EECS6432 - Adaptive Software Systems (Fall 2018-2019)
Daniel Marchena Parreira - 216181497

Part 1

Feedback systems from different application areas

Smart Power Socket was an IoT project focused on developing a cheap device to collect electric current and tension data from various electrical outlets. The project was divided in 2 main systems, the first being the actual IoT device that would use electrical current sensors and tension detection sensors to collect environmental data. In addition to that, a RESTful API system was also developed in order to store the data collected from the devices, and also control their behavior.

For instance, whenever the electric current of a certain device reached a determined threshold, the RESTful API would keep collecting data, but it would also send a message to the device. Having said that, both the speakers in the IoT device and a push notification would be triggered, that way the user was always aware of any changes.

Finally, we could characterize this feedback loop as being Positive, since a continuous load of data is being delivered, but the feedback loop also informed the user of an abrupt case scenario.

Treasure Box Braille (TBB) is a DIY project maintained by the PiET Lab that has a special focus on developing a device for visually impaired kids to learn braille. In addition to that, the project proposes that the kid should be able to develop its own device following a simple manual.

The TBB contains a series of apps, but one of them called SCALP actually makes use of a simple feedback loop structure. On that note, SCALP is a simple Java application that starts with the OS bootup process. Its responsibility is to constantly check for new pendrive stick connections on the USB port (non-stop). Whenever a pendrive is connected to the machine, the app (in a new thread) scans the pendrive and loads any available audio files to the IoT device memory stick. In case a USB stick is disconnected, the app sends an audio message to the user.

Based on that description, we could say the type of feedback loop in use is Positive, since the application is constantly checking the USB port no matter what, and it also communicates to the user whenever an abrupt environmental change (disconnecting the USB device) happens.

Arduino Line Follower is an IoT project that aims on developing and assembling a basic line following robot. This machine is powered by many components, but we will focus on the 2 DC Geared motors, the 6 TCRT500L IR Emitter/collector pairs and the 2 wheels. On that note, the feedback loop is quite simple.

The loop starts with a continuous input of data coming from the IR Emitters, this flow of data then gets passed to a function that is responsible for processing it. The algorithm in this function first converts the data to a more normalized format to facilitate the analyses, and right after that it calculates any necessary adjustments it needs to make in order for the robot to keep following the right coordinates. As soon as the function returns those coordinate values, the device regulates the motor units in order for the robot to keep following the line.

This feedback loop could be categorized as Negative, since the loop is constantly using data from the sensor in order to calibrate where the robot should go.

Part 2

An Ultra-Large-Scale Systems (ULS) is a group of software, hardware and non technological systems that are architect to be decentralized, that way working as unique components. As a result, what makes ULS systems unique is not only the fact that they are composed by a group of systems that interfaces with each other, but also because they are developed by an astounding number of different people with different opinions and objective (stakeholders, users, etc..).

An equally significant aspect of ULS Systems is how they break the idea of technology-centric design, brining companies, and people from many areas to contribute together. On that note, it presents the concept of decentralized control where many systems and people are responsible for monitoring and controlling the grid, different from a centralized perspective where everything is managed by a single unit.

It is important to note however, that ULS Systems face many challenges. A really noticeable problem with those systems is the lack of documentation and frameworks on how to design, develop, maintain and expand them. On that note, ULS Systems are being developed simply as systems of systems of systems, and that is not sufficient to describe the complete complexity of those. Another problem to be tackled is the lack of understanding of the socio-technical ecosystem.

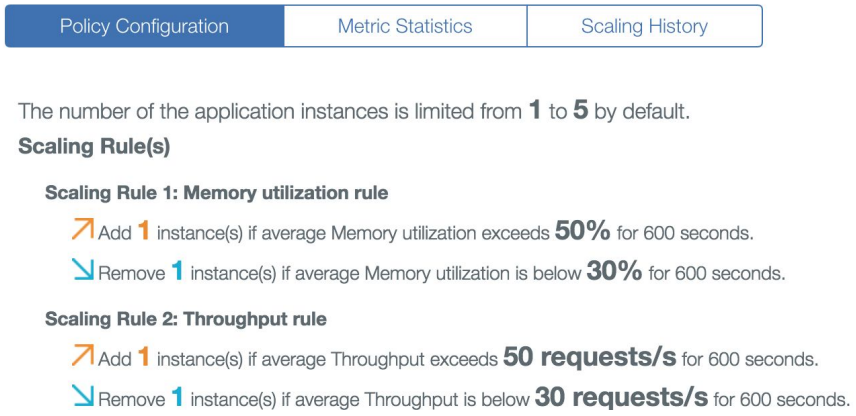
Reference:

Northrop, L., Feiler, P., Gabriel, R., Goodenough, J., Linger, R., Longstaff, T., Kazman, R., Klein, M., Schmidt, D., Sullivan, K., Wallnau, K.: Ultra-Large-Scale Systems. The Software Challenge of the Future. Technical Report, Software Engineering Institute, Carnegie Mellon University, 134 pages ISBN 0-9786956-0-7 (2006)

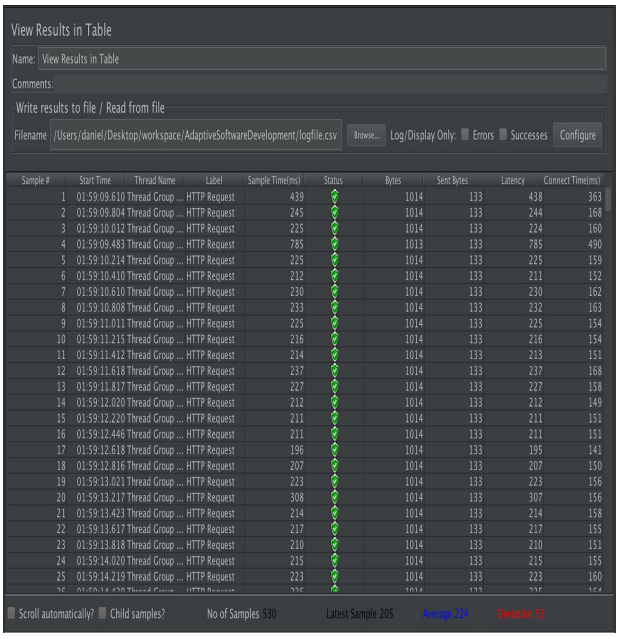
Part 3

- Auto scaling architecture was done using IBMs auto scaling solution. You can learn more about it [here]

- The policy rules used were:



- Workload graph and data:



- Discuss the limitations:

The application we deployed (Java Liberty App) is extremely simple. It is composed of a Tomcat web server that is currently serving a lean HTML page on port 80. Because of that, whenever we run our tests against (https://sampleapp-eecs6432.mybluemix.net/) the only true overload occurs whenever too many people try to request the main page. Those many connection attempts will generate various Threads in the server (one per user), since every new Thread needs a chunk of memory, eventually the instance will run of memory. In a reaction to that, the load balancer will spawn new instances of the application so all the users can connect to it.

- Video: https://youtu.be/7z8opbPqY9U