Documentation

BENZENE BOTS

07.07.2021

1 Team members

- Abdullah Zafar
- Mustafa Touqir
- Syed Muhammad Saim
- Dawar Zaman

2 Introduction

Truck Platooning is basically linking two or more trucks, that drive in a queue. This helps with reducing the air resistance experienced by the trucks at the back of the first truck and reduces the likelihood of being indulged in accidents.

This simulation enabled as to add new trucks on our own and to remove them as we wish. The trucks were not hard coded, in fact they were soft code, that provided the ease of adding or removing the trucks. A dynamic Platoon Leader selection was implemented that helped in determining the leader of the platoon. An important aspect of truck platooning is that trucks must be able to communicate with each other. This simulation included that feature in the most efficient and effective manner. Moreover, a feedback system was integrated, that basically ensured that everything is running fine and there are no possible errors in the system.

A website monitoring system was incorporated as well, where steering the speed or acceleration of the Platoon Leader was possible by using the website interface. Altering the acceleration or speed of each individual truck was also possible, but that was done using CLI. Website monitoring also enabled us to view the trucks that are currently in the platoon, this could also be seen in CLI as well. Another additional feature that integrated by TEAM BENZENEBOTS was speed monitoring using the website interface, A dynamic speedometer was used, only the truck number was to be added on the website and the speedometer showed the speed of that specific truck at that point in time. The Speedometer was real-time, it was clearly evident to the fact that when acceleration was set to a value other than zero, the speed on the speedometer kept of changing as the acceleration that was inserted. Trucks were taken as threads or processes that communicated with each other to form a platoon. Microservices architecture was implemented in the whole project. Figure 1 below gives a better representation of the architecture.

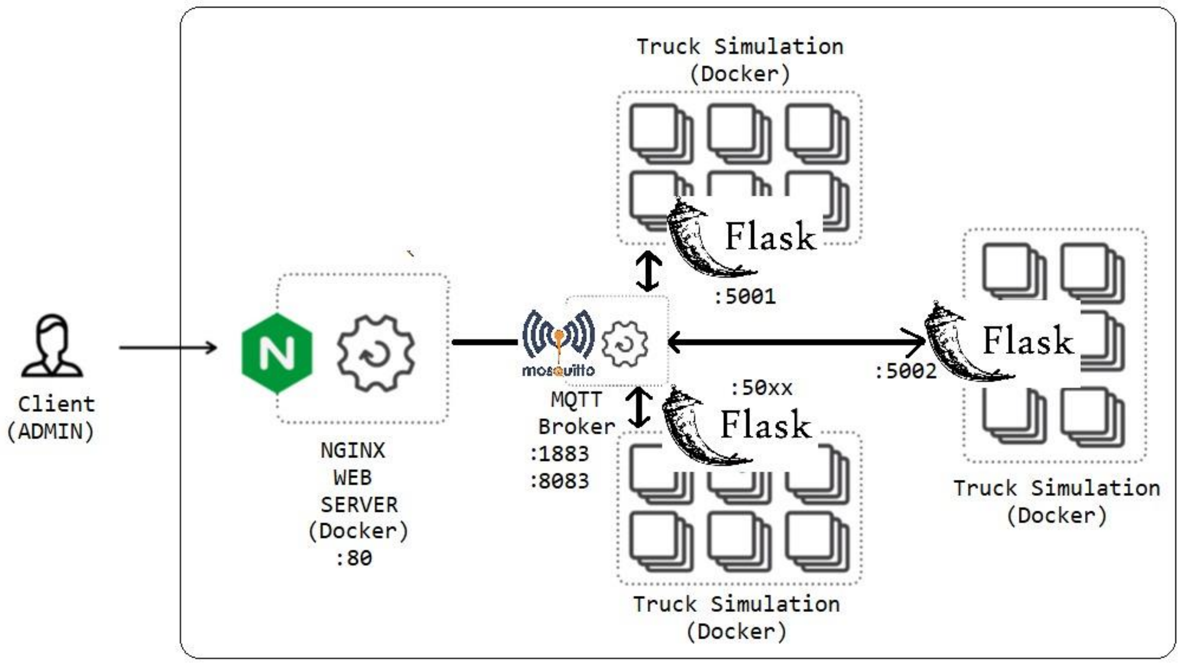


Figure 1

3 Product description

- Addition of a new Truck

Our simulation empowers us to add a truck at our own will. As briefly discussed earlier, The trucks were not Hard-coded; moreover, the trucks automatically reflected on the website monitoring system and the CLI as soon as the new truck was add.

- Removing a truck

As for addition of a new truck, removal of truck or multiple trucks is also possible. CLI or docker desktop could be used in order to execute that command. Incase, the platoon leader gets removed for the platoon, another trucks is selected as the leader based on the leader selection algorithm (Nearest Neighbor) that team BENZENE BOTS implemented.

- Obstacle Ahead

Incase there is an Obstacle ahead and the truck platoon needs to stop, that can easily be done by changing the speed of the platoon leader to zero. That basically helps the entire platoon to stop. The communication flow in the later section will help to understand how communication between the trucks works.

- Speed Synchronization

Another aspect that we were keen to implement in our project was speed synchronization. That means that all the trucks should share the speed. This is important in order to avoid collision. A collision could occur in case the trucks at the back have a higher speed as compared to trucks in the front. Setting the truck of the master truck automatically synchronizes the trucks at the rear. Communication section of this documentation will give a much clearer picture of the implementation of speed synchronization.

4 Technologies

- *Architecture Pattern : Microservices*
- *Programming Language: Python and JavaScript*
- *Application Programming Interface(API) : Flask and JSON*
- *Container service : DOCKERS*
- *Communication Protocol : MQTT*
- *Algorithm used(Platoon Leader) : Nearest Neighbour*
- *Web Interface : HTML*
- *Programming Model : CRUD*

5 Use-Cases

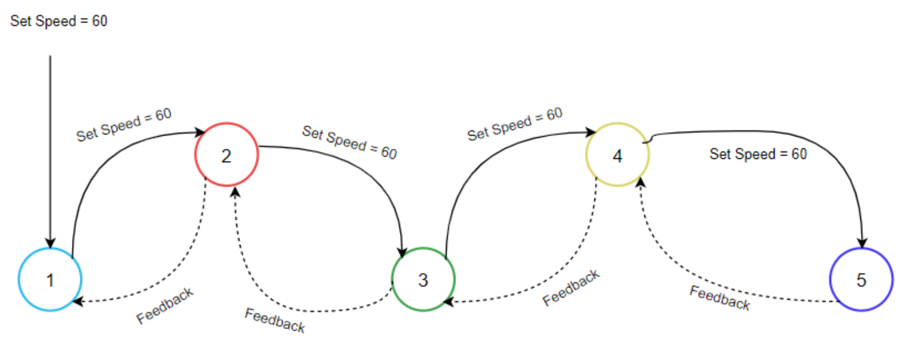
Describe the use cases of your simulation environment with a use case diagram and a brief description of the features.

6 Structure of the software

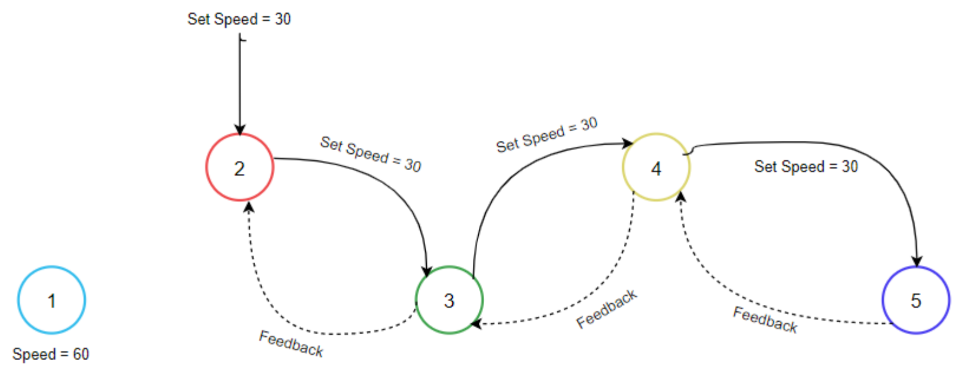
Describe the static structure of the simulation environment. Provide a class diagram for this purpose and briefly explain the classes or modules.

7 Communication flow

\\\diagrams are needed here\\\\
An important aspect of the entire project was to show communication between the induvial trucks. The diagram below shows on how the communication works, each circle represents a truck, and each arrow represents a message/signal. In the given example, the master truck received a signal to set the speed to 60, that information was passed on to truck 2 and from truck 2 it was passed on to truck 3 and so on. When the last truck receives the signal, it sends a signal to the master truck that the signal was received. This also helps us to implement a system that helps us to do a fault checks, incase the master truck does not receive a feedback signal from the last truck, that means there is a problem. Moreover, the route taken by the feedback signal is the same as before but in the opposite direction.

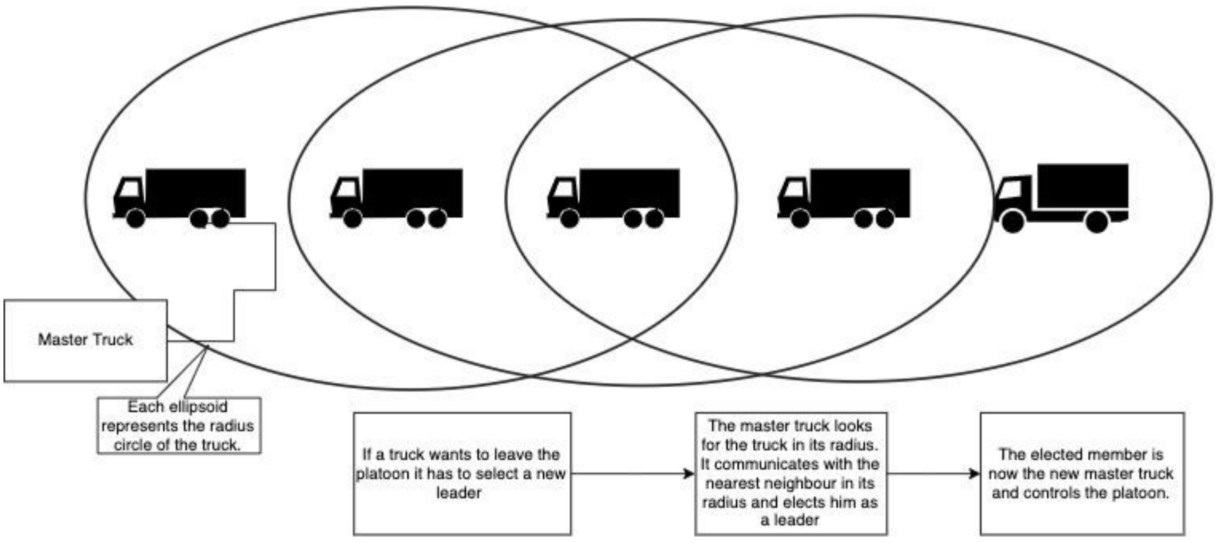


In the same example, now the signal was sent to truck 2 to change the speed to 30. This basically clarifies on how the whole mechanism works.



8 Election algorithms

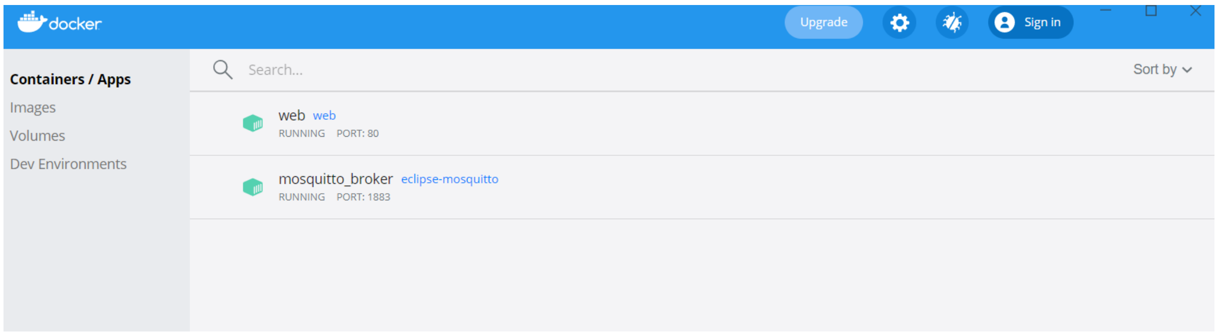
Nearest Neighbor was selected as the algorithm to elect the new platoon leader. Initially, the first truck forming the platoon is declared as the platoon leader. For instance, there is a platoon with 5 trucks(1,2,3,4,5) and they are arranged in ascending order with TRUCK 1 being the first truck and TRUCK 5 being the last one. Initially, Truck 1 is the platoon leader, when Truck 1 leaves, truck 2 gets to be the leader. This is due to the fact that Truck 2 was the nearest neighbor to Truck 1. Incase Truck 2 and Truck 3 leaves the platoon, Truck 4 becomes the leader. The illustration below gives a better representation of the entire scenario.


Master Truck
Each ellipsoid represents the radius circle of the truck.
If a truck wants to leave the platoon it has to select a new leader
The master truck looks for the truck in its radius. It communicates with the nearest neighbour in its radius and elects him as a leader
The elected member is now the new master truck and controls the platoon.

9 Operating instructions

ADDING TRUCKS TO THE PLATOON

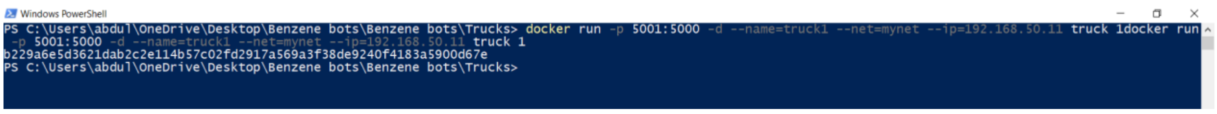
1. Run the docker files using special commands given in “how to run” file in “mosquito folder”- use PowerShell for convenience. After the step is executed correctly, the docker desktop should look like the window below. It must be noted that mosquitto_broker needs to run before the web.



web web
RUNNING PORT: 80

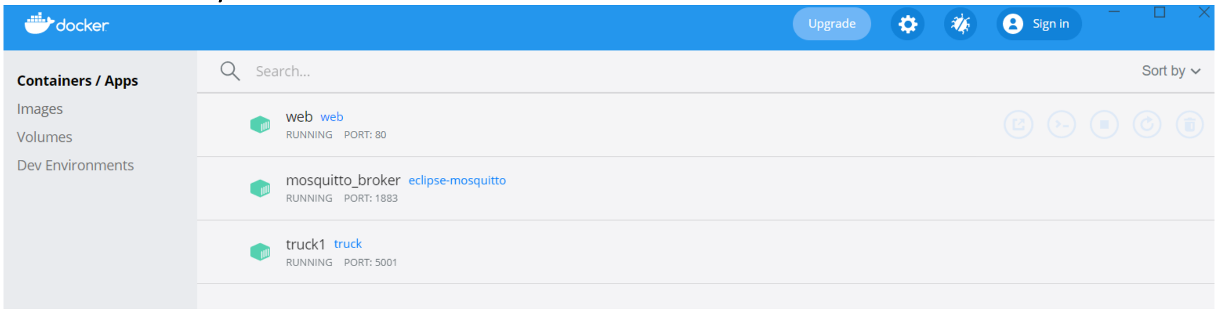
mosquitto_broker eclipse-mosquitto
RUNNING PORT: 1883

2. Now that things are running. We can proceed with adding trucks. For adding trucks, go to “Trucks” folder, open powershell there and execute the command mentioned in “how to run” notepad under the heading “add trucks”. A snippet of the command to be run is mentioned below.



```
PS C:\Users\abdul\OneDrive\Desktop\Benzene bots\Benzene bots\Trucks> docker run -p 5001:5000 -d --name=truck1 --net=mynet --ip=192.168.50.11 truck 1docker run
 -p 5001:5000 -d --name=truck1 --net=mynet --ip=192.168.50.11 truck 1
b229a6e5d3621dab2c2e114b57c02fd2917a569a3f38de9240f4183a5900d67e
PS C:\Users\abdul\OneDrive\Desktop\Benzene bots\Benzene bots\Trucks>
```

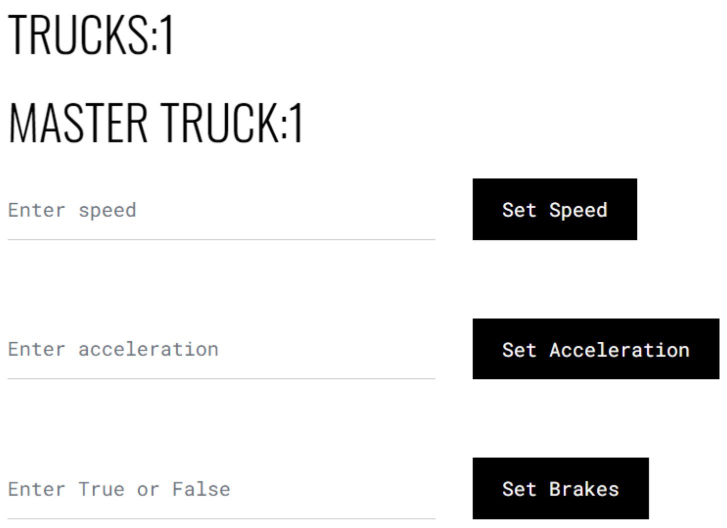
Docker and the web interface should look like the images below after the truck has been successfully added.



web web
RUNNING PORT: 80

mosquitto_broker eclipse-mosquitto
RUNNING PORT: 1883

truck1 truck
RUNNING PORT: 5001

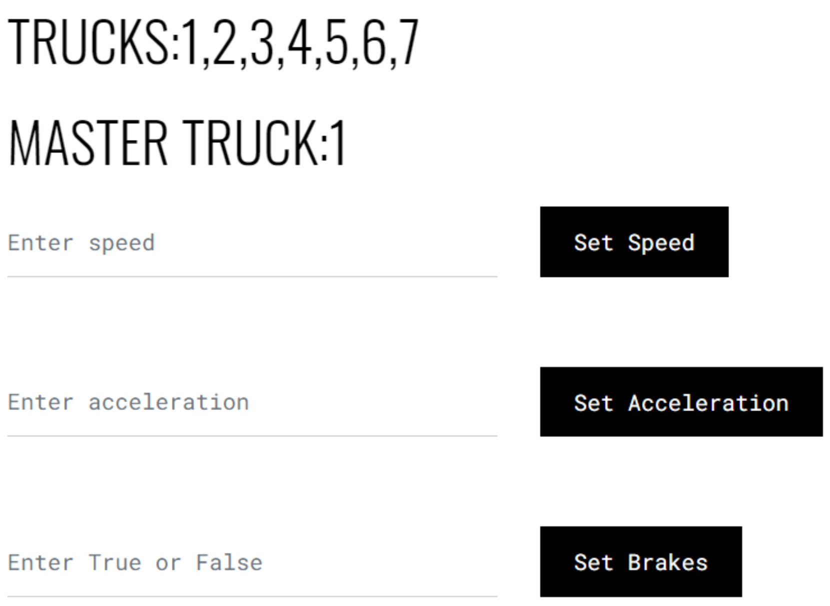


TRUCKS:1

MASTER TRUCK:1

3. With the same method, we will proceed forward and add 7 trucks to make our scenario more realistic. A snippet for the commands to be added to PowerShell are mentioned below. Our webpage should look like the snippet below after 7 trucks have been added, and the master truck is the TRUCK 1, as it can be clearly seen.

```
docker run -p 5001:5000 -d --name=truck1 --net=mynet --ip=192.168.50.11 truck 1
docker run -p 5002:5000 -d --name=truck2 --net=mynet --ip=192.168.50.12 truck 2
docker run -p 5003:5000 -d --name=truck3 --net=mynet --ip=192.168.50.13 truck 3
docker run -p 5004:5000 -d --name=truck4 --net=mynet --ip=192.168.50.14 truck 4
docker run -p 5005:5000 -d --name=truck5 --net=mynet --ip=192.168.50.15 truck 5
docker run -p 5006:5000 -d --name=truck6 --net=mynet --ip=192.168.50.16 truck 6
docker run -p 5007:5000 -d --name=truck7 --net=mynet --ip=192.168.50.17 truck 7
```

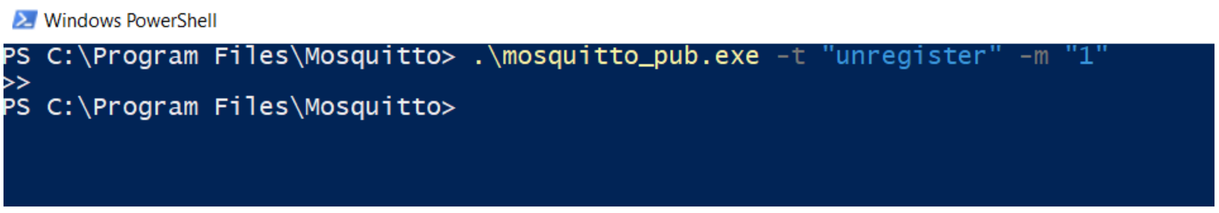

TRUCKS:1,2,3,4,5,6,7

MASTER TRUCK:1

REMOVING TRUCKS FROM THE PLATOON AND NEAREST NEIGHBOR

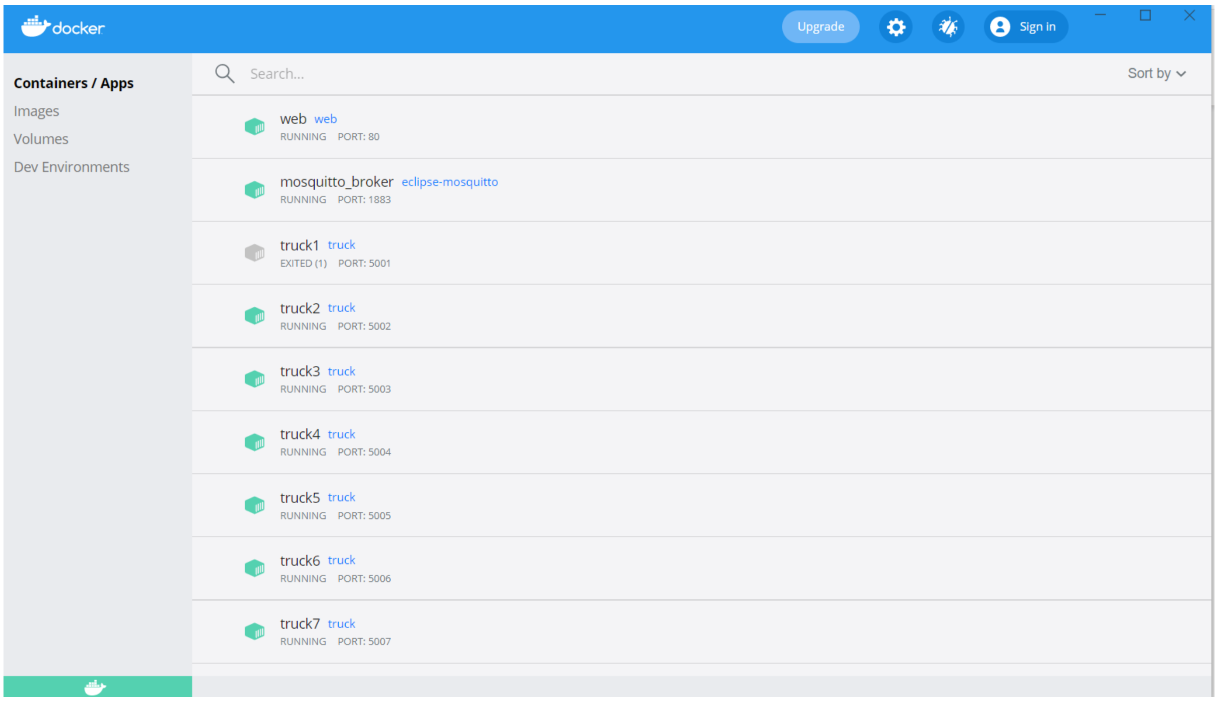
Trucks can either be removed from docker desktop or from the CLI itself.

1. For deleting or removing the truck from the platoon, simply the container can be deleted from the docker desktop.
2. For removing the truck by using CLI, Powershell needs to be opened in the mosquito file that is downloaded from the internet. Following snippet shows the command to be run.



```
PS C:\Program Files\Mosquitto> .\mosquitto_pub.exe -t "unregister" -m "1"
>>
PS C:\Program Files\Mosquitto>
```

Dockers container and the web monitoring interface should look something like this:



web web
RUNNING PORT: 80

mosquitto_broker eclipse-mosquitto
RUNNING PORT: 1883

truck1 truck
EXITED (1) PORT: 5001

truck2 truck
RUNNING PORT: 5002

truck3 truck
RUNNING PORT: 5003

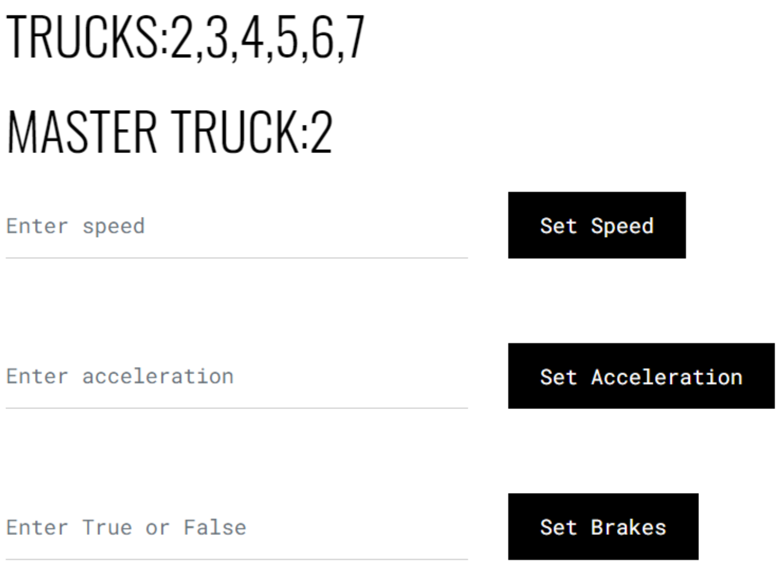
truck4 truck
RUNNING PORT: 5004

truck5 truck
RUNNING PORT: 5005

truck6 truck
RUNNING PORT: 5006

truck7 truck
RUNNING PORT: 5007

As it can be clearly seen that TRUCK 1 has exited.


TRUCKS:2,3,4,5,6,7

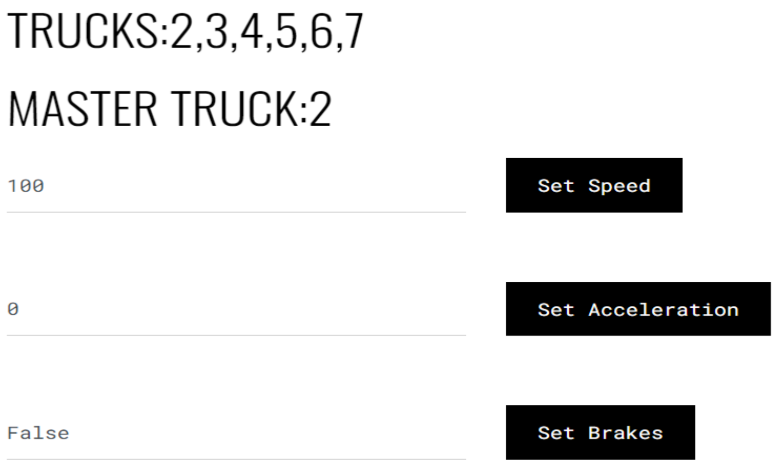
MASTER TRUCK:2

Web monitoring interface showing that TRUCK 1 is no longer in the platoon and TRUCK 2 is the master truck. This basically demonstrates how Nearest Neighbor works.

SETTING THE SPEEDS

1. In the earlier sections, our communication model was shown, this section will reinforce on the concept that was discussed earlier. Speed and Acceleration can be set for the master truck by using the web interface and CLI. For setting speed of induvial trucks, CLI must be used. In the example, We will first set speed to the master truck to 100 and check the speeds of the other trucks localhost, speedometer and dockers container.

   The following method can be used to select the speed of the master truck.


TRUCKS:2,3,4,5,6,7

MASTER TRUCK:2

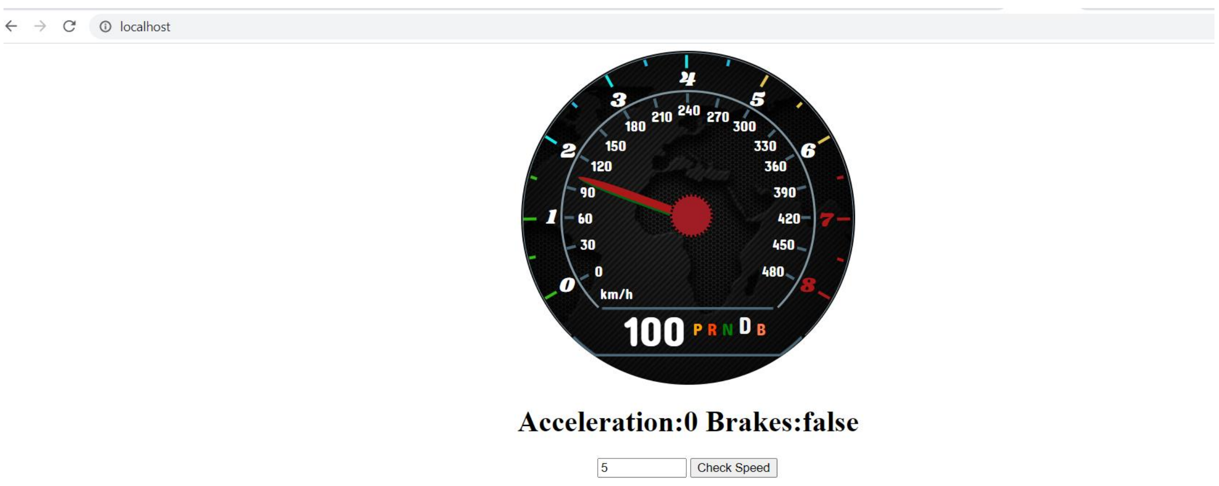
100 Set Speed

0 Set Acceleration

False Set Brakes

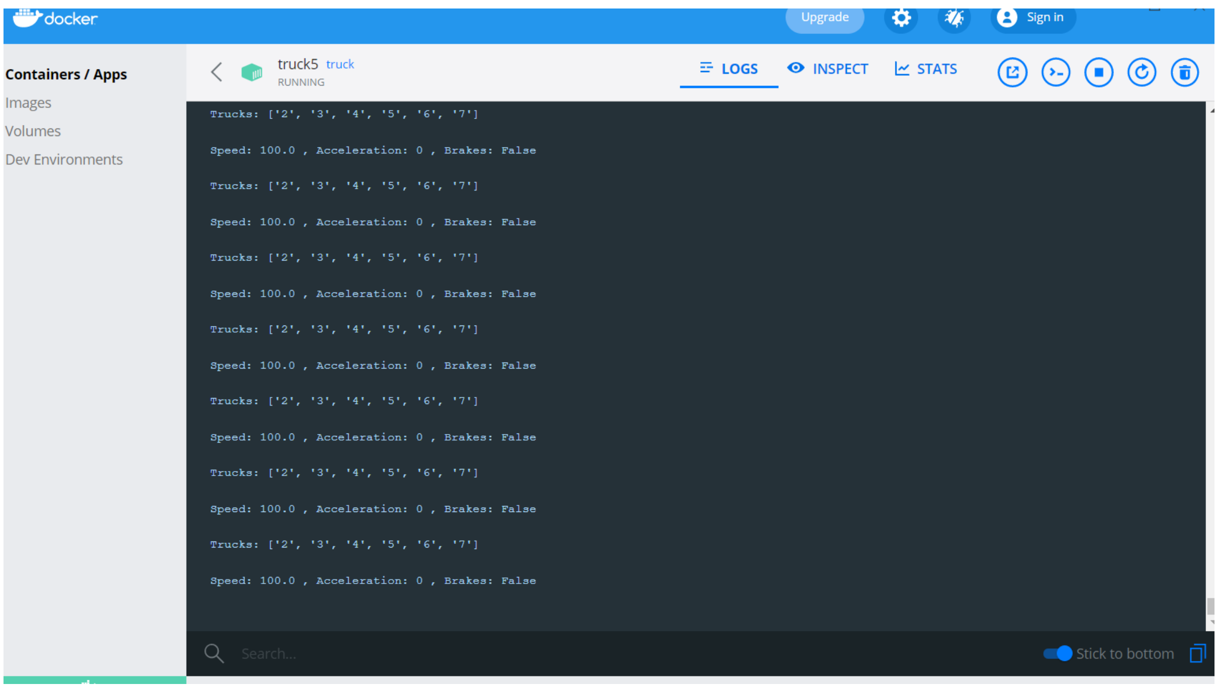
Now let check the speed of truck 5, that lend credence to our prospective regarding the communication model.

Type localhost in url for the speedometer.

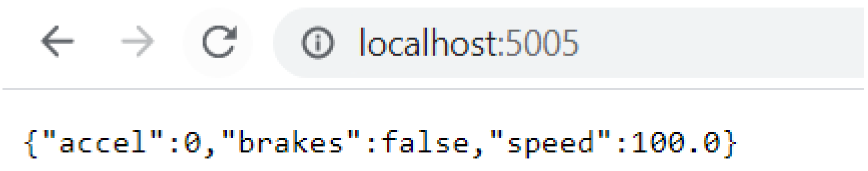


Acceleration:0 Brakes:false

5 Check Speed



```
Trucks: ['2', '3', '4', '5', '6', '7']
Speed: 100.0 , Acceleration: 0 , Brakes: False
Trucks: ['2', '3', '4', '5', '6', '7']
Speed: 100.0 , Acceleration: 0 , Brakes: False
Trucks: ['2', '3', '4', '5', '6', '7']
Speed: 100.0 , Acceleration: 0 , Brakes: False
Trucks: ['2', '3', '4', '5', '6', '7']
Speed: 100.0 , Acceleration: 0 , Brakes: False
Trucks: ['2', '3', '4', '5', '6', '7']
Speed: 100.0 , Acceleration: 0 , Brakes: False
Trucks: ['2', '3', '4', '5', '6', '7']
Speed: 100.0 , Acceleration: 0 , Brakes: False
Trucks: ['2', '3', '4', '5', '6', '7']
Speed: 100.0 , Acceleration: 0 , Brakes: False
```

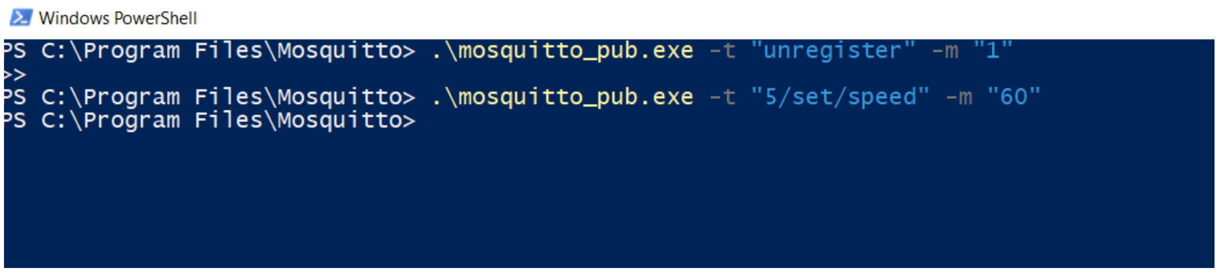


```
{"accel":0,"brakes":false,"speed":100.0}
```

Above are 3 methods of efficiently check the speed of truck 5.

2. Now we will carry forward the speed setting example and we will continue from the example where truck are running at a speed of 100. Now we will set the speed of truck 5 to 60. That would mean that truck 2 , 3 , 4 would be driving at 100 and truck 5 , 6 , 7 would run at 60.

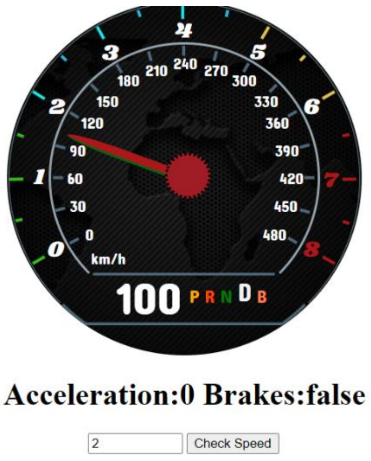
   For setting the speeds of the induvial trucks, the following commands must be used on PowerShell in Mosquito file (downloaded from the internet):



```
PS C:\Program Files\Mosquitto> .\mosquitto_pub.exe -t "unregister" -m "1"
>>
PS C:\Program Files\Mosquitto> .\mosquitto_pub.exe -t "5/set/speed" -m "60"
PS C:\Program Files\Mosquitto>
```

Now we will check the speed of TRUCK 2 and TRUCK 6 respectively

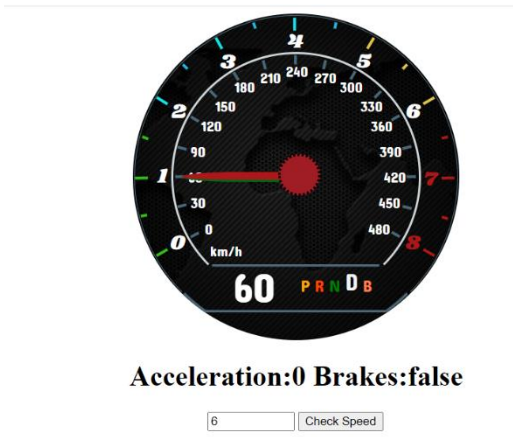
TRUCK 2 :



Acceleration:0 Brakes:false

2

TRUCK 6

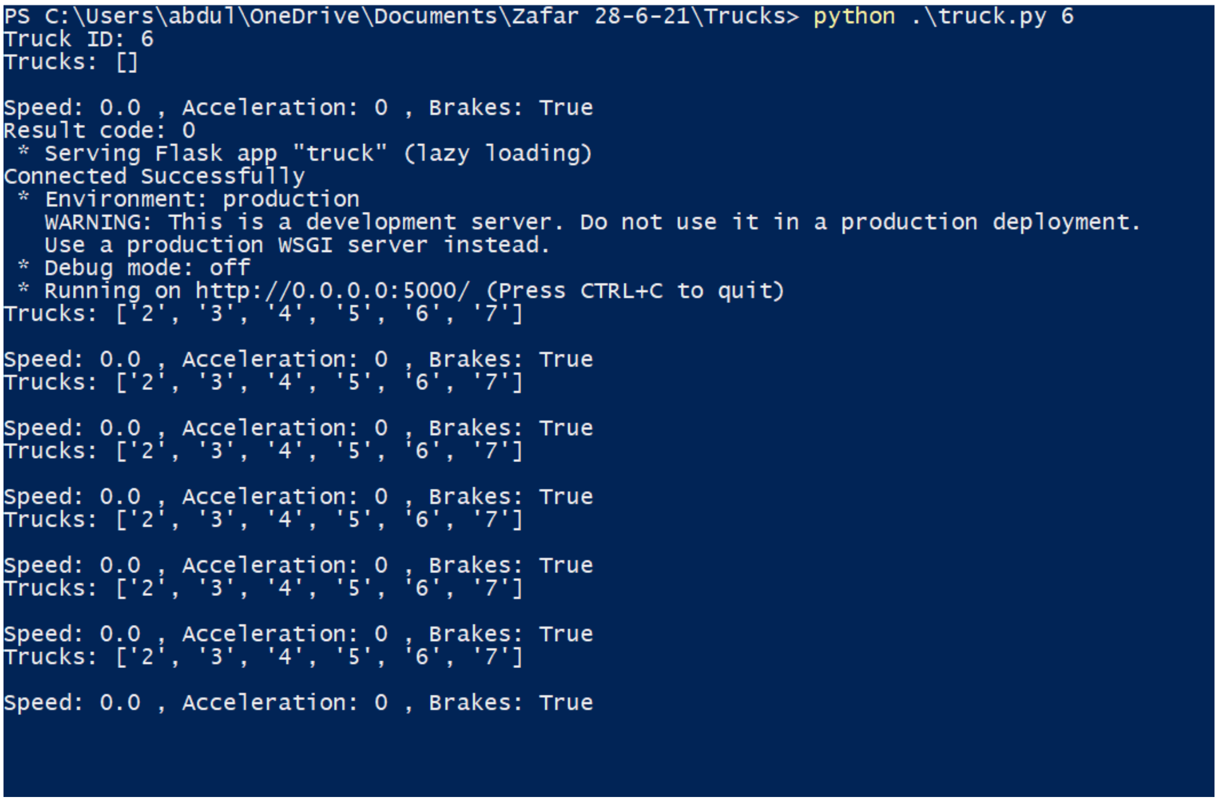


Acceleration:0 Brakes:false

6

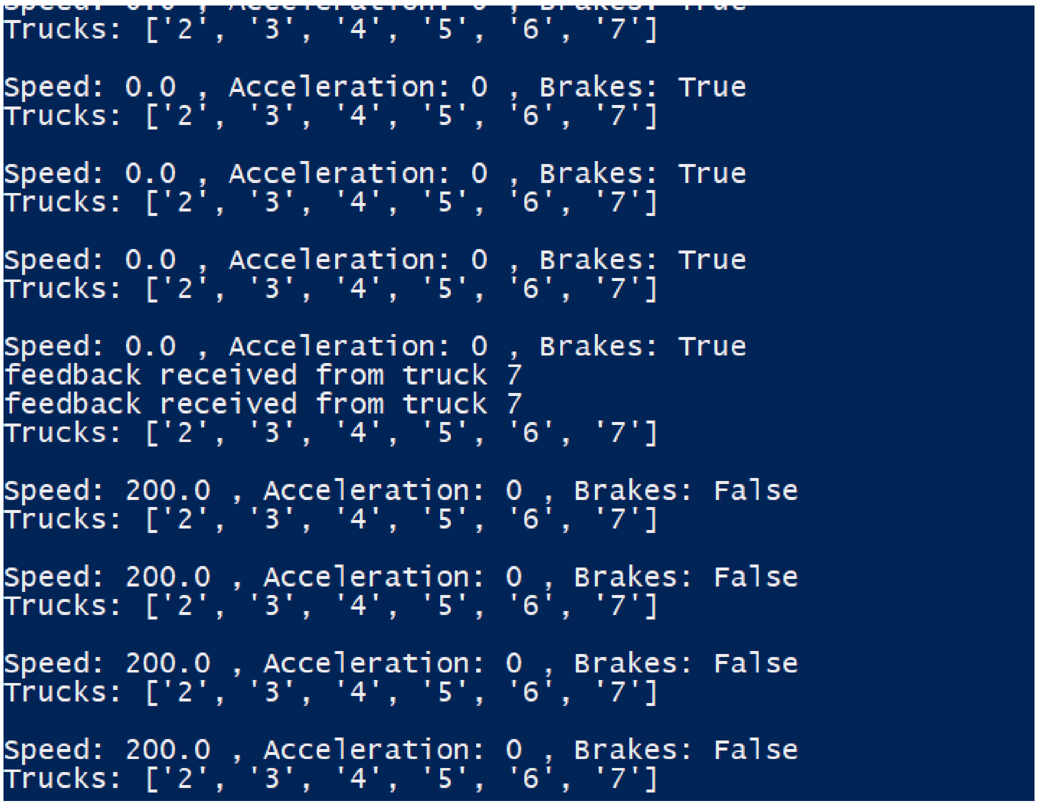
Feedback

To check for the feed back mechanism, We will initially set the speed to 0 and we will start monitoring the truck on CLI. The following command can be used to see the feedback message from Truck 7.



```
PS C:\Users\abdul\OneDrive\Documents\Zafar 28-6-21\Trucks> python .\truck.py 6
Truck ID: 6
Trucks: []

Speed: 0.0 , Acceleration: 0 , Brakes: True
Result code: 0
 * Serving Flask app "truck" (lazy loading)
Connected Successfully
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
Trucks: ['2', '3', '4', '5', '6', '7']

Speed: 0.0 , Acceleration: 0 , Brakes: True
Trucks: ['2', '3', '4', '5', '6', '7']

Speed: 0.0 , Acceleration: 0 , Brakes: True
Trucks: ['2', '3', '4', '5', '6', '7']

Speed: 0.0 , Acceleration: 0 , Brakes: True
Trucks: ['2', '3', '4', '5', '6', '7']

Speed: 0.0 , Acceleration: 0 , Brakes: True
Trucks: ['2', '3', '4', '5', '6', '7']

Speed: 0.0 , Acceleration: 0 , Brakes: True
Trucks: ['2', '3', '4', '5', '6', '7']

Speed: 0.0 , Acceleration: 0 , Brakes: True
```

Now the speed of the trucks is set to 200 and the feedback message is received from truck 7. It can be seen the snippet below.



```
Trucks: ['2', '3', '4', '5', '6', '7']

Speed: 0.0 , Acceleration: 0 , Brakes: True
Trucks: ['2', '3', '4', '5', '6', '7']

Speed: 0.0 , Acceleration: 0 , Brakes: True
Trucks: ['2', '3', '4', '5', '6', '7']

Speed: 0.0 , Acceleration: 0 , Brakes: True
Trucks: ['2', '3', '4', '5', '6', '7']

Speed: 0.0 , Acceleration: 0 , Brakes: True
feedback received from truck 7
feedback received from truck 7
Trucks: ['2', '3', '4', '5', '6', '7']

Speed: 200.0 , Acceleration: 0 , Brakes: False
Trucks: ['2', '3', '4', '5', '6', '7']

Speed: 200.0 , Acceleration: 0 , Brakes: False
Trucks: ['2', '3', '4', '5', '6', '7']

Speed: 200.0 , Acceleration: 0 , Brakes: False
Trucks: ['2', '3', '4', '5', '6', '7']

Speed: 200.0 , Acceleration: 0 , Brakes: False
Trucks: ['2', '3', '4', '5', '6', '7']
```

10 Installation

Describe how to install/unpack, launch and use the simulation.

//// Talk about dockers, mosquito, flask(pip install flask). I think that is what is need to be done.?????

10.1 Prerequisites

What are the requirements to run the software?

11 Allocation of tasks

Breakdown: who did what
Describe which team member did which tasks.

12 Sources

Provide sources on the technologies and algorithms you want to use.