FC SS 2022 Prototyping Assignment

TsungJui Wu
Technical University of Berlin
Berlin, Germany
111012912jerry@gmail.com

Tim Dockenfuß
Technical University of Berlin
Berlin, Germany
dockenfuss@campus.tu-berlin.de

I. Introduction

Imagine you were admitted to the hospital and a very infectious lung disease was currently widespread throughout the country. Would you not want everyone in the hospital to wear a wear, and have the air quality checked every so often to ensure your optimal recovery?

That is why our idea for the Fog Computing Summer Term project was a combination of a face sensor combined with an air quality control sensor to check these two data sets for conformity.

II. Design

Our project includes number clients, which are all in charge of seperate things. These all communicate with one cloud system, which handles certain responsibilities.

A. Face Sensor/Camera

The job of the face sensor is to check whether or not a person standing in front of it is wearing a mask. For this, we trained a Tensorflow model with a small set of data to check if a person is wearing a mask. This then checks the camera feed, and uses its model to return a confidence score of its sureness that the person is or isn't wearing a mask. This score is then sent to the cloud. Also, to prevent overloading the cloud with excess data, we do not constantly exchange data with the cloud but rather send the data a time per 2 seconds. This time could be manually manipulated according to the users need.

B. Air Quality Sensor

Our air quality sensor detects the parts per million of carbon dioxide in the air of the hospital and sends it to the cloud. This is just a purely simulated sensor though, so in order to test our setup we have it increase by a random value between 10 and 20 every second while air conditioning is turned off, or decrease by 10 if it is on.

C. Door Opener

The door opener is a sensor that receives input from the cloud. From there, it gets the information of whether or not it should open the door and keep it open for three seconds in order to let a person pass through.

D. Air Conditioner

The air conditioner also gets information from the cloud and is responsible for turning the air conditioning on and off in order to improve the existing air quality. It receives this input from the cloud, which makes its decision based on data given by the air quality sensor.

E. Cloud

The cloud is responsible for different things depending on the sensor/client it is communicating with.

For the face sensor, it receives the confidence score and communicates the decision to open or close the door to the door opener.

The air quality sensor sends its data to the cloud, where the cloud then decides whether or not to turn the air conditioner on or off depending on the value it has received.

The cloud sends data to the door opener, which the opener then uses to keep a door open or shut.

Lastly, the air conditioner receives on/off signals from the cloud to help regulate the air quality and save electricity in the case that the air conditioner can be turned off.

III. Reliable Messaging

To ensure that message is reliably transfers between the end node and the fog server, we implement a TCP-like algorithm that replies to the sender an ACK once the message is received. When the client sends a request to the server, it also pushes the message into a queue. The queue is a list of key-value pairs, with a unique id as the key and the value being a compound of the timestamp it was sent, whether it was ACKed or not, and the message it sent. Before sending another message to the server, our client checks if the queue is empty. If not, it then checks for a timeout value to see if the message should be sent again. If the messages are to be resent, it shall be pushed back into the queue, but will not be considered again for resend. On the server side, once a message is received, before any calculation, it will resend an ACK to the client to ensure the connectivity. The client will then drop the messages in the queue that are ACKed.

IV. Limitations

While our sensor checks whether or not a person is wearing a mask going into the hospital, it can not guarantee that the person keeps it on while in it. For this, cameras would have to be placed at all public locations within the hospital.

To accurately determine and correctly identify all masks, our model would have to be trained more extensively to insure the safety of the hospital.

In order to be used in a real setting, an actual air quality sensor would have to be implemented.

The door opener opens the door if a person in front of the face sensor is wearing a mask, but it could be that that person has more people coming in with them, for which a mask has not been established.

Algorithm 1 Client side message reliability

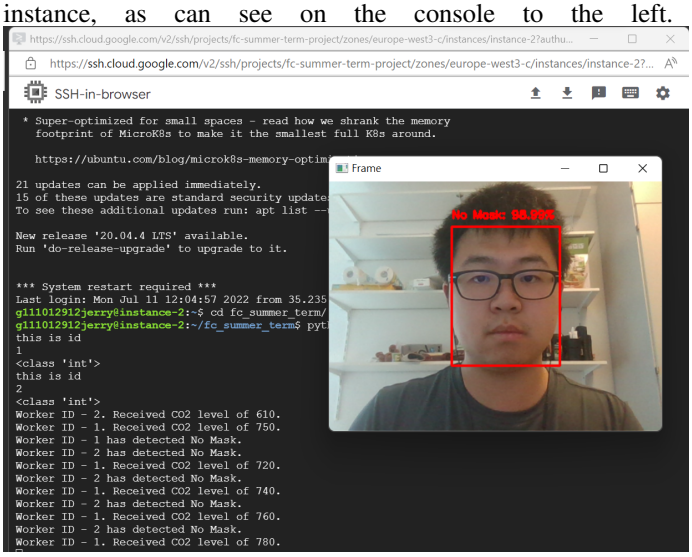
Ensure: Server is connected
  Define queue Q = (uid, ts, isACKed, msg)
  while True **do**
    for messages in Q **do**
      $q \leftarrow Q.pop()$
      if q not ACKed and ts+$timeout$ $>$ now **then**
        resend q
        push q into queue with new timestamp and uid

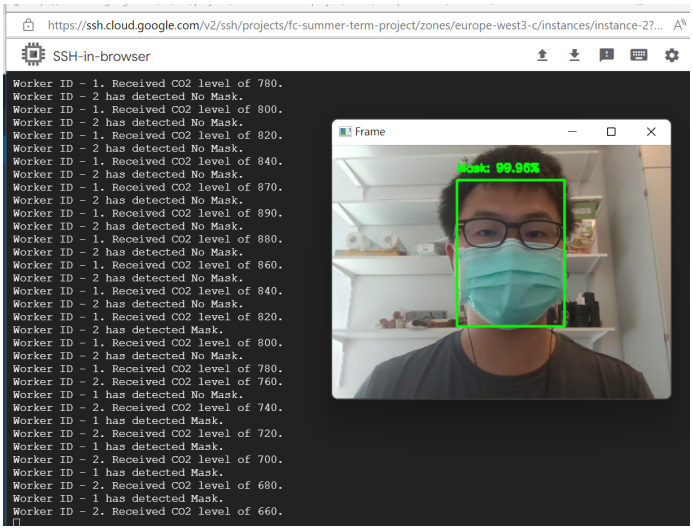
    send msg to server

    if Received ACK from server **then**
      pop message from Q

V. Screenshot Section

Below are screenshots of the face mask detector sending data to our google cloud instance, as can see on the console to the left.





For the message reliability, we implement our own algorithm to check if there are any messages that need to be resent.

```
if self.msginQueue():
  for msg in self.queue:
    try:
      if msg["timestamp"]+timedelta(0,self.timeout) >= datetime.now() \
        and not msg["isAckd"]:
        self.send(socket, f'{msg["uid"]};{msg["body"]}')
    except:
      break
```

References

[1] Kasun, Zeromq Client-Server example using Python. *Github*. (2011), https://github.com/kasun/zeromq-client-server

[2] Chandrika Deb, Face-Mask-Detection based on computer vision. *Github*. (2021), https://github.com/chandrikadeb7/Face-Mask-Detection