An Application of Automata Cloud Processing

Project Report 1

By: Jason Walker

Advisors
Official: Dr. George Markowsky
Unofficial: Zach Hutchinson Ph.D student

A capstone project proposal for partial fulfillment for the degree of
Bachelors of Science in Computer Science
University of Maine Orono
October 2016

Introduction

The Internet of Things is a big buzzword right now. There are a lot of concerns on the implementation and how things are going to play out in the future. Currently there are a few major key players in the arena however there is no agreed upon standard for IoT. It is estimated that there will be 20.8 billion things connected by 2020. (Gartner) This is an incredible number of connected devices and poses huge question on how all these things will be implemented let alone the security aspect of it all.

This project focuses on the 'how' and not so much on the security part. The security aspect of the internet things is a legitimate concern and should not be taken lightly however my focus is of an application for automata processing within a implemented framework. Some of the questions I will answer are (to be refined):

- Is there a place for automata in smart connected devices?
- Can complex deterministic finite state automata machines be described and manipulated using a variety of components?
- How can a user control such complex machines from afar?
- Can these deterministic FSA machines be dependent on other machines?

Automata graphs can be used to describe transition of devices. These devices can be as simple as a light bulb with two states of as complicated as a traffic light with more than two states. The question is whether we can use these descriptions in an implementation to control the states (and transitions) of those machine. For example a DFSA graph can be described using the following adjacency matrix:

```
{
        1: {2:1},
        2: {3:1, 4:1},
        3: {4:1},
        4: {1:1},
}
```

The matrix has a series of states (1-4) and its corresponding edges, each edge has a weight of 1. If we take this matrix in JSON format and we read this in we can reconstruct a graph depicting a DFSA machine. Through a variety of methods including; utilizing a RESTful interface, a MQTT message broker, and a special Automata Processor class. We can manipulate the automaton into transitioning to different states. We can then build a Queue that would utilize the MQTT message broker to communicate with the hardware being used to create the light. In *Figure 1* you can see that the basic structure of the architecture.

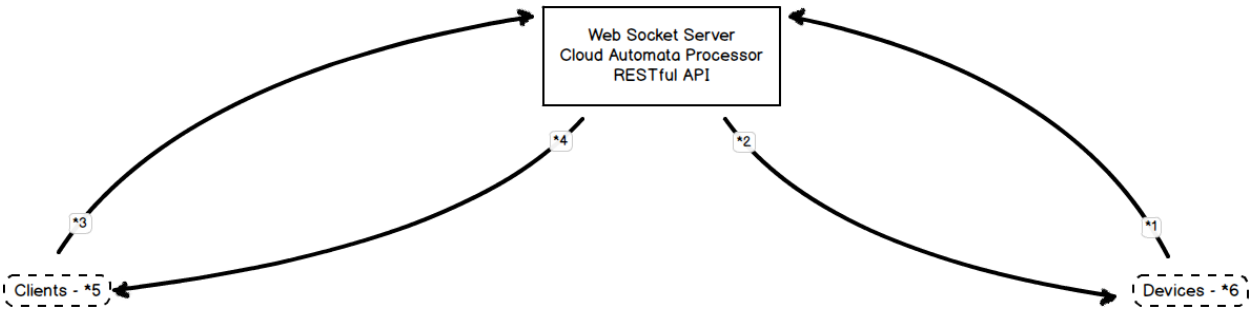


Figure 1 - Basic Architecture Diagram

The diagram above describes the basic architecture as I currently have it planned. The client (*5) can be a desktop web application or a mobile device. This is what triggers the desired state. When you tap a button on the GUI that is meant to turn the device on or off it sends a request (*3) to the servers. The servers are integrated and connected to the database. The database holds the automata graphs and the current state of the machine. The automata processor will then run the algorithm to find the actions needed and store them in a queue. The MQTT message broker (*2) will send the commands to the devices (*6) to handle one by one until the queue is empty.

In the most basic sense the system will take in automata depicted using an JSON document and a desired state. The system will then output a series of actions or transitions needed to be in that state. At the core of my project will be the automata processor. This system will be integrated with a RESTful service and MQTT message broker that will allow quick communication between the client and the server. An integral part of the project will revolve around the DFSA transition dependencies and how we can use those to create complex machines involving other DFSA machines. Think of this as grouping devices together to control them (i.e. turning three lights on in a room).

Project Definition

For this project I will mainly focus on just the implementation of the overarching system. Meaning anything that corresponds to getting hardware to transition to different states using the GUI. Ideally in this project a user will hit an action on the GUI that will trigger an action to get the device to transition (a light to turn on/off). The action will trigger the automata processor to figure out the transitions it needs to execute to get the device/s into their proper state.

Components

The framework contains a few pieces of critical components. Some of the main components are:

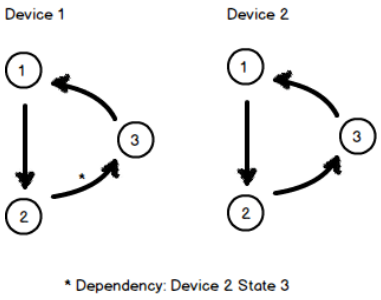
- Web Application and/or Smart Phone Application
- RESTful Interface
- MQTT Message Broker
- Automata Graph Processor with Dependency Handling
- Mongo Database

- Device Library

The web application or smart phone application will consist of an interface that lists all their devices and their current states. They will be able to tap/click on a button that will trigger a state change for the device. For instance they want to turn their lights on. The button will send a request to the server indicating the device id and the desired state. The server will receive that request and send it to the automata processing system. The AP system will then traverse the corresponding adjacency matrix that is stored in the Mongo database. The AP system will have a queue of transition on the device/s that need to be taken to get the device into the desired state. The shortest path to the state will be calculated using Dijkstra's algorithm. Once we have a Queue of transitions this will then be sent to the MQTT message broker. The MQTT message broker works be subscribing each device to a channel. Using this method, the MQTT message broker can send each transition to it's proper device. Essentially what will happen here is the MQTT server will tell the device what function it needs to execute. The MQTT server will continue until the Queue of transitions is empty. At this point the device/s should be in the requested state and should then be sent back to update the GUI and database with the new state. The device library will really be a utility library to work with the server.

Dependency Handling

Example:



This diagram represents two automata graphs. The current state of Device 1 is state 1. The current state of Device 2 is state 1 also. Let the transition between each state be fn12, fn23 and fn31. The transition on Device 1 has a dependency that Device 2 need to be in state 3. When we run our algorithm to find the needed actions to take we will end up with: `[D1-fn12, D2-fn12, D2-fn23, D1-23]`. This queue represents the transitions needed to be taken by the devices to be in the desired state.

Libraries & Methods

The server side application will be written in Node.js. I have a variety of dependencies the Node application relies on to work properly. Some of the more important ones include; a implementation of Dijkstra's algorithm, Mosca MQTT Node library, Express.js HTTP engine, and Mongoose for use with MongoDB. The AutomataProcessor class will be the major focus of this project. The class will be mainly a series of available functions. Some of the feature that are to be included in this class are:

- Fetching of the Automaton from the DB
- Computing the transition Queue
- Computing the dependencies in the Queue
- Validating circular dependencies
- Validating DFSA graphs
- Returning the Queue to execution

Related Work (NEEDS WORK)

There are a few companies that are in on this field to date. Amazon has a platform that is extremely similar to mine however I am not working on this project with theirs in mind. In fact I honestly haven't taken a deep look into how they implemented there. They both have common feature such as the MQTT message broker and the RESTful interface. That is really where the commonalities stop. Another company that has a similar system is Particle.io. They have a very nice interface and platform for building devices.

References (NEEDS WORK)

http://www.gartner.com/newsroom/id/3165317
https://www.particle.io/