2P” is not accepted, but is it *structured / unstructured / hierarchical*?

# 2. Implementation

Describe how you implemented the functionality for each system component (e.g. client and server), including software solutions and possible hardware.

Advice: For the design, is it already beneficial to think about the implementation of the components. You save a lot of iterations in development, if you, at this point take the design seriously. For example: what are the components’ internal architectures, how components are virtualized, what software is needed to realize functionalities, etc?

# 3. Communication

Design: Describe how the components in your distributed system interact to implement the application, and possible communicate with system-level services.

It pays to describe the following:

* Interaction pattern, e.g. publish / subscribe
* Application-level protocol (if applicable), i.e. the messages / events you use in your application
* Interfaces, e.g. RPC or REST API
* Communication protocol (stack) you used in the project

For example, you are using HTTP atop TCP/IP and the RESTful interface is the following / With MQTT, these events are published..

# 4. Naming

Design: Describe briefly how the components in your distributed system identify and/or discover other components and resources. For example, is your system using flat / structured /attribute-based naming? What kind address + name scheme you implement? How do peers know their neighbors? etc..

# 5. Coordination

Design: If your project implements a synchronization / coordination / election / etc scheme or uses a such protocol, describe briefly your solution here. Refer back to section 1 and describe the components and their roles and the utilized algorithm (i.e. election / gossip / etc) in more detail. Again, a picture is worth thousand words, see examples in the course book.

# 6. Consistency and replication

Design: If your project follows a data- or client-centric consistency model or implements replica management / protocol or such a scheme, describe it here briefly. Refer back to section 1 and describe the components and their roles in more detail. Again, a picture is worth thousand words, see examples in the course book.

# 7. Fault tolerance

Design: If your project implements a fault detection or tolerance mechanism, solution for reliable communication or distributed commit protocol describe it here briefly. Refer back to section 1 and describe the components and their roles in more detail. Again, a picture is worth thousand words, see examples in the course book.

# 8. Security

Design: If your project implements security, authentication, authorization mechanism, describe it here briefly. Refer back to section 1 and describe the components and their roles in more detail. Again, a picture is worth thousand words, see examples in the course book.

# Evaluation

A table with numeric data speaks thousand words. Include a (very) brief analysis of the evaluation results.. what can be seen, is something missing, etc.

Advice: Already, for the design, is it necessary to think of the evaluation you planned for your project. This way you will have a “placeholders” built-in into your system. enabling easy evaluation once you are at this stage. Think of the test cases and what and how data is collected in your system, do you use a logging tool, etc. General rule is “everything is related to everything”..

# Workload distribution

Plan of sharing the workload (wl )and estimated hours. Please fill in the real calculated workload only on the final submission of the report.

|  |  |  |  |
| --- | --- | --- | --- |
| Student name | Tasks | Estimated wl. (h) | Real wl. (h) |
|  |  |  |  |
|  |  |  |  |

# References

In case you utilized existing work or software in your project, please list the sources here.

Design: You should already have some ideas which (existing) software you are using in your project..