**Abstract**

FoG computing is not a new idea, but it is an unexplored and mostly untested theorem that has a lot of potential to supercharge the capabilities of IoT devices. FoG architectures could potentially be an extremely valuable asset to latency sensitive applications. The goal of this project was to demonstrate a simple use case of an IoT-Fog-Cloud architecture in which we also aim to display some of the advantages low latency through a small, scaled down version of a smart-city bigger picture.

**Introduction**

The solution envisioned was to represent FoG architecture at a smaller scale. The problem arises for time dependent applications when they must contact a cloud server that could be across the country or possibly not even in the same international territory. To demonstrate such an idea, we have chosen to replicate a simple traffic camera detection system. The system’s role is to monitor live traffic and determine if a light change should be prioritized based on waiting cards at an intersection. In this case, the camera system represents the IoT device. An IoT device such as this one will not have high processing power or large memory capacities. Therefore, it will require some offloading of tasks and space onto a more powerful system. Traditionally this would be accomplished by a cloud server with the ability to receive, process, and deliver a response. While this solution is clearly a step up from trying to force the work onto the IoT device, distance can certainly be an issue for certain cases such as this one. Time is of the essence here. If it takes longer to process that a car is at the light than the default change time of 45 seconds, for example, then this sped up process becomes meaningless. Our solution is this example it to have a local server setup to act as a FoG server. It will still process or store data like the cloud server, will contact and send data to the cloud server, and the cloud server can take upon its duties if it runs into failure or maintenance.

**Architecture: IoT**

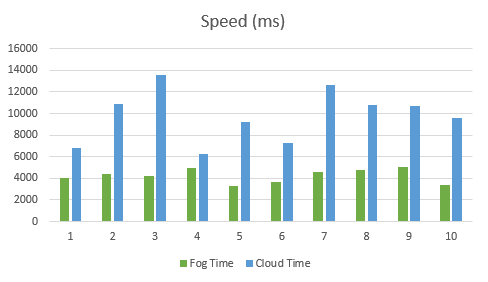
The IoT device in this case is the traffic camera. We have chosen to represent this by a laptop that will pull data from a live stream video of the Toomers corner intersection and send this data to the corresponding FoG or Cloud server to be processed, analyzed, and stored. The laptop itself is not the best representation of an IoT device but it serves its purpose correctly by not attempting to process any of the image recognition itself and not storing any statistical data.

**Architecture: FoG**

The FoG server for this project is running on an i7 processor with 3.10 GHz and 16GB of RAM. The FoG and Cloud server have the same application deployed to them so that they can communicate with each other but also process the requested task independently of each other. The FoG machine has a ping from the IoT devices that averages at around 8-10ms.

**Architecture: Cloud**

The cloud server is being represented by a free US West Heroku instance. The processor specifications were not given besides that it is a single process and that it has 512MB of RAM available. The cloud server experienced an average ping of 93-95ms. As mentioned, the cloud server has the full capabilities of the FoG server while also having the ability to interact with it.

Figure 1: FoG response vs Cloud response

The cloud server also plays a role of gathering statistical data from the FoG server, or gathering it independently if it is the one processing the entire task itself. Ideally, the FoG server will never be in charge of keeping statistics but rather focusing on keeping the latency low and delivering a prompt, immediate response to the IoT device.

**Software**

There were several technologies we used across the stack to assist in testing this use case. First, we used a NodeJS server with Google’s Puppeteer to simulate a headless browser to load the live stream of Toomers corner and capture the live and latest frame. Next, we submitted that image via POST request to a Flask webserver. This Flask application saved the image temporarily and then used Tensorflow Image Recognition to process and analyze the image and discover if any cars were present. After doing so, a response was sent back to the Node server for further actions and responses to the results.

**Experimentation & Results**

The laptop representing the IoT camera of this scenario was tested 10 different times for both IoT to Fog and IoT to Cloud. As mentioned previously, the pings for the FoG and Cloud differed greatly. Just by looking at those numbers at first glance, you would expect the FoG to be the idea choice. That expectation matched closely to the results received in this experiment. After examining Figure 1 it is obvious that the FoG server can return a response in half or even one third of the time on average that the Cloud server can. This is a desired result for a time-sensitive application such as monitoring live traffic data.

**Conclusion**

The idea of a IoT-FoG-Cloud architecture is beginning to become the topic of discussion for overcoming barriers that are being faced at the IoT level due to their heterogeneity and mobility. While we may not have discovered and ground-breaking material in this experiment, it is re-assuring to at least visualize the significance and future capabilities of FoG oriented designs and architectures. There is, of course, plenty of room for improvements and discussions on the topic. An evident and obvious issue we encountered is that we had to use a free cloud server, all of which don’t offer extreme or overbearing processing capabilities. However, we do not believe this skewed our results too heavily. We do wish that we had the ability to reverse the specs of the FoG and Cloud setups we had. It would be more ideal to have the lower spec architecture at the FoG level rather than the cloud level considering the cloud will be storing statistics and data long-term. Regardless, this procedure is just the very tip of the iceberg in terms of all the research required into this developing architecture and we hope to be able to improve it beyond just testing the bare bones of the idea next time.