Introduction

About Rule conflict

Need for Rule conflict

Demo 1 – Rule Conflict using Interval tree

Implementation details

Database schema

Output

Shortcomings

Demo 2 – Rule Conflict implementing algorithm in referred paper

Modification

Classification

Input format

Database schema

Explanation of each conflict

Output

Conclusion

References

Appendix

Project structure

Dependent libraries

Introduction

Internet connected and interconnected devices, collectively referred to as Internet of Things (IoT) are becoming reliable means to automate daily activities for people and organizations. This interconnection among devices and web services requires a need for representing and managing interactions between them.

In traditional systems, policies are typically used to govern these interactions. However, most of these systems are static in nature when compared with IoT systems. In IoT, devices act with respect to context and how they have been configured. Thus, efficient tooling and framework are required for governing such heterogenous systems.

Rule Conflicts

A key way in which the programming of IoT systems can become unsafe is through conflicts which we refer it as Rule Conflict.

1. Conflicts can emerge when two or more instructions given to IoT devices cannot be satisfied simultaneously. A simple but practical example of this is when two instructions are provided to a single device simultaneously, both of which cannot be executed. For instance, a single light-bulb may have two simple rules provided to it – one that requires it to be turned on during evening hours, and other that requires it to be turned off when no one is in the room. Conflicting IoT programs can occur with a single user who perhaps does not realize instructions can conflict. Or through multiple users who encode opposing preferences.
2. Conflicts can arise between an app rule and a predefined policy. For example, turning on a light based on time can violate an energy-conserving policy that turns off light based on room occupancy. In these two cases, what is required are automated methods to highlight to users when such situations arise before they become a problem.

Programming Goal:

Key source of complexity for IoT applications include a significant amount of event-based (for example, context driven) logic that is well known to be error prone. This problem does not disappear even if the building blocks of programming model are simple as the logic that needs to be encoded does not change. The present implementation of Aquaponics does not have a conflict checking mechanism for detecting rule conflicts and providing feedback to the user. The challenging part is to accurately detect conflicts and provide feedback to the users when the size and complexity of IoT systems increase. The Overview of rule detection mechanism is shown below.