Fog Computing Prototype

Submitted By:
Hitesh Sharma

Miguel Barbero

31-01-2021

—

Fog Computing Prototype Assignment

—

Submitted To:

Prof. Dr. D. Bermbach

J. Hasenburg

Preface

This project prototypes a Fog Computing scenario. We have emulated two sensors on local machine. Data from these sensors is sent to an edge node, which calculates an average and forwards it to the Cloud device, which performs some logic with this data and returns an answer. Reliability of messages in ensured.[1]

GITHUB: https://github.com/mmbarbero/fog-computing-TUBerlin/blob/main/README.md

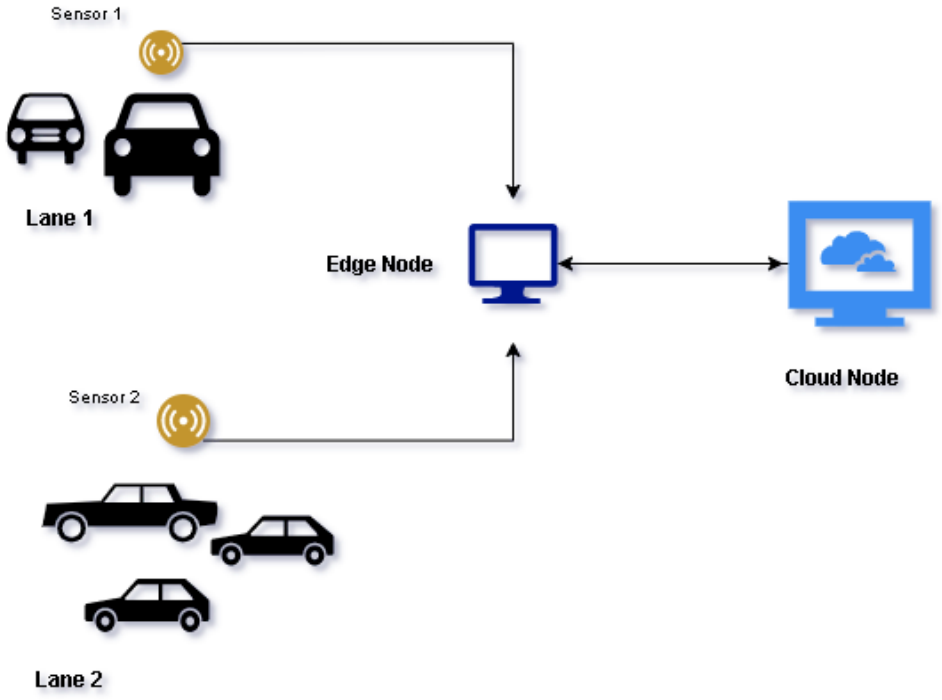
Description

Purpose

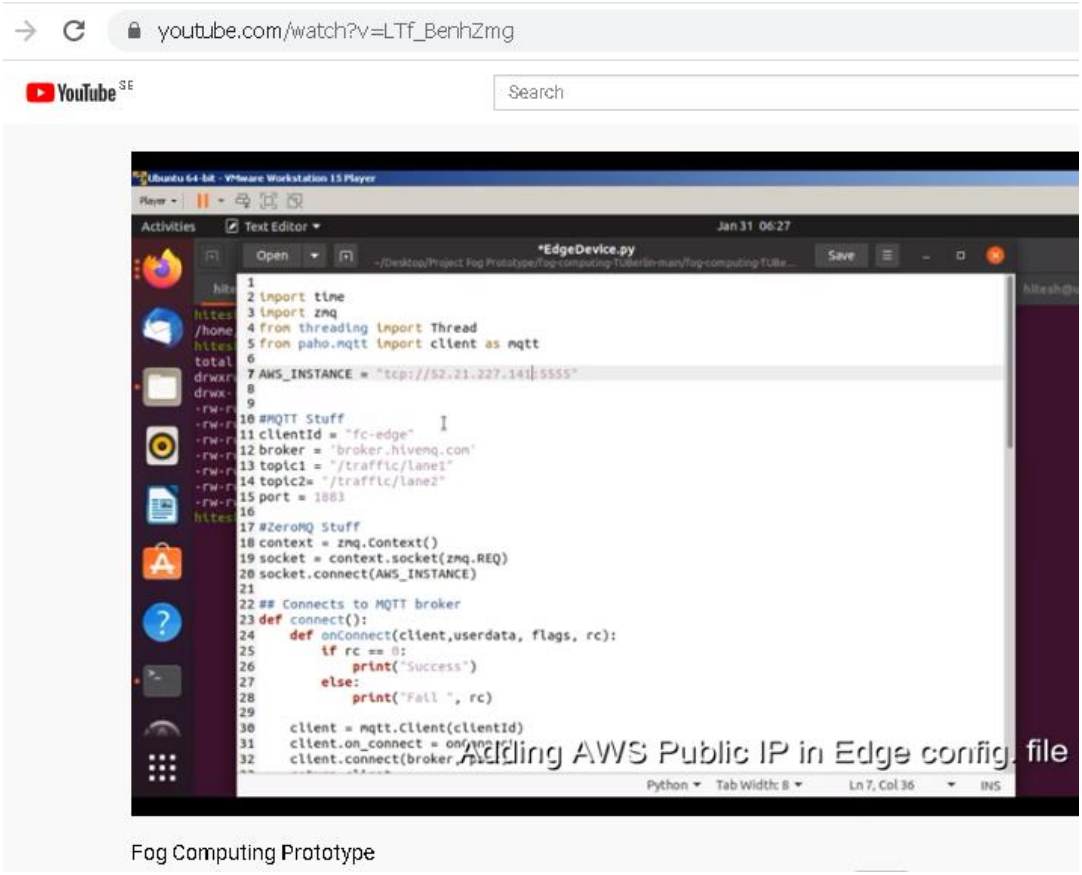
The virtual sensors have been designed for a scenario where we have two perpendicular roads, with a sensor counting the cars that pass in each road. They are being used to determine the duration of the traffic lights on each of the road, optimizing them for maximum flow of cars. They sense the number of vehicles passing through the road every ten seconds and send the result to the edge device. There could be more sensors on additional roads or traffic directions. The edge device integrates all the incoming data from each sensor to send it to Cloud node for computation purposes. This helps the cloud node to determine the amount of traffic congestion in the areas and send back the response with some logic computation on the cloud node.

Technical Implementation

We use two different methods of data transfer between all the different nodes. In the first place, we use MQTT (Message Queuing Telemetry Transport, ISO/IEC 20922), a lightweight, publish-subscribe network protocol to transport the sensor data to the edge node. It uses a broker located on the cloud, but this broker could also be implemented locally for testing (Explained in GitHub readme). Each sensor (or lane) is considered a separate topic for ease of implementation. Edge device is subscribed to both sensors' topics on the broker. The sensors publish the number of cars passed every 10 seconds. A logic has been implemented on the edge device that calculates the average of the vehicles that pass in a certain span of time. Edge device is also connected to our AWS instance using ZeroMQ REQ/RES mode. It runs on a thread and continuously calculates an average and sends it to the cloud. Every 20 seconds it calculates the average of cars that passed in and forwards it to the AWS instance. It then waits for a response before sending more, thereby, ensuring reliability. If the AWS instance goes down, ZeroMQ ensures that the last message is delivered once it comes back up. At the same time, an average is still being calculated during the time the problem persists, and once it's back up, the AWS still receives data to perform its calculations. This ensures that no data is lost.



Architecture Diagram



Fog Computing Prototype

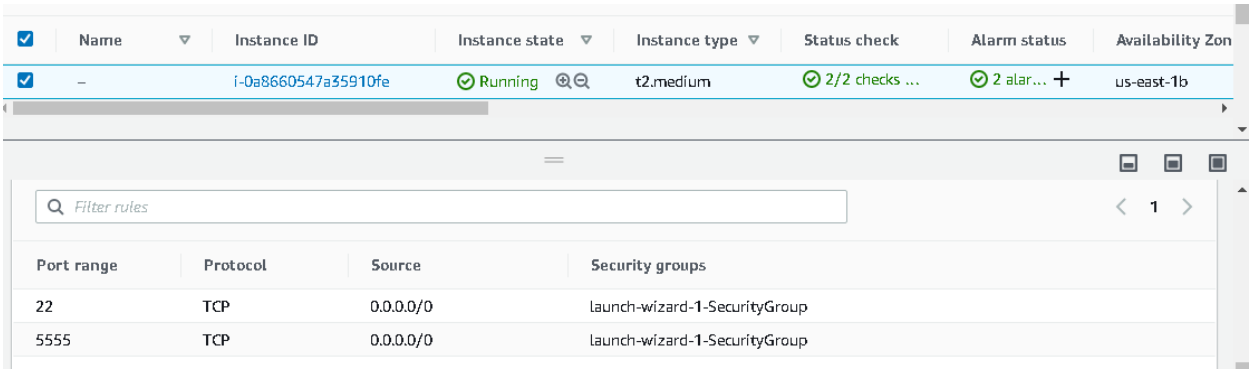
[YOUTUBE Prototype Demonstration Video]

```
hitesh@ubuntu:~/Desktop/Project Fog Prototype/fog-computing-TUBerlin-main$ python3 SensorLane1.py MED
Success
Send 7 to topic /traffic/lane1
Send 10 to topic /traffic/lane1
Send 7 to topic /traffic/lane1
Send 9 to topic /traffic/lane1
Send 6 to topic /traffic/lane1
Send 7 to topic /traffic/lane1
Send 9 to topic /traffic/lane1
Send 9 to topic /traffic/lane1
Send 10 to topic /traffic/lane1
Send 10 to topic /traffic/lane1
Send 8 to topic /traffic/lane1
Send 8 to topic /traffic/lane1
Send 8 to topic /traffic/lane1
```

Sensor 1: publishing data

```
hitesh@ubuntu:~/Desktop/Project Fog Prototype/fog-computing-TUBerlin-main$ python3 SensorLane2.py LOW
Success
Send 5 to topic /traffic/lane2
Send 2 to topic /traffic/lane2
Send 5 to topic /traffic/lane2
Send 0 to topic /traffic/lane2
Send 0 to topic /traffic/lane2
Send 1 to topic /traffic/lane2
Send 4 to topic /traffic/lane2
Send 5 to topic /traffic/lane2
Send 4 to topic /traffic/lane2
Send 3 to topic /traffic/lane2
Send 1 to topic /traffic/lane2
Send 2 to topic /traffic/lane2
Send 4 to topic /traffic/lane2
Send 2 to topic /traffic/lane2
Send 4 to topic /traffic/lane2
Send 1 to topic /traffic/lane2
Send 0 to topic /traffic/lane2
Send 1 to topic /traffic/lane2
Send 3 to topic /traffic/lane2
Send 4 to topic /traffic/lane2
Send 0 to topic /traffic/lane2
Send 2 to topic /traffic/lane2
```

Sensor 1: publishing data



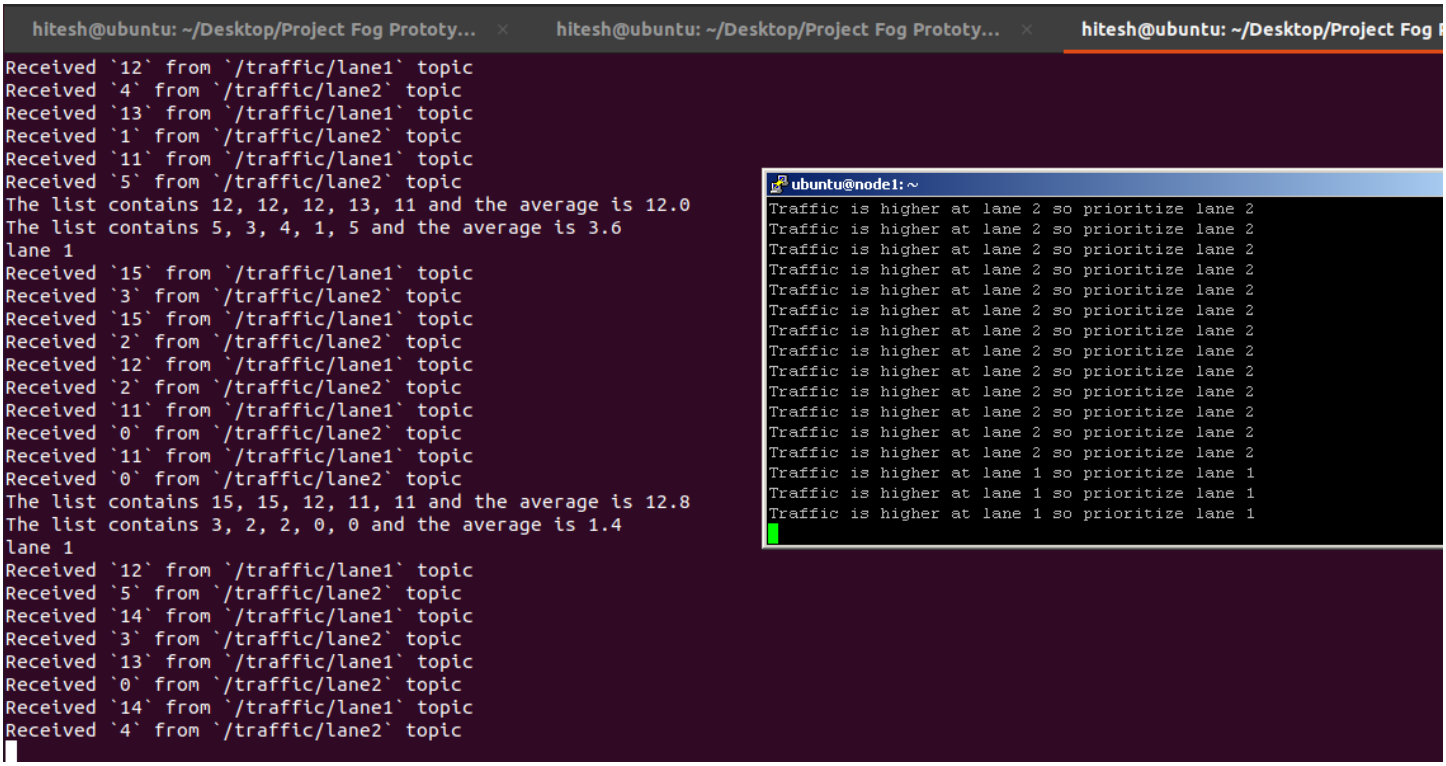
Cloud Instance Setup- AWS

```
hitesh@ubuntu:~/Desktop/Project Fog Prototype/fog-computing-TUBerlin-main$ python3 EdgeDevice.py
Success
Received `9` from `/traffic/lane1` topic
Received `0` from `/traffic/lane2` topic
Received `6` from `/traffic/lane1` topic
Received `4` from `/traffic/lane2` topic
Received `7` from `/traffic/lane1` topic
Received `1` from `/traffic/lane2` topic
Received `9` from `/traffic/lane1` topic
Received `2` from `/traffic/lane2` topic
Received `9` from `/traffic/lane1` topic
Received `1` from `/traffic/lane2` topic
The list contains 9, 6, 7, 9, 9 and the average is 8.0
The list contains 0, 4, 1, 2, 1 and the average is 1.6
Message Received
Received `10` from `/traffic/lane1` topic
Received `2` from `/traffic/lane2` topic
Received `10` from `/traffic/lane1` topic
Received `3` from `/traffic/lane2` topic
Received `8` from `/traffic/lane1` topic
Received `0` from `/traffic/lane2` topic
Received `8` from `/traffic/lane1` topic
Received `4` from `/traffic/lane2` topic
Received `8` from `/traffic/lane1` topic
Received `1` from `/traffic/lane2` topic
The list contains 10, 10, 8, 8, 8 and the average is 8.8
The list contains 2, 3, 0, 4, 1 and the average is 2.0
Received `7` from `/traffic/lane1` topic
Message Received
Received `3` from `/traffic/lane2` topic
```

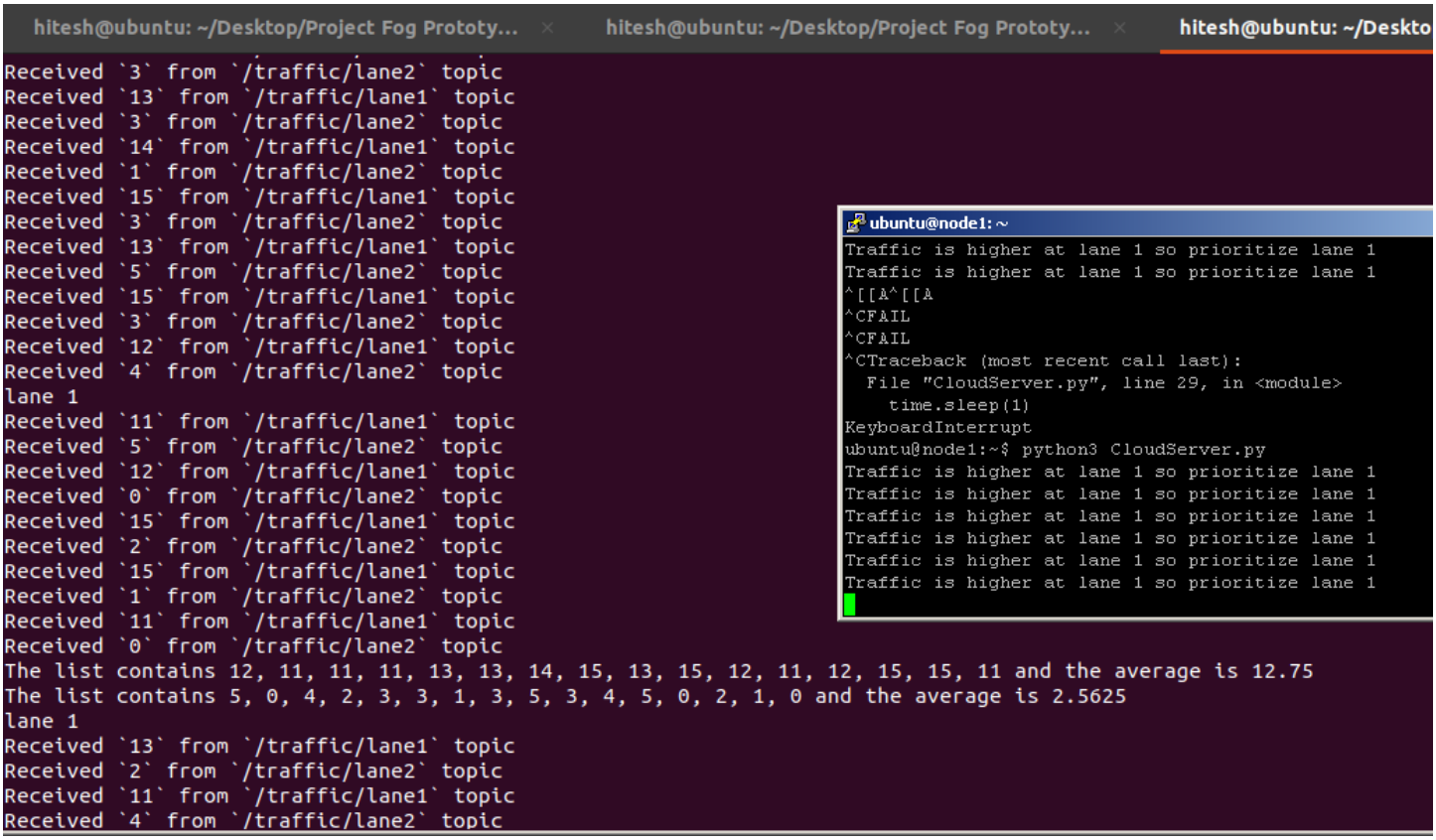
Edge Device

```
ubuntu@node1:~$ python3 CloudServer.py
Traffic is higher at lane 1 so prioritize lane 1
Traffic is higher at lane 1 so prioritize lane 1
Traffic is higher at lane 1 so prioritize lane 1
Traffic is higher at lane 2 so prioritize lane 2
Traffic is higher at lane 2 so prioritize lane 2
Traffic is higher at lane 2 so prioritize lane 2
```

Cloud Device



Edge and Cloud node (Side by Side View)



Ensuring reliability

(Intentionally failing cloud device, it queues up the incoming values from the sensors. It then computes & sends the result to cloud when cloud is up again.)

Citations:

1. https://zeromq.org/get-started/?language=python&library=pyzmq#
2. https://mqtt.org/software/