Software Architecture (BE4M36SWA)

Smart Traffic Management

Semestral Project

Adel Shavalieva
Regina Latypova
Nadiya Yangirova
Zulfiia Galimzianova
Anastasiia Puzankova

Contents

Description of the chosen problem, explanation of the bottleneck and possible solutions

Traffic lights that use real-time data feeds are being used to smooth traffic load. We suggest that each traffic light mounted at a strategic place is a “client” with a bunch of sensors. The goal is to gather data about high traffic junctions and areas diverting vehicles from these places. The client can detect current traffic through its sensors and switch the light, also it is able to send data about the current light and traffic to server. So there is a “server” which controls the general situation in a city and is able to provide specific commands to its clients (such as switch the light) based on the data from different clients in order to achieve the best situation in general, not in the specific hot-spot. The server can analyze the information from sensors further and figure out alternative routes, as well as better traffic signaling to ease congestion.

So the **bottleneck** is a server and the reason of it is a server-client architecture itself, because server is going to process a huge amount of data.

Solutions:

1. One possible **solution** is the following architecture:

   The clients communicate with the neighbour clients and adapt to changing traffic conditions to reduce the amount of time that cars spend idling. The new technology monitors vehicle numbers and makes changes in real time to avoid congestion wherever possible.

2. Another possible **solution** is to divide a city to regions with their own servers. By the “region” we suppose a strategic place with a usual heavy

traffic that can be, for example, a main road of a city with some area around it.

As the **quality aspects** we suggest to measure:

1. For each traffic light - the average time which car spends waiting for a green light
2. The number of traffic lights at which car stops during an “average route” (by the “average route” we mean an average route inside a city for a vehicle i.e. home-to-work or something like this)
3. The average waiting time for a green light during the “average route”

High level view of the system

Sensors' main goal is to find out how many cars are on the road what determines if there is a traffic jam.

Traffic light controller is on each traffic light. All sensors on this traffic light are connected to this traffic light's controller, i.e. they send real-time updates about their current states through fiber-optic cables. So controller aggregates all the data from the sensors. And it sends data through the wireless connection to the segment controller.

Traffic sensors on the road with its controller - sensors which are on the highway out of the traffic light. They detect traffic on the road outside intersections. Each such sensor (its controller) inside one segment is connected to this segment controller through the wireless connection.

Section controller serves as a central controller. It controls every traffic light in the segment and collects the information from them and traffic sensors. It sends data to the server and gets from it behavioral commands. It is equipped with 3G module to obtain fast Internet connectivity and real-time update.

Wireless bridge - when there is more than one section to be controlled, it links multiple segments to one central controller by creating a wireless network.

Cloud/Server provides monitoring and automatic/manual control traffic light.

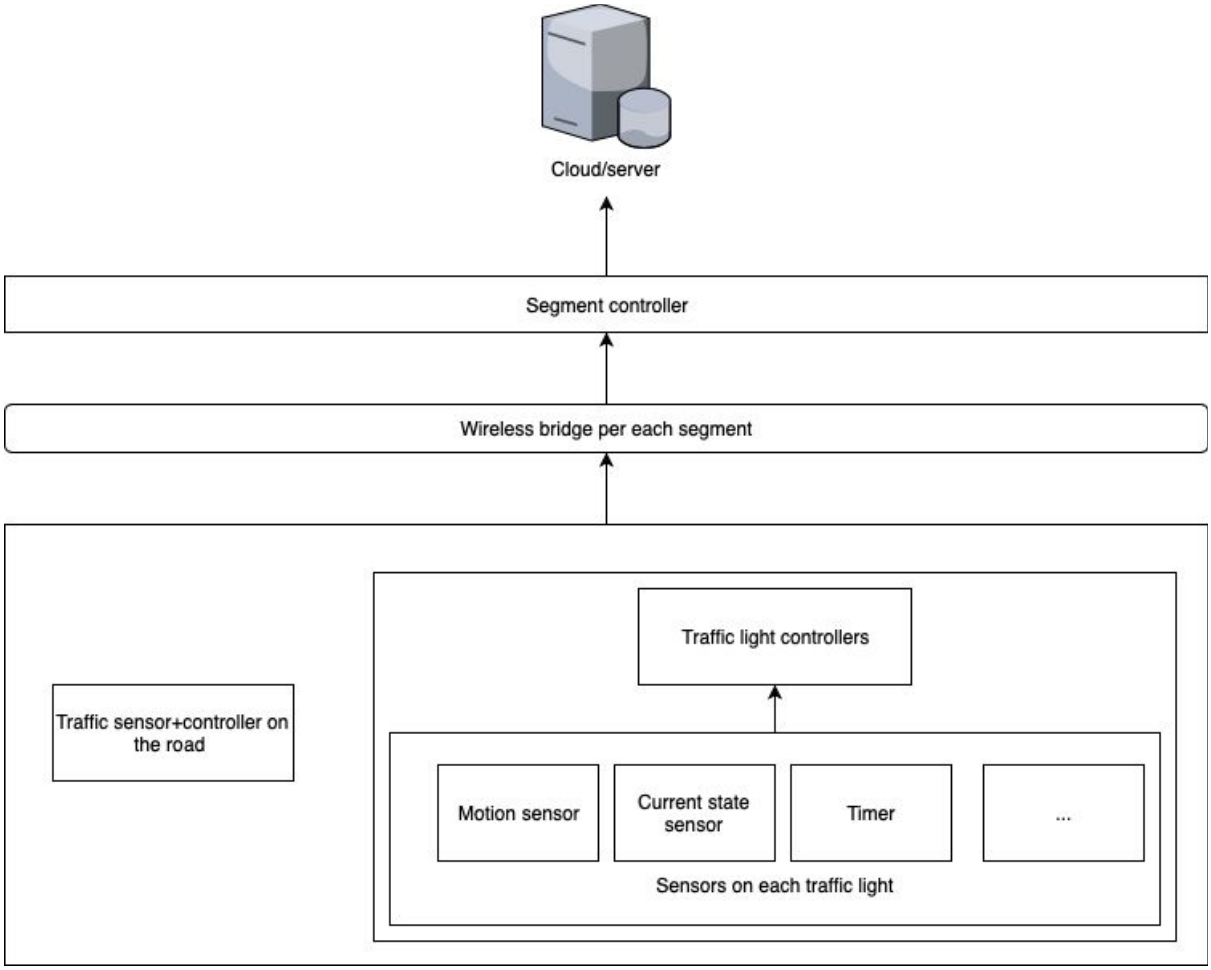
Cloud/server
Segment controller
Wireless bridge per each segment
Traffic sensor+controller on the road
Traffic light controllers
Motion sensor
Current state sensor
Timer
...
Sensors on each traffic light

Figure 1

Logical view

1. The second architecture

We decided to take a MVC architectural pattern as a base for our system.

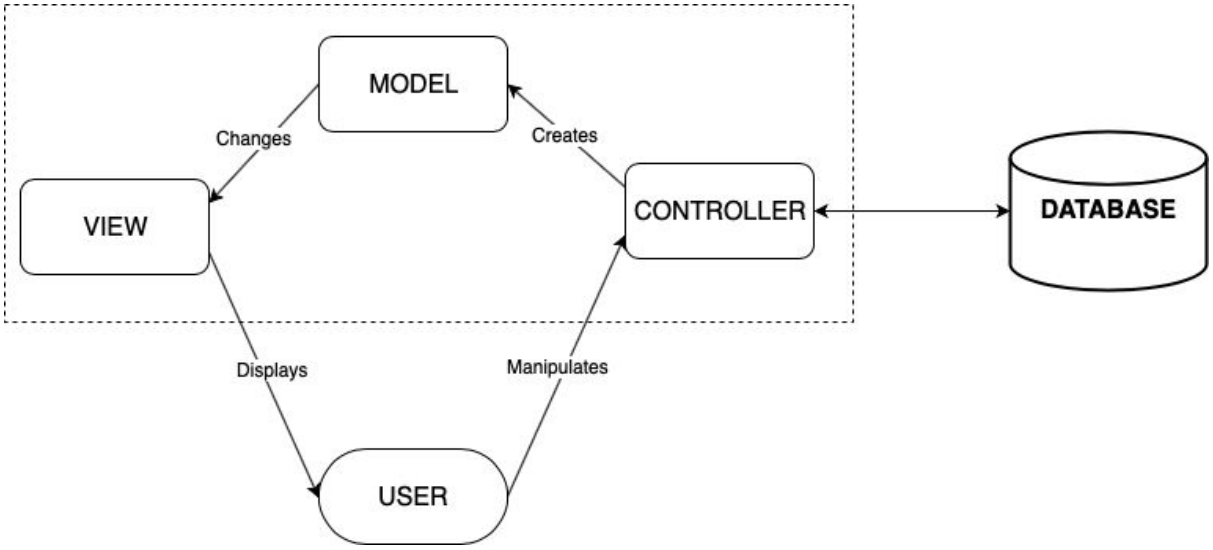

Figure 2

For our system:

The **USER** are the cars on the road because a fact that car is on the road makes the sensors on the traffic lights to generate incoming data for the system.

The **MODEL** is a kind of algorithm using which the traffic light changes the signal (the light).

The **VIEW** is the signal on the traffic light.

The **CONTROLLER** gets data from the sensors, aggregates and processes it and generates the algorithms for each individual traffic light (for the model).

View classes:

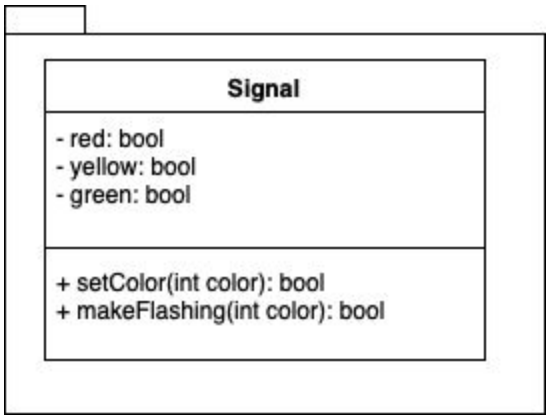

Figure 3

'Signal' is a class that represents traffic lights and their actions.

setColor is a method that changes color of a traffic light, parameter of the method is a number meaning a color, e.g. 0 - red, 1 - yellow, 2 - green.

makeFlashing is a method setting flashing light of a color according to its parameter.

Model classes:

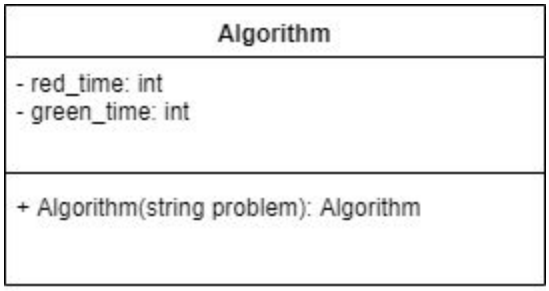

Figure 4

'Algorithm' is a class storing signal timing.

Controller classes:

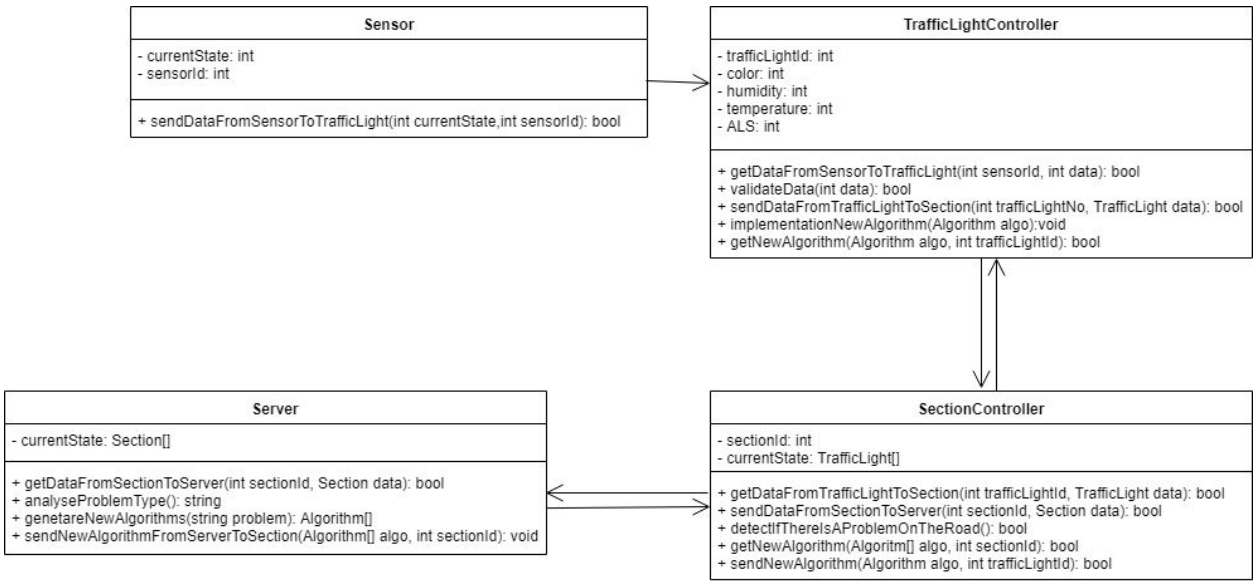

Figure 5

Forward propagation:

'Sensor' represents a sensor and stores its value (temperature, humidity etc.) This class has method *sendDataFromSensorToTrafficLight* which sends data to its TrafficLightController.

'TrafficLightController' collects values of multiple sensors, verify their validity and transmits it to its SectionController.

'SectionController' receives data (list of sensors' values) from TrafficLightControllers of this section. It has its own id, currentState that is basically a list of current states of this section's traffic lights and also it stores a history of states of the previous period (i.e. last hour). SectionController can analyze the current situation on the road through a method *detectIfThereIsAProblem* and detect changes on the road (e.g. a traffic jam, some accidents on the road, an increased machine traffic). If there is some problem on the road, SectionController sends data from sensors to Server.

'Server' determines a type of problem according to the received data. Then it generates new algorithms for the corresponding SectionControllers.

Backward propagation:

After **Server** generated new algorithms, it sends them to the SectionControllers.

If **SectionController** receives some new algorithms from Server, it will send them to appropriate TrafficLightControllers.

Once **TrafficLightControllers** recieved new algorithms from their SectionController, they set a signal timing according to it.

Sensor state diagram:

'Sensor' detects events or changes in its environment and sends the information to Traffic light controller.

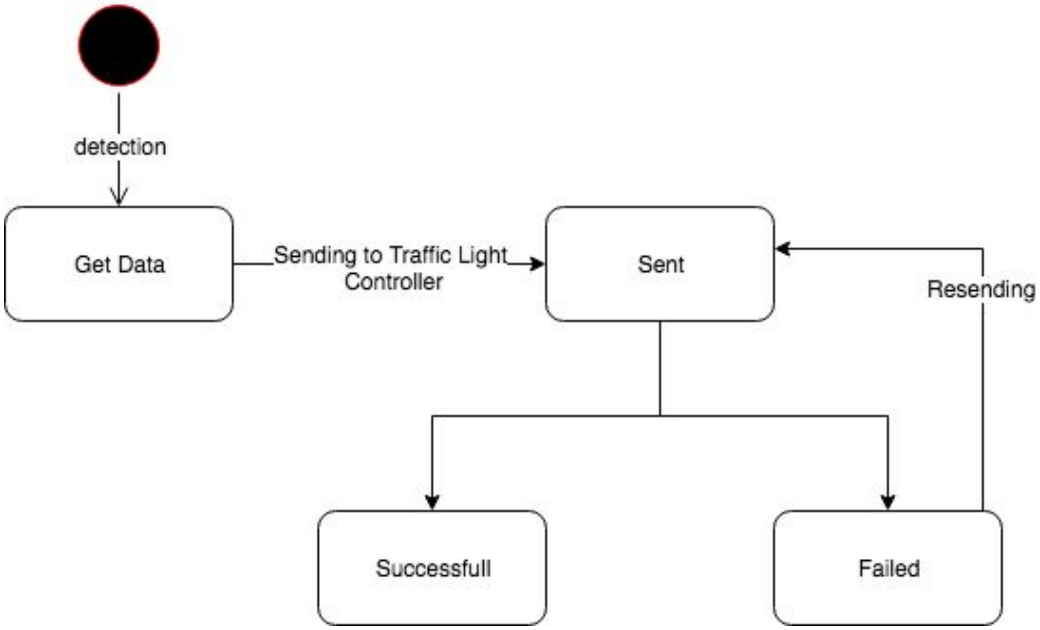

Figure 6

Traffic Light diagram:

After receiving the data Traffic Light Controller check its validity. If the data is solid the Traffic Light Controller will send it to Section Controller.

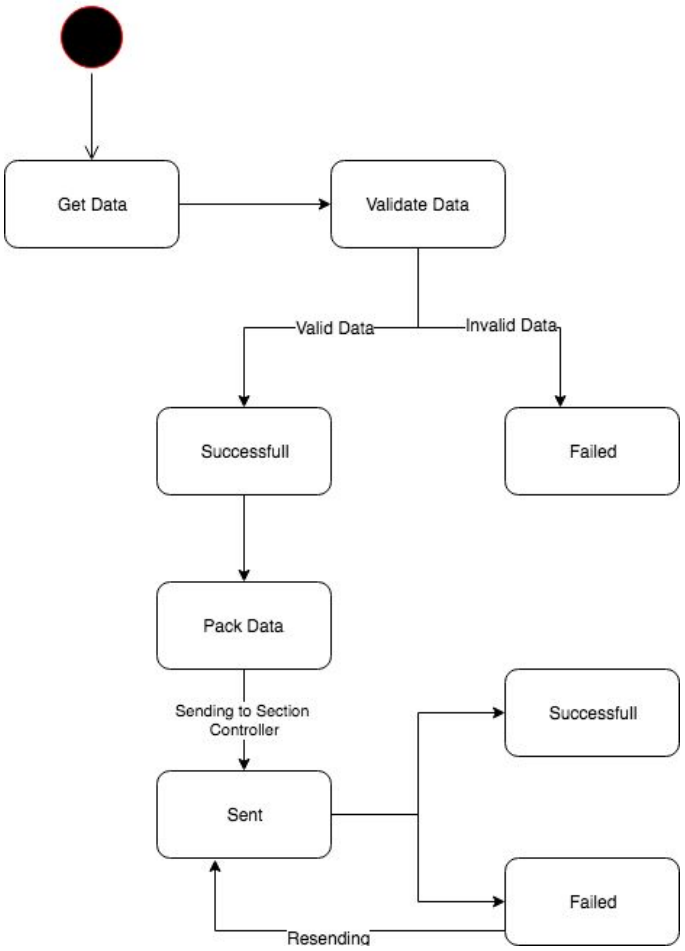

Figure 7

Section Controller and Server diagram:

After receiving the data Section Controller saves it to database (cache). Section controller analyse the data and detect if there is a some problem on the road. If the it finds out the problem, Section Controller will send the data to the Server. After that the Server analyse the problem and create new algorithm that will be sent to Section Controller.

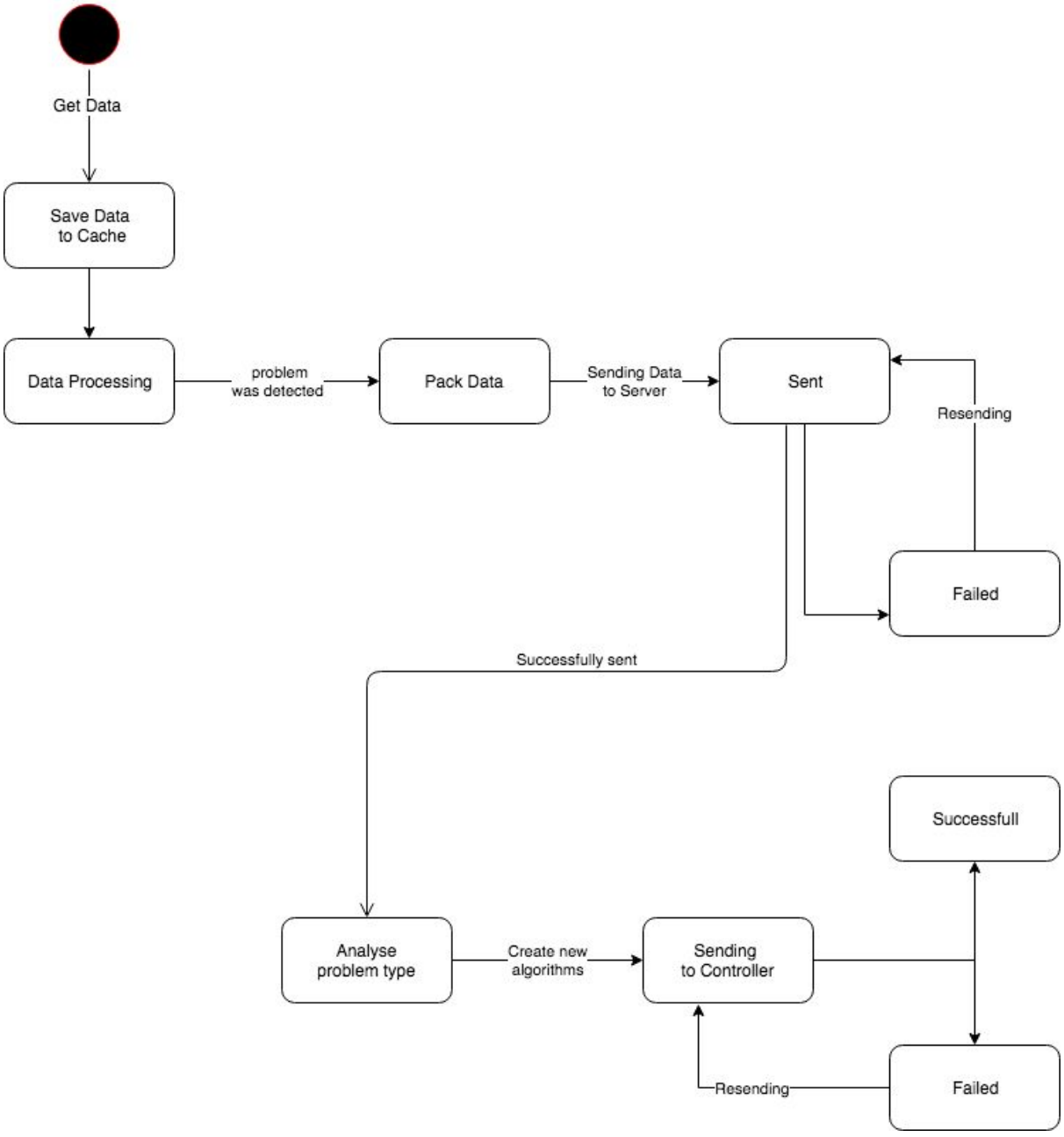
Get Data
Save Data to Cache
Data Processing
problem was detected
Pack Data
Sending Data to Server
Sent
Resending
Failed
Successfully sent
Successfull
Analyse problem type
Create new algorithms
Sending to Controller
Resending
Failed

Figure 8

2. The first architecture

The first architecture is the same but without SectionController:

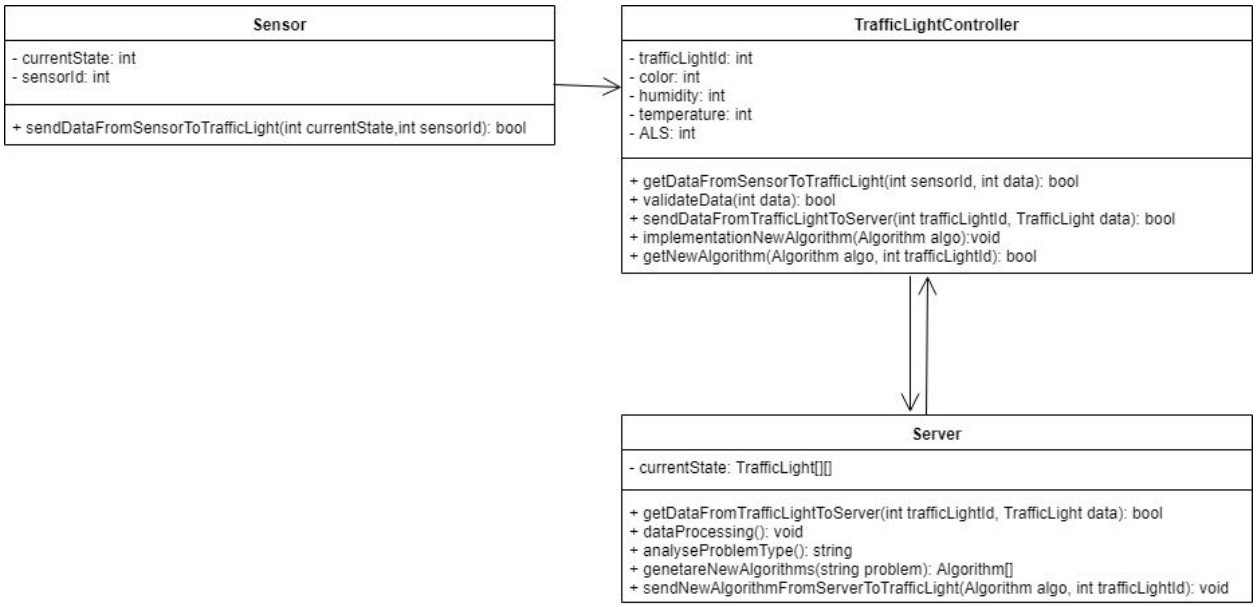

Figure 9

All of the components of this mean the same as at the previous diagram.

Process view

We designed 2 different architectures for Smart Traffic Management System in order to explore what is more effective.

First design stands for the architecture without computing unit controlling traffic lights (Section Controller). In this case the system works as follows: sensors send their data to Traffic Light Controller which aggregates all the data from the sensors and sends it to Server. It happens every time unit. The server analyzes the received data in order to decide if there are some accidents on the road. If there are then it generates the algorithm to solve the problem. So then Server sends it back to the Traffic Light Controller which changes the lights according to the received algorithm. Obviously in this case Server is the bottleneck.

The second design of architecture has Section Controller as a middle module in data transfer between Traffic Light Controller and Server. The idea is to decide whether there is a traffic accident on the road “on the fly” of data from Traffic Light Controller and send the data from it to Server only if the traffic accident takes place on the road. The purpose of this is to reduce the load on the Server that will be done because it will not process all the data from all the sensors anymore, but only the data containing the problem.

1. The first architecture

Send data from the Sensor to Traffic Light Controller:

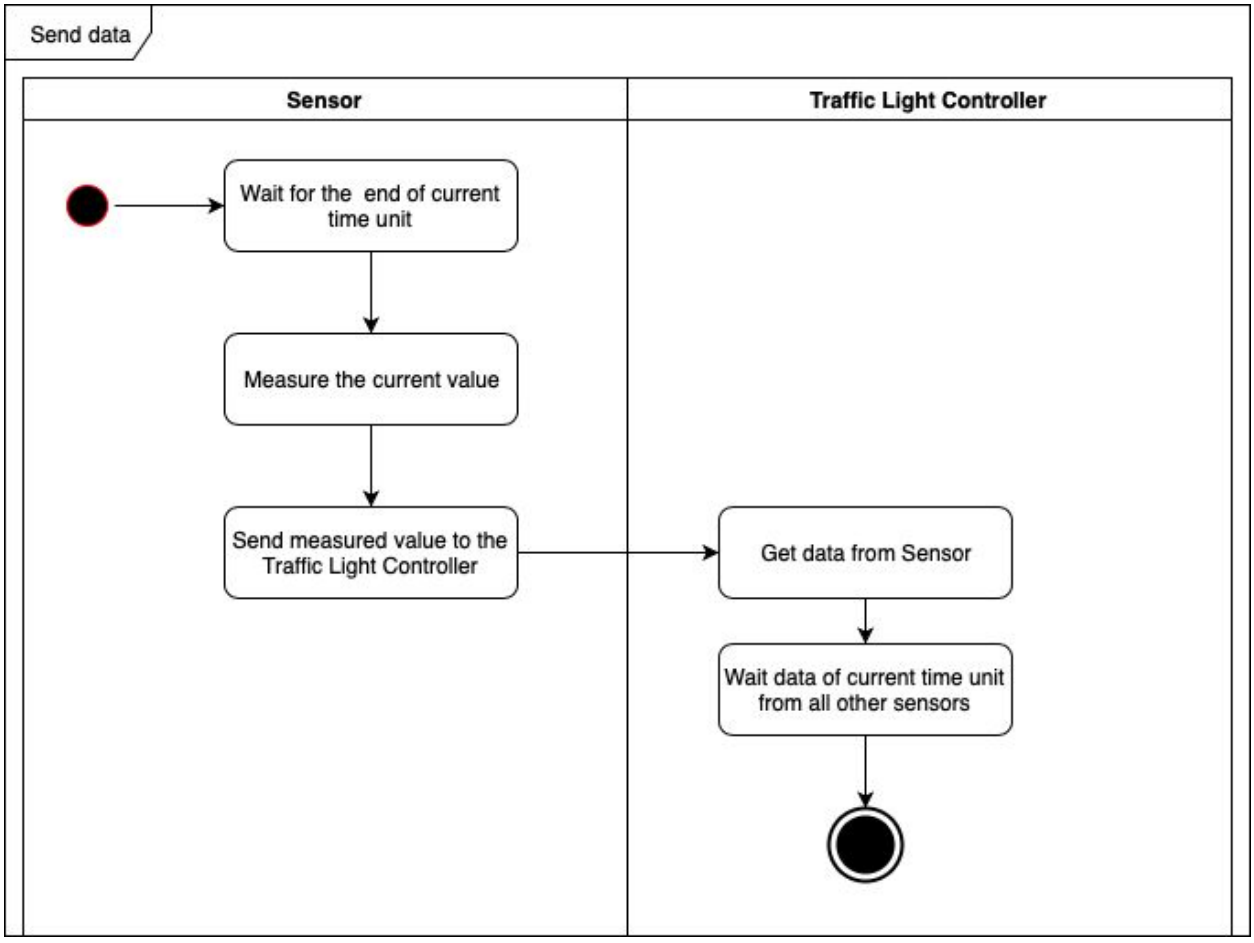

Figure 10

Send data from Traffic Light Controller to Server:

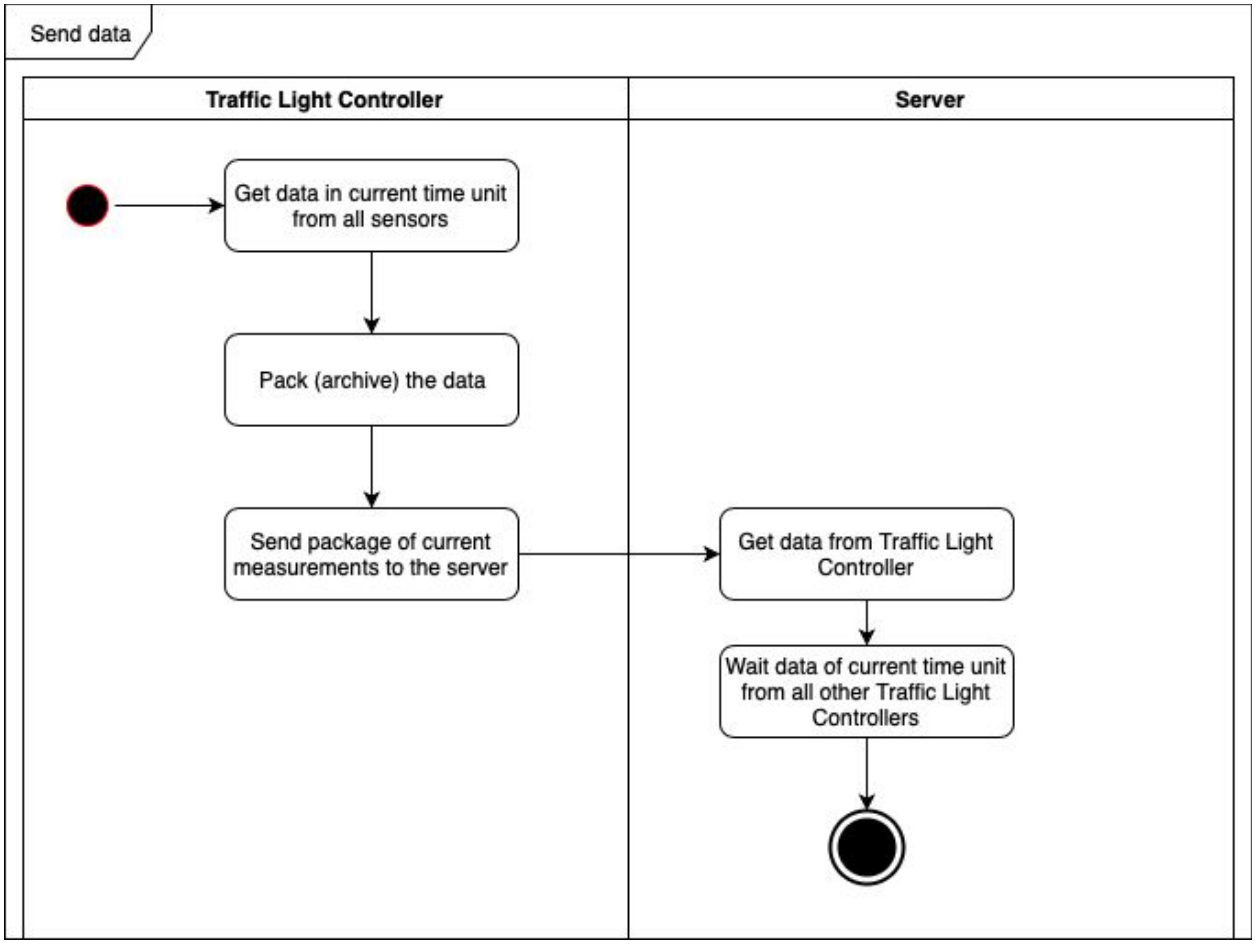

Figure 11

Server detected an accident:

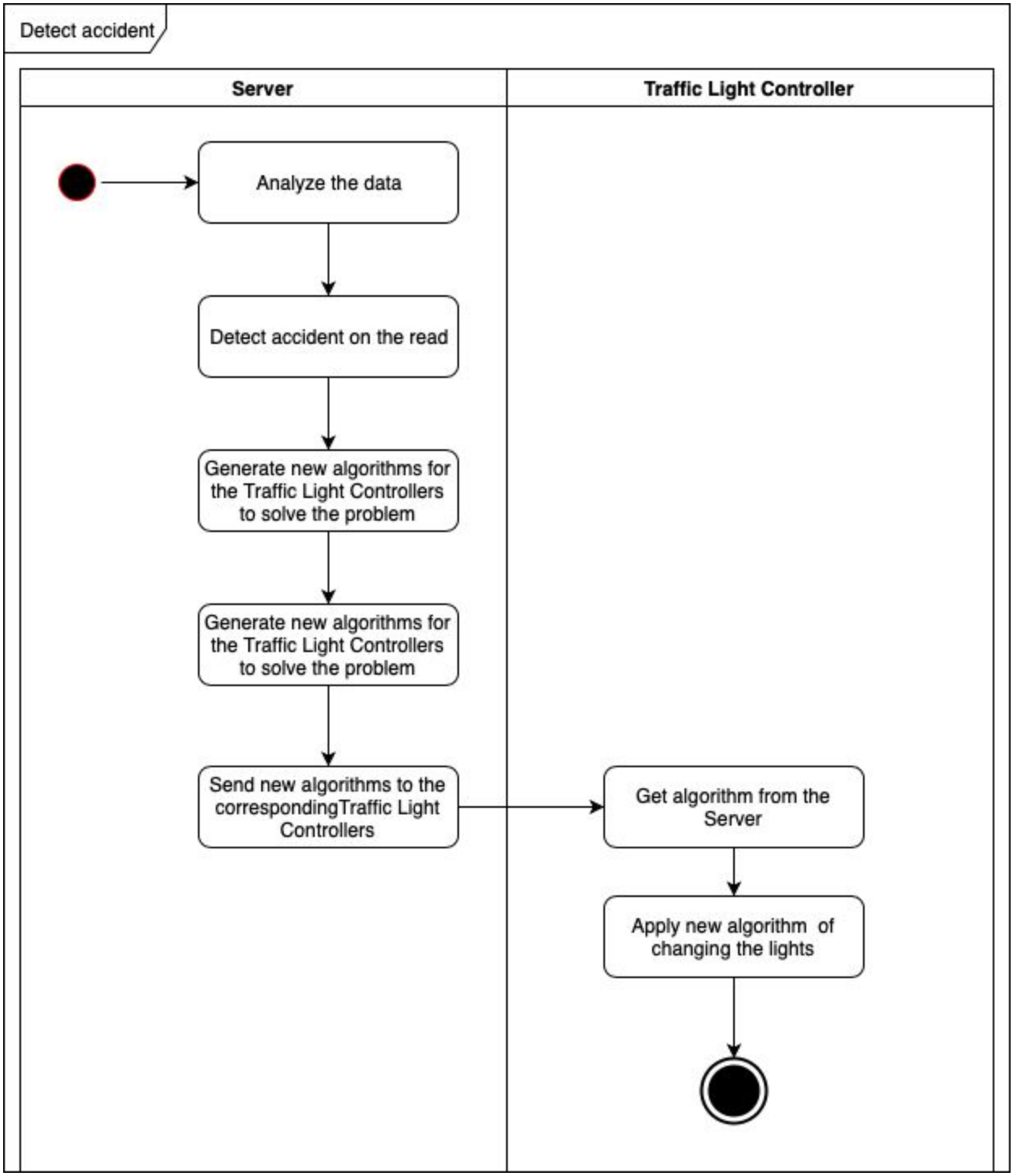

Figure 12

2. The second architecture

Send data from the Sensor to Traffic Light Controller:

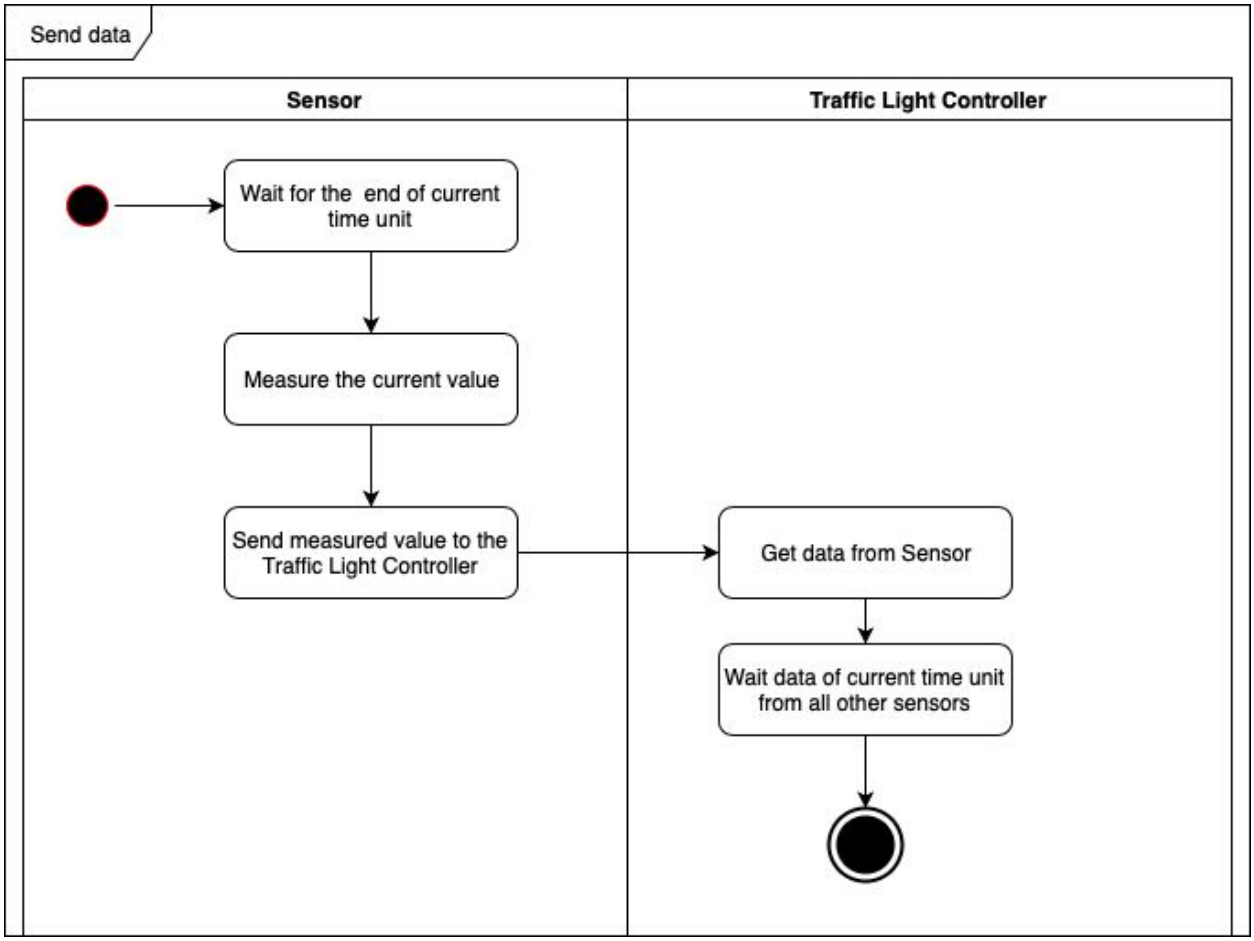

Figure 13

Send data from Traffic Light Controller to Section Controller:

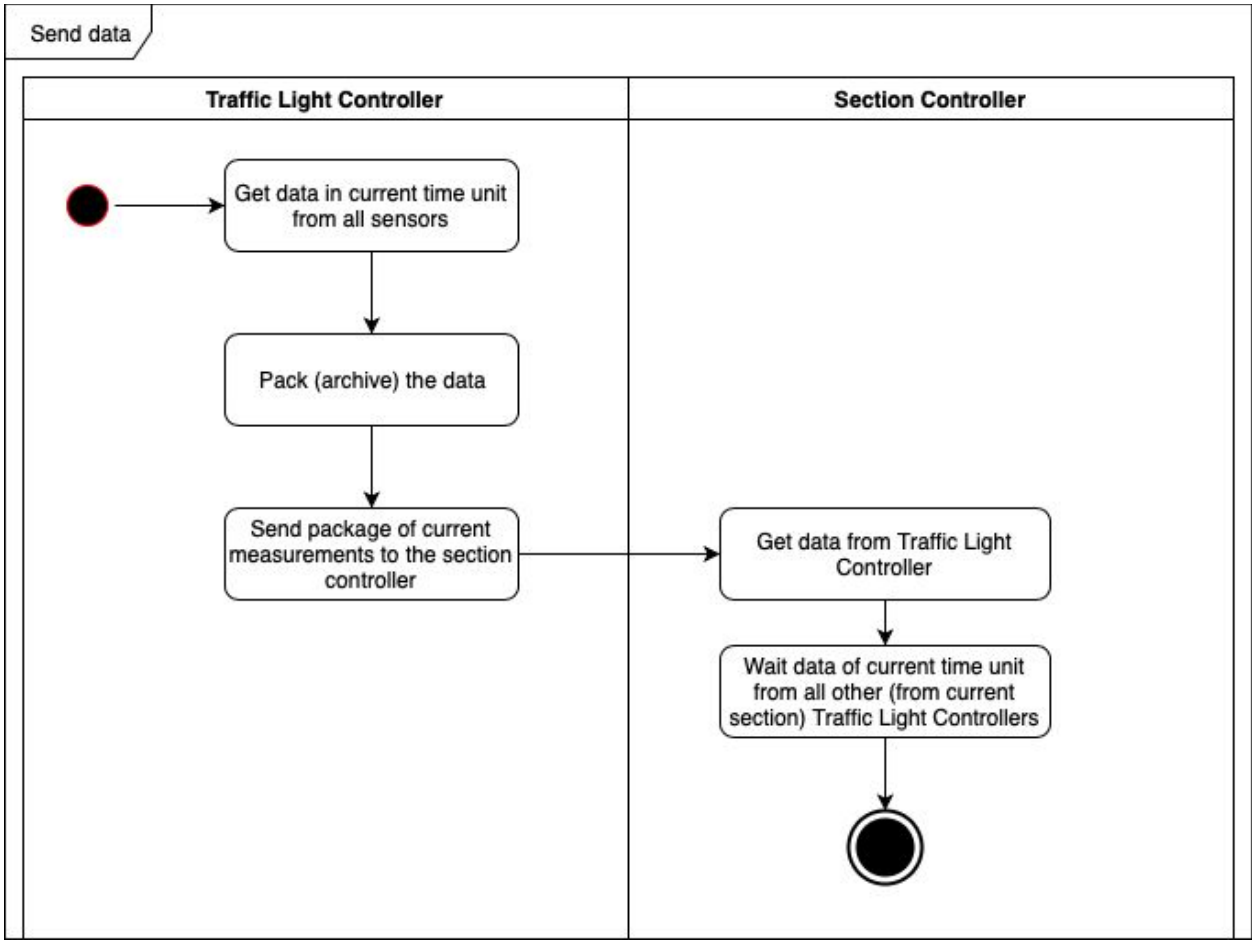

Figure 14

Section Controller detects an accident:

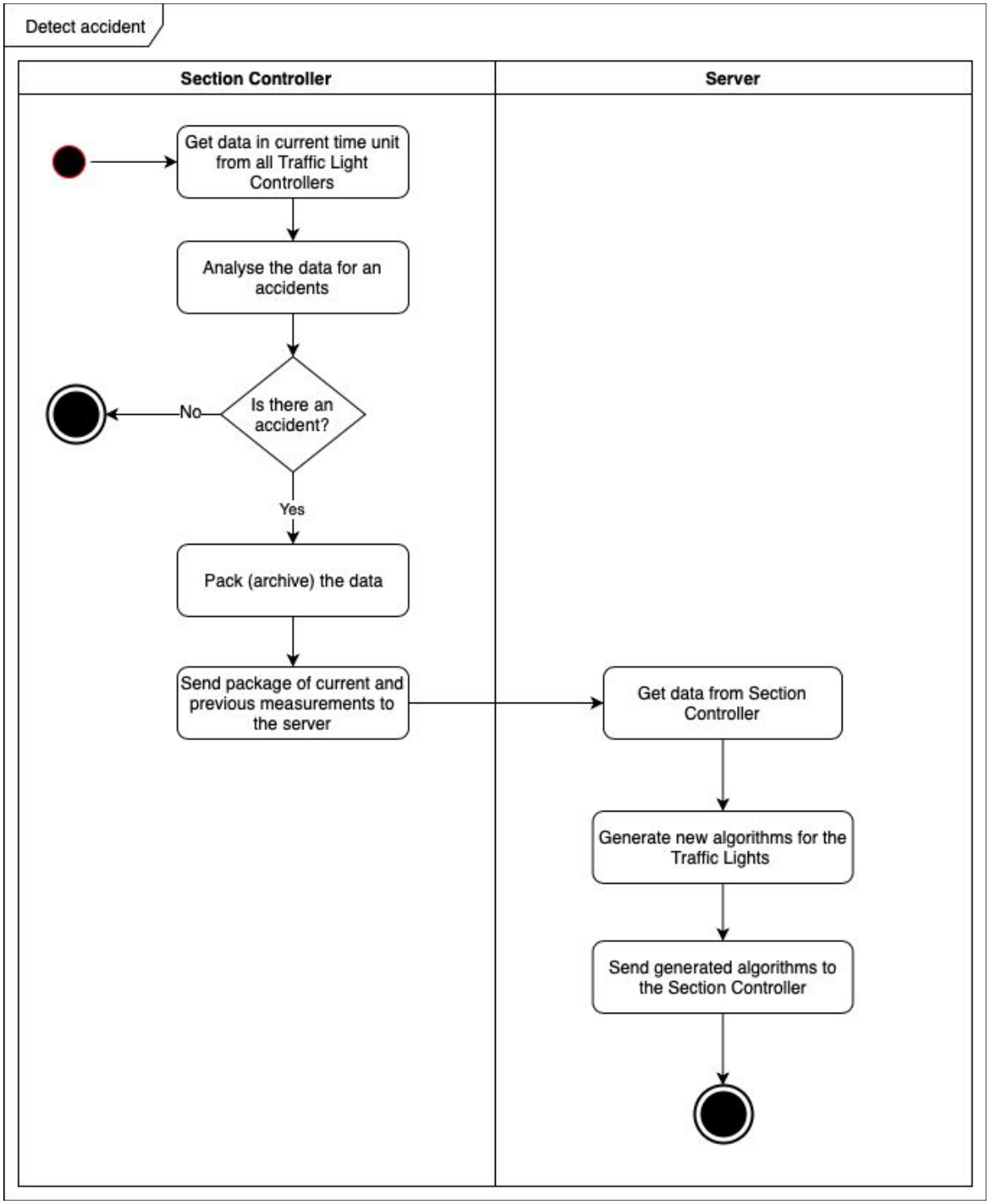

Figure 15

Traffic Light gets new algorithm:

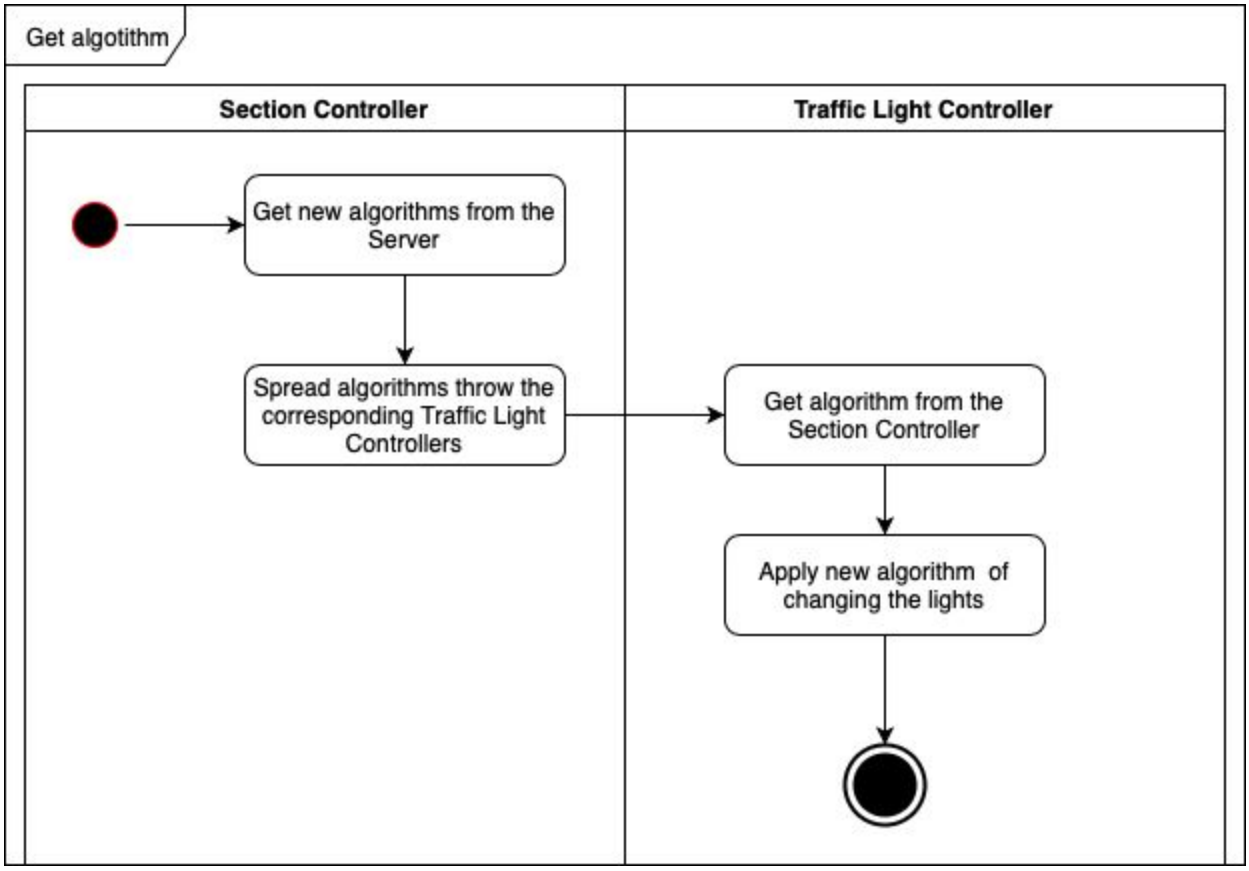

Figure 16

Development View

For processing of a large amount of data from sensors and creating on their basis an algorithm for the work of traffic lights the Big Data and Machine Learning methods are usually used. In the case of the first architecture, almost all work takes place at the server. Server processes all the data from the Traffic Light Controllers, detects the problem, which is a binary classification, analyzes this problem and creates new algorithm.

In case of the second type of architecture, Section Controller takes some tasks from Server. It processes all data from Traffic Light Controllers, performs the binary classification. After that if it is necessary, it will send the data further to Server. And server creates the algorithms for each Traffic Light.

Creating new algorithm is a regression problem that is most time consuming and resource intensive task in this system. In the case of the first architecture, the regression problem is solved a way more often than in the architecture with Section Controller. So by adding new element like Section Controller we decrease the amount of work in term of development view.

There are some major algorithms which are used to classify and optimize the data. They are

1. k-nearest neighbour
2. Decision trees
3. Naïve Bayes
4. Logistic regression
5. Support vector machines
6. Artificial Neural Networks

As a programming language can be used Python or C++. There are a lot of nice libraries and packages of machine learning algorithms in them.

To describe system components we used the component diagrams.

1. The first architecture

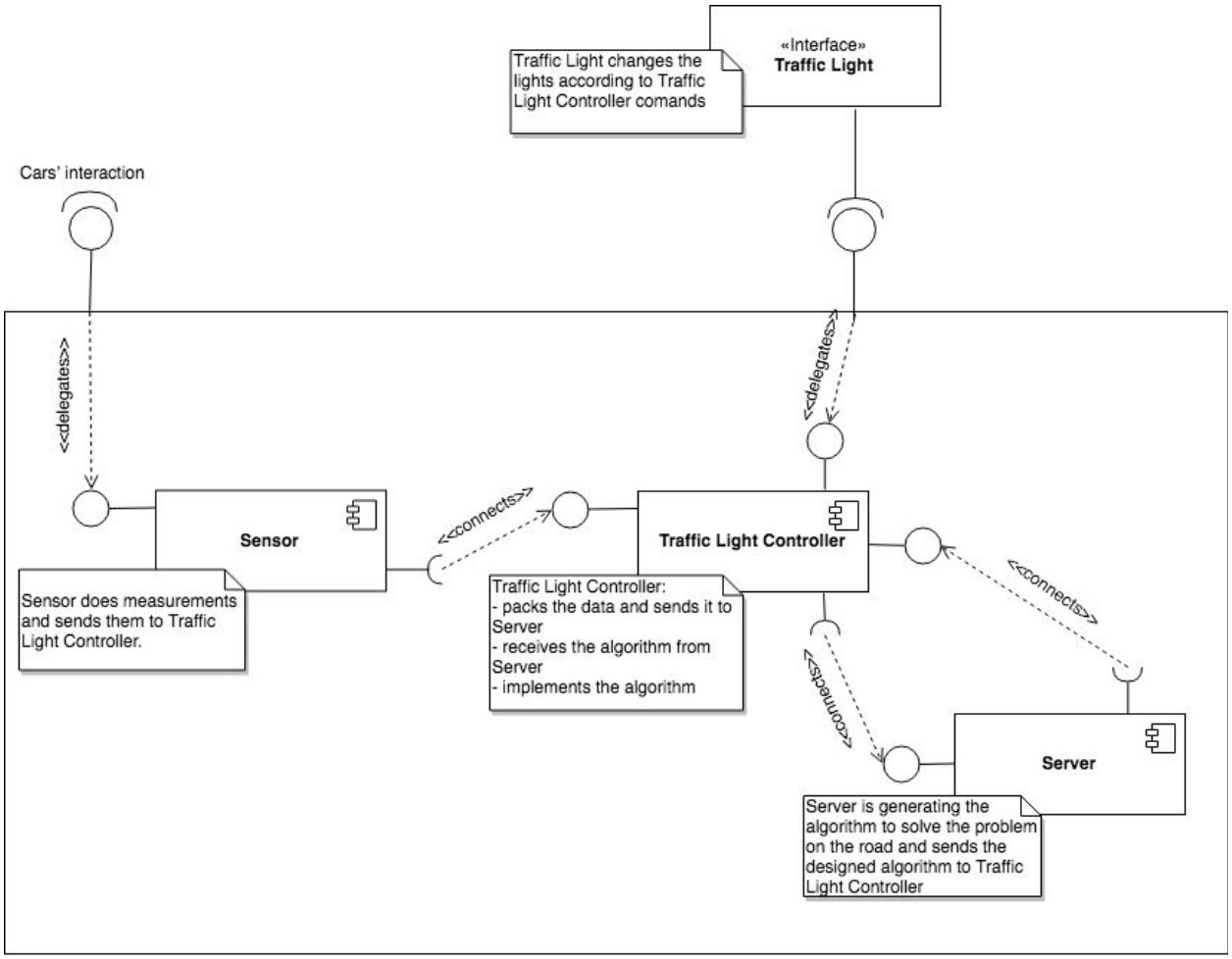

Figure 17

2. The second architecture

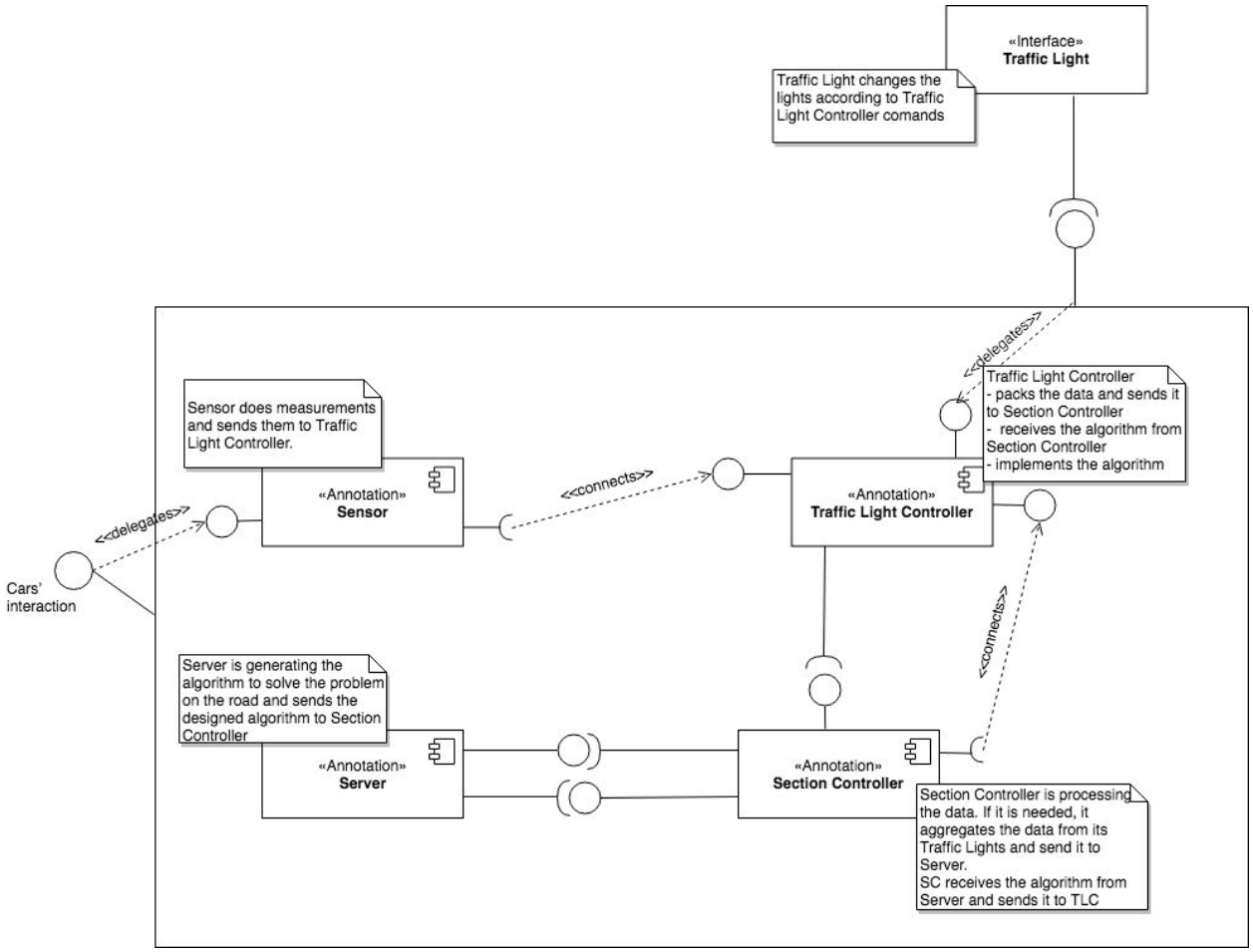

Figure 18

Deployment View

Both architectures have common hardware modules. The only difference is that the second architecture has the additional intermediate element - Section controller. Both architectures has chips hardware connected with sensors. Chips communicating with Controllers via USB as a virtual serial port.

1. The first architecture

In the first architecture Controller communicating with Server via https requests. The biggest disadvantage of this architecture is a scalability. With the growth of the number of the Traffic Lights the server load growths exponentially as a complexity of the task. Software part on the Controller basically switching lights on the Traffic Light and receiving/packing/sending data with measurements.

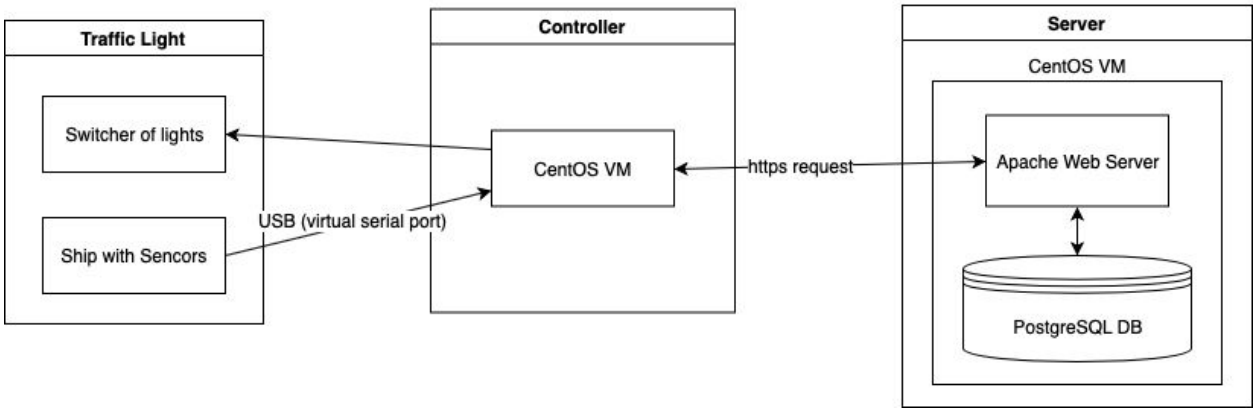

Figure 19

2. The second architecture

In the second architecture we divided all city into sections. For each of them an appropriate Section Controller is assigned, which takes part of the Server functionality and make the server load much lower. The fact that this architecture has the Section Controllers changing the data flow in the system. Controllers (Traffic Light Controller and Section controller) communicating via https requests as well as a Section Controller with Server. In this architecture the Server basically is a generator of the algorithms. This type of communication allows scaling the system in two points which make the whole system more flexible.

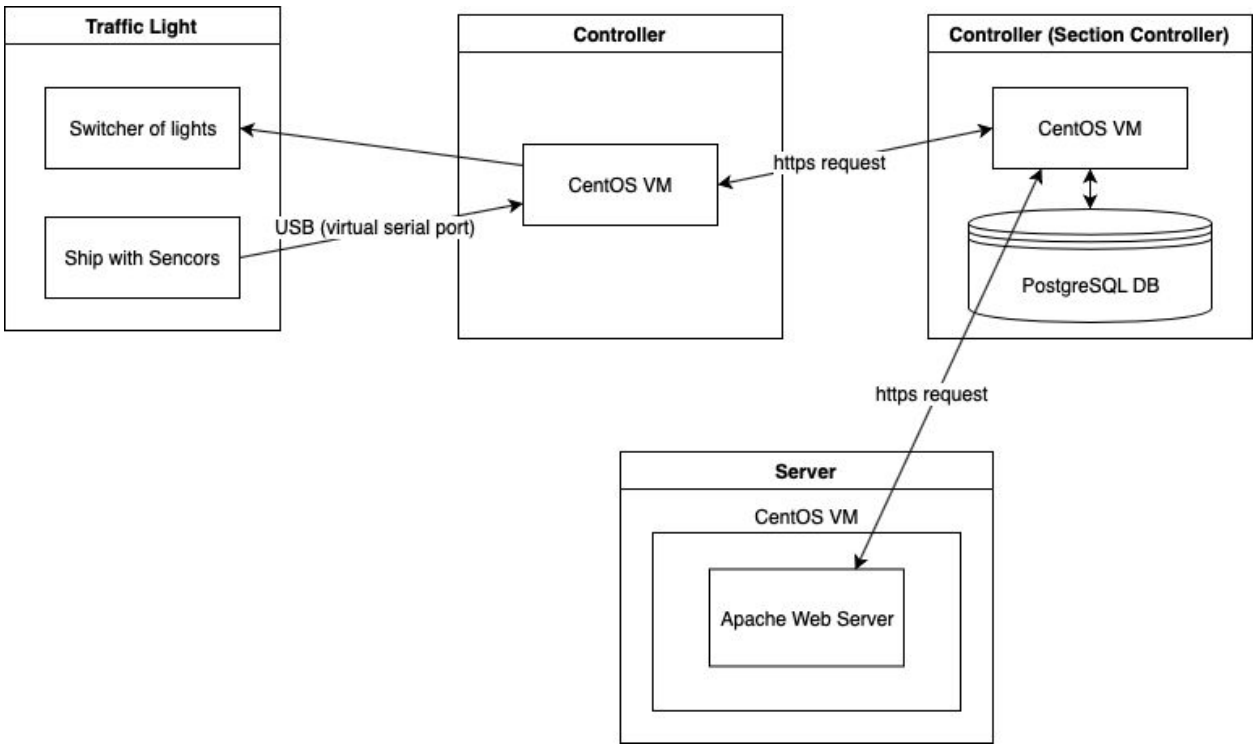

Figure 20

Scenarios

In Smart Traffic Management System cars play role of users. It can be explained in a way that cars' interaction on the road makes the data for the sensors on the Traffic Lights. Further data processing is made on the System depending on its content (whether there is an accident on the road etc). It means that basically there is only 1 scenario for our system.

Scenario for the first architecture:

1. Sensors are doing measurements: temperature, humidity etc every time unit.
2. Sensors send the generated data to Traffic Light Controller.
3. Traffic Light Controller packs the data and sends it to Server.
4. Server is generating the algorithm according to received data
5. Server sends the designed algorithm to Traffic Light Controller.
6. Traffic Light Controller starts to change the lights according to the received algorithm.

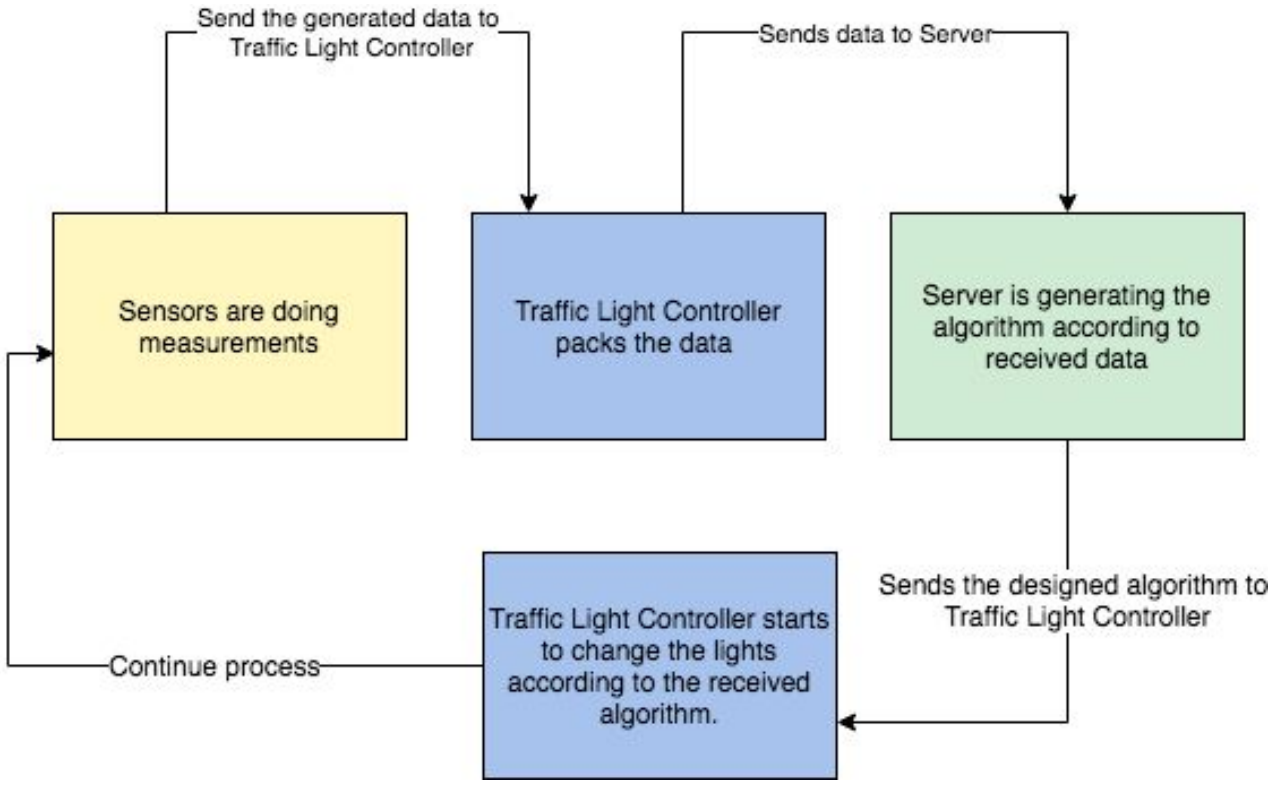

Figure 21

Scenario for the second architecture:

1. Sensors are doing measurements: temperature, humidity etc every time unit.
2. Sensors send the generated data to Traffic Light Controller.
3. Traffic Light Controller packs the data and sends it to Section Controller.
4. Section Controller is processing the data. It has to decide if there is an accident on the road. If there is then Section Controller aggregates the data from its Traffic Lights and send it to Server.
5. If not, nothing happens. Traffic Lights continue to work in usual way.

6. Once the data is received, Server is generating the algorithm to solve the problem on the road.
7. Server sends the designed algorithm to Section Controller.
8. Section Controller is distributing algorithms among its Traffic Lights and sends it.
9. Traffic Light Controller starts to change the lights according to the received algorithm.

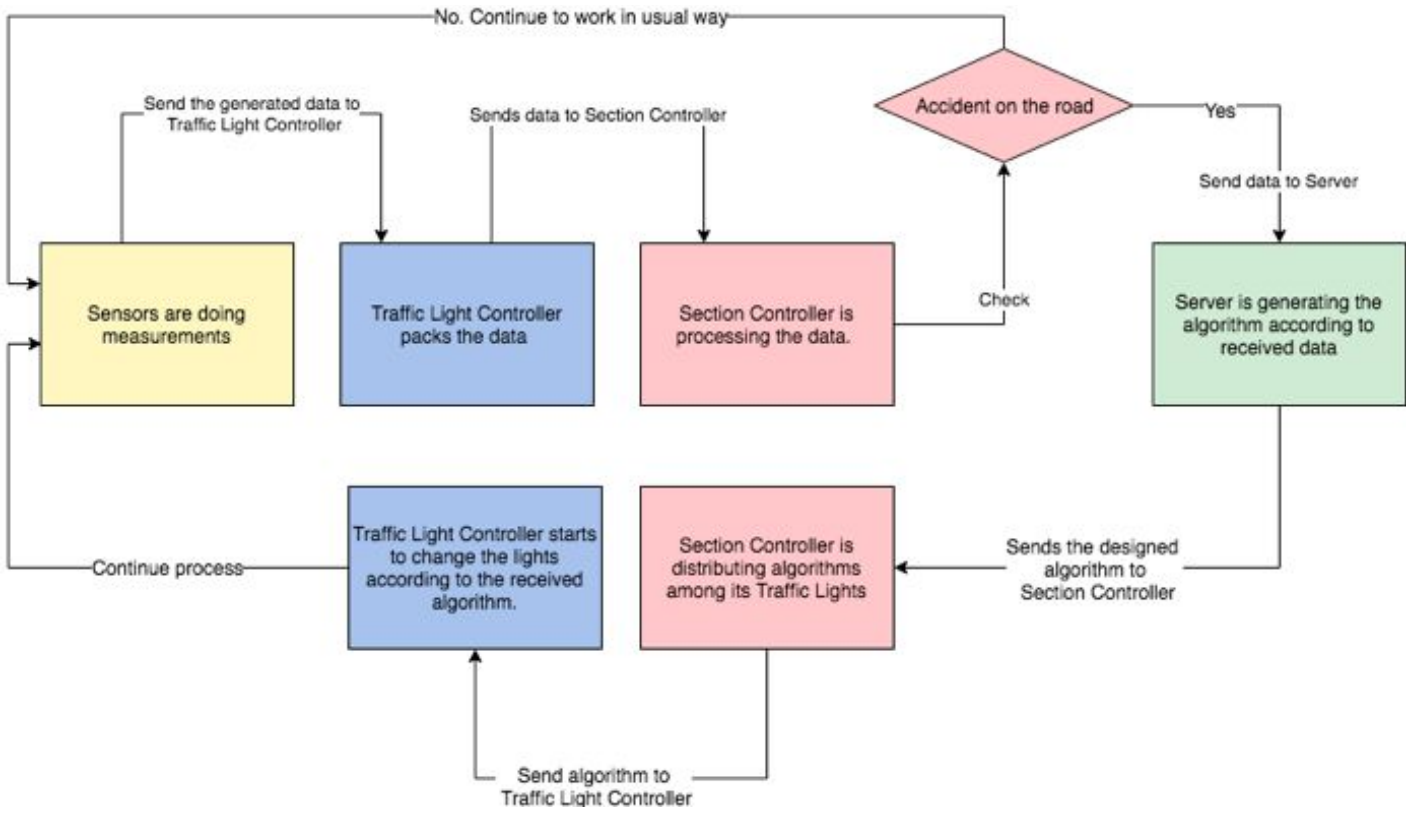

Figure 22

Palladio experiments

1. The first architecture

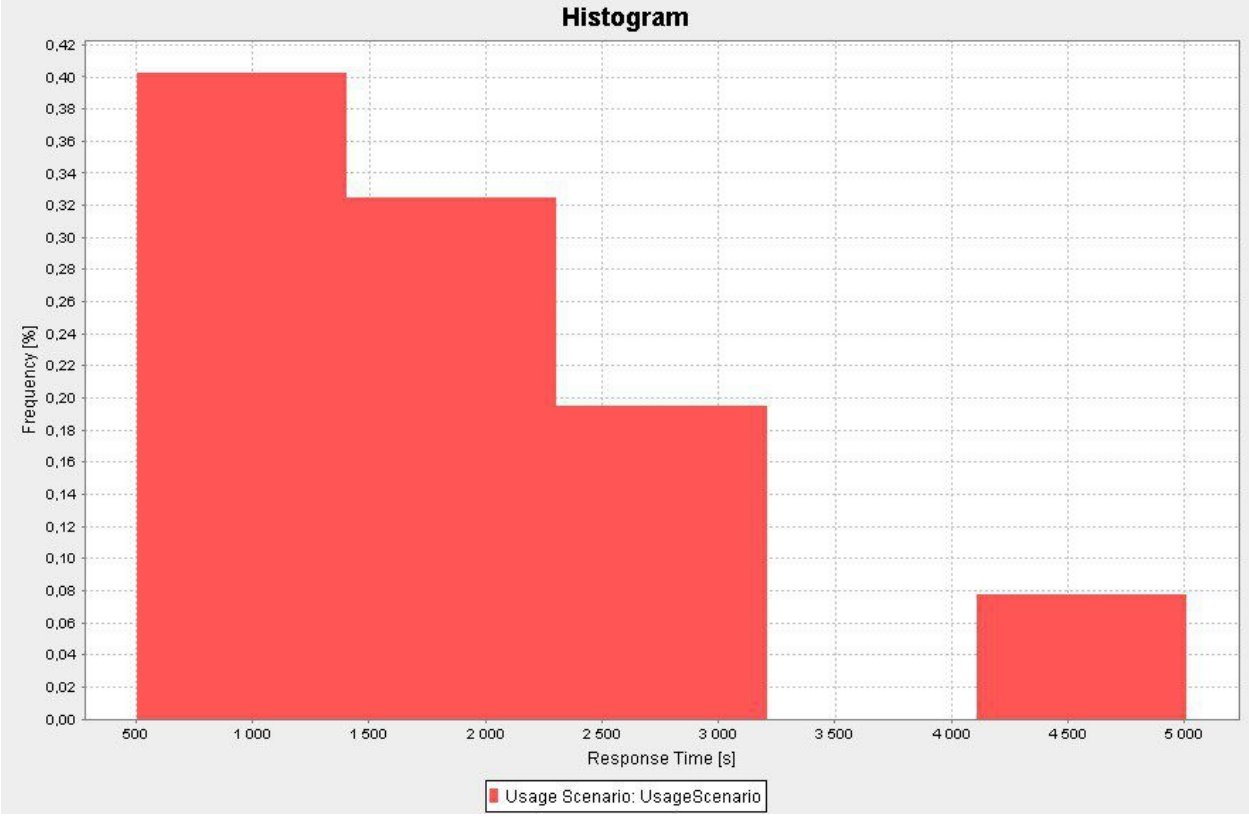

Figure 23

This histogram shows evaluation of UsageScenario of the first architecture. We can see that almost in 10% of all of cases the system takes 4000-5000 s. to process the data. It describes an option then there is a heavy traffic on the roads. At the same time, the system usually takes up to 3000 s. to process the data (it happens in case of usual traffic).

2. The second architecture

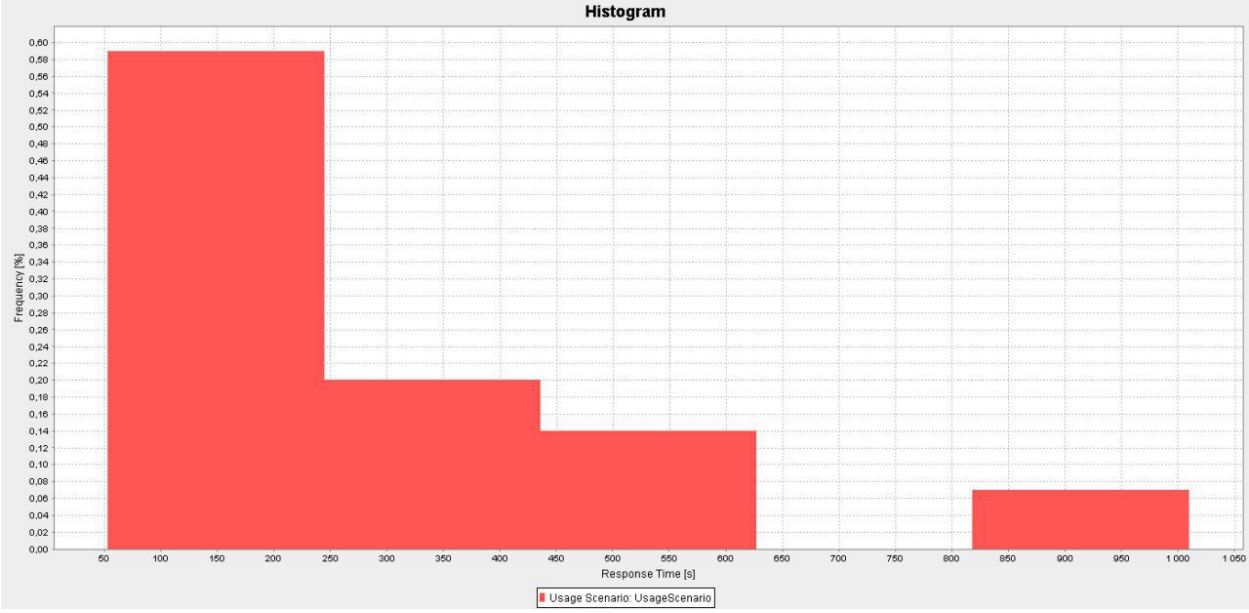

Figure 24

This histogram stands for the evaluation of the second architecture. We can see a way better performance here. So, the system takes 850-1000 s. in the worst case (case of the heavy traffic). In case of usual traffic, it takes up to 650 s. to process the data.

Conclusion

We have created two different architectures for Smart Traffic Management System. It has been explored that the first architecture contains a bottleneck that is Server. The experiments provided by Palladio proved that the second architecture successfully eliminates this bottleneck. It transfers the load from Server to Section Controllers, each of which controls one section on the road (containing several Traffic Lights).