CM2307: Object Orientation, Algorithms and Data Structures

DESIGN DOCUMENT

Lakshmi Ksheeraja Sikha | 23112887 | 8th January 2025

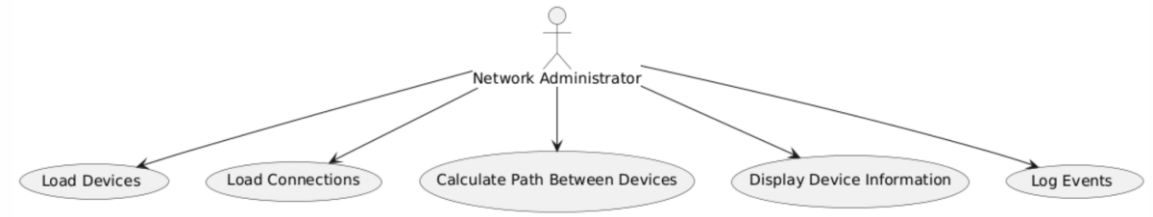
Introduction

This project's purpose was to design and implement a Network Management System (NMS) that manages network devices and routes between them. The NMS is a software system used to monitor, configure, and manage the network in real-time. It consists of key components such as Network Device Management, Route Management, Monitoring and Diagnostics, Logging and Events Management, and User Interface.

This document presents a comprehensive overview of the system's design through various UML diagrams, including use case diagrams, class diagrams, and sequence diagrams. Each diagram serves a specific purpose, illustrating the interactions between users and the system, the structural relationships among classes, and the dynamic behavior of the system during key operations. By applying principles from SOLID, GRASP, and APIE, as well as relevant design patterns from the Gang of Four, the design ensures that the NMS can easily accommodate new device types, calculate paths from one device to all others, and implement weighted connections in the future.

I've decided to make three diagrams to describe the assignment's code design: through UML diagram, Class diagram and a sequence diagram. The following sections will delve into the details of each UML diagram, explaining their significance and how they contribute to a well-designed, user-centric network management solution. Through this structured approach, the NMS project is positioned to meet current needs while remaining adaptable to future challenges in network management.

UML Diagram



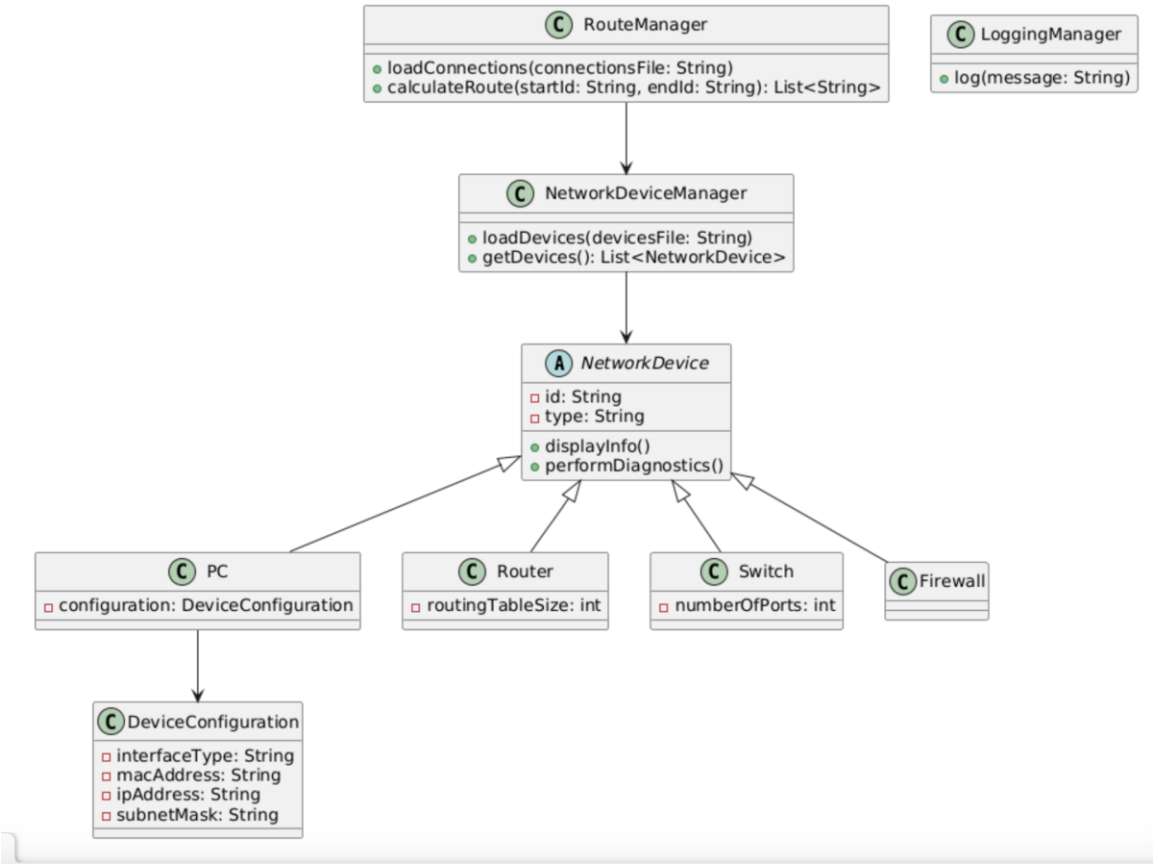
The use case diagram illustrates the interactions between the user (Network Administrator) and the system. It includes the following use cases:

- Load Devices
- Load Connections
- Calculate Path Between Devices
- Display Device Information
- Log Events

Reasoning:

- **User -Centric Design**: The use case diagram focuses on the user's perspective, showing what functionalities the system provides. This aligns with the User - Centered Design principle, ensuring that the system meets user needs.
- **SOLID Principles**: The use case diagram supports the Single Responsibility Principle by clearly defining distinct functionalities that the system must provide, making it easier to manage and extend.
- **Extensibility**: New use cases can be added in the future without affecting existing functionalities, allowing for easy adaptation to changing requirements.

Class Diagram



The class diagram represents the structure of the system, showing the classes, their attributes, methods, and relationships. Key classes include:

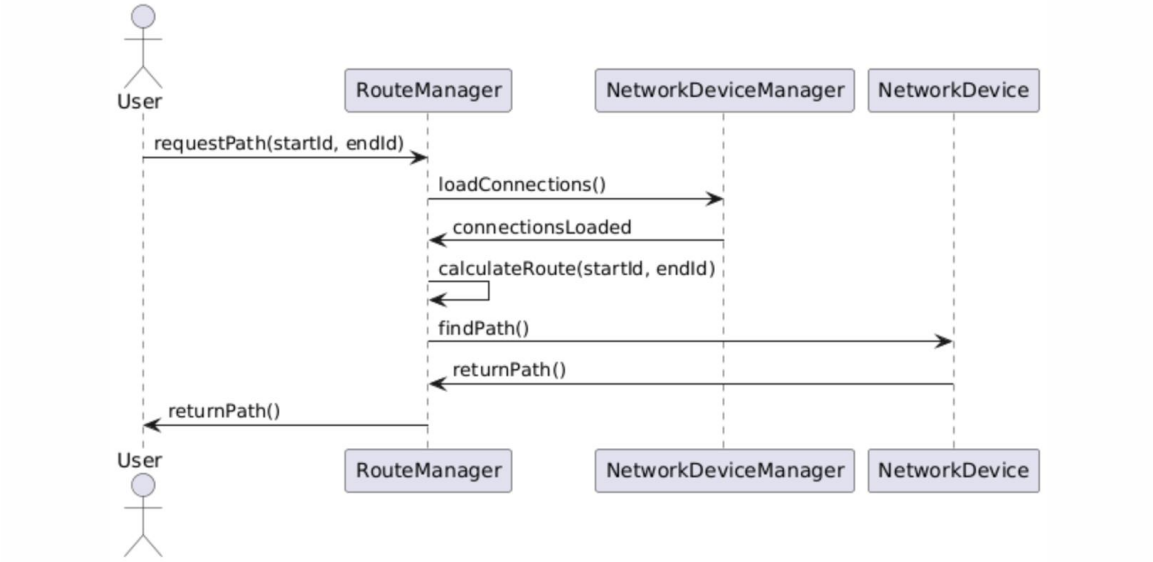
- NetworkDevice
- PC, Router, Switch, Firewall
- DeviceConfiguration
- NetworkDeviceManager

- RouteManager
- LoggingManager

Reasoning:

- **Object-Oriented Principles**: The class diagram applies the Open/Closed Principle by allowing new device types to be added through inheritance without modifying existing code. For example, adding a new device type like Server would only require creating a new class that extends NetworkDevice.
- **GRASP Principles**: The Information Expert principle is evident as each class is responsible for managing its own data. For instance, NetworkDevice knows its configuration, and NetworkDeviceManager handles device management.
- **Design Patterns**: In NetworkDeviceManager, instances of different device types are created based on input data. This promotes loose coupling and enhances maintainability.
- **Reliability and Extensibility**: The clear structure of the class diagram allows for easy addition of new classes and methods, ensuring that the system can grow without major redesigns.

Sequence Diagram



The sequence diagram illustrates the flow of messages between objects during the process of calculating the path between two devices. It includes the following interactions:

1. User requests a path.
2. RouteManager calls loadConnections() on NetworkDeviceManager.
3. RouteManager calls calculateRoute().

4. NetworkDevice instances are accessed to find the path.
5. The result is returned to the user.

Reasoning:

Dynamic Behavior: The sequence diagram captures the dynamic interactions between objects, showing how they collaborate to fulfill a user request. This aligns with the Behavioral Design aspect of object-oriented design.

SOLID Principles: The Dependency Inversion Principle is demonstrated as high-level modules (like RouteManager) depend on abstractions (like NetworkDeviceManager), rather than concrete implementations. This makes the system more flexible and easier to test.

Extensibility: If new functionalities are added, such as calculating paths to all devices, the sequence diagram can be easily modified to reflect these changes without altering the overall structure of the system.

Conclusion

The given diagrams above for the NMS assignment illustrate the systems components and their principles while adhering to the object-oriented principles. The use case diagram focuses on user needs, while the class diagram provides a clear structure for extensibility and reliability, and the sequence diagram captures the dynamic behavior of the system. By applying relevant design patterns and principles such as SOLID and GRASP, the design ensures that the system can adapt to future requirements, such as adding new device types or implementing weighted connections, without requiring major changes to the existing code or the design.