2017 SBESC – Embedded System Competition – WND IoT Challenge

Final Report

Project Title ： iTraffic: Smart Semaphore Network

Students: Levi Barros Nóbrega

Luiz Antônio Bezerra Leite de Queiroz

Matheus Sobreira Farias

Professor: Edna Natividade da Silva Barros

University: UFPE - Universidade Federal de Pernambuco

JEMS ID: 170215

2017 SBESC – Embedded System Competition – WND IoT

Challenge

Declaration of Originality

We hereby declare that this report and the work reported herein was composed and originated entirely by ourselves. Information derived from the published and unpublished work of others has been acknowledged in the text and a list of citations is given in the references section

Professor's Signature: Edna Natividade da Silva Barros
Name (in Block Letters): EDNA NATIVIDADE DA SILVA BARROS

Student's Signature: Matheus Farias
Name (in Block Letters): MATHEUS SOBREIRA FARIAS

Student's Signature: Luiz Antônio Queiroz
Name (in Block Letters): LUIZ ANTONIO BEZERRA L. de QUEIROZ

Student's Signature: Levi Barros Nóbrega
Name (in Block Letters): LEVI BARROS NÓBREGA

Date: 30/10/2017

iTraffic: Smart Semaphore Network

ABSTRACT

Faced with the excessive problems generated by the poor management of infrastructure in the Brazilian cities, the congestion of vehicles in public roads is one of the greatests. An element that is directly related to this issue is traffic signaling, which, has limitations that ends by harming even more the circulation of vehicles in extreme situations, such as: heavy rains, traffic accidents, road works etc. Therefore, observing that current traffic signaling systems can not meet the needs of a road model characterized by a high flow of vehicular traffic, the project stands out because it proposes, in an innovative way, a practical, viable, and accurate proposal to solve such problem.

iTraffic comes up with the main objective of reducing as much as possible the traffic congestion of the main metropolises, acting specifically on the timing of traffic light cycles, adapting them to the road situation in real time, in a dynamic and intelligent way: getting vehicles average speed data from Google Maps API [9] on a traffic light net and detecting the corresponding car flow.

With these numbers, the system makes simulations using a genetic algorithm to find a better timing for traffic lights cycles. Then, the system sends these results to each iTraffic-Device through SIGFOX infrastructure, when the data reaches a device, the microcontroller send these times to the traffic light using NTCIP-Protocol.

These calculated solutions are stored in our database and are constantly analyzed by Google API, ensuring that the macroscopic traffic flow was decreased and making future interventions faster.

.

Key words: traffic jam, genetic algorithm, dynamic, sumo simulator, smart traffic lights, smart cities.

Content

Chapter 1 Introduction

1.1 Motivation and Objectives

With the constant need to have fast and safe ways to locomove, the use of individual automotors vehicles turns to be something recurrent in the life of inhabitants of great metropolis. Used to go to college, to work, and to perform various types of routine activities, it is observed that any hindrance that directly influences this locomotion emerges as a real problem for the actual society. In Brazil, It is not uncommon to observe huge stretches in which the traffic is completely jammed in urban areas, as a consequence of poor infrastructure that accompanies the development of the country. Furthermore, It is not surprising that three Brazilian capitals are at the top of the 10 cities in the world that suffer most from traffic congestion, according to a survey conducted by the company TomTom (Dutch manufacturer of navigation systems for automobiles) [1].

Recent data shows the impact that slow traffic can have on both people's lives and economy, such as the study done by *Agência O Dia* that simulated a "Engarrafômetro" ("traffic jam meter") capable to conclude that the average time lost in traffic jams has tripled in the last 10 years [2], as a research on urban mobility, carried out by IBOPE intelligence, pointing out that paulistanos (people who were born in São Paulo) spend on average 2h38 min in traffic jams to perform their daily activities [3]. Still in São Paulo, according to the survey of TomTom GO GPS navigation software, traffic jam generates losses of up to 80 billion reais (24 billion dollars) per year [4].

From this, and knowing that one of the main reasons for the excessive congestion of public roads is the limitation of the current conventional traffic lights (that consider only the pre-established speed to the routes, so as not to consider variations due to external factors such as climatic changes, accidents or any setback that causes traffic problems), we have the main objective to develop a system of intelligent traffic lights in order to alleviate the problems of the main thoroughfares of cities, which need to maintain a certain flow of vehicles to have a rapid circulation of vehicles.

We can cite the numerous cases of congestion due to the lack of synchronization of traffic lights. As in Av. Antônio Sales (CE), where the reason for the congestion was solely and exclusively due to the lack of synchronization of traffic lights [5]. And in Águas Claras (DF), in which, although there are no public works, abrupt climate change, or even traffic accidents, the flow of cars was greatly weakened, all on account of problems at traffic lights [6].

Nowadays, some systems propose a solution for smart traffic lights that is limited to obtain informations of the speed of cars near traffic signals by cameras installed therein, or by inductive loops capable of acquiring the car speed from the magnetic field generated by its bodywork. In this way, these systems are not so viable, precisely because they have a certain complexity of both installation and maintenance and not taking care of, as in the case of inductive loops, motorcycles, that also transit on public roads. And in both cases, the speed across the lane, considering only the speed of the vehicles in a limited stretch and near the traffic lights. They are, therefore, solutions with low availability and viability.

Thus, to solve the problem of traffic lights from requests and verifications in real time of the speed of traffic in a certain way, It is advantageous to the extent that it can overcome the aforementioned limitations since It deals with any kind of vehicle passing through the road as well as its speed. In addition, the system stands out because It is easy to install, since it only needs a microcontroller connected to the traffic light and with a server in the cloud responsible for

performing all the procedures and send them back. By having access via Google's APIs, regardless of any physical system (which requires installation), of the speed of traffic in the most varied routes, the system is able to do more calculations over time and generate traffic solutions that are more in line with the reality of public roads. Finally, creating a traffic jam prevention system that is more efficient, accurate and easy to maintenance.

1.2 Challenges

- Calculate new timing of traffic light cycles:

The initial and biggest challenge was find a way to calculate timing of traffic lights cycles to reduce the traffic congestion. As traffic can not be represented by mathematical formulas we decided to use a simulator of the real traffic and we discover that SUMO was the best choice, because it is a Open Software and has the TraCI API, a library that allows communicate with the simulator using Python scripts during and before the simulation. After that, we did research and verified that the most used way to calculate better times of traffic lights cycles was using genetic algorithms, since these algorithms have as great quality the resolution of complex problems of optimization. Then we found a genetic algorithm and adapted it so that it could be used in our SUMO simulation environments.

- The use of average speed:

One real challenge that was dealt is that the genetic algorithm do not receive as input the average speed, but the car flow of the road. So, It was implemented an algorithm to detect the vehicles flow from the average speed received by the Google API, using an open source simulator called SUMO Simulator, testing changeable flows until reaches those average speeds to be used in our algorithm. To validate the flow detection algorithm it was requested to CTTU (Companhia de Trânsito e Transporte Urbano, the company that is responsible for managing the traffic of Recife) data of main roads vehicle flows of Recife, and then it was compared to the obtained results from SUMO simulator.

- How to validate the genetic algorithm:

The validation of the genetic algorithm was difficult because it was not possible to apply the results of genetic algorithm directly in the semaphores of the analyzed roads. So it was necessary to get the data of the SUMO simulator and analyze the situation before and after the application of the genetic algorithm, plotting the results and being able to see how effective the algorithm was.

- Communication with traffic lights:

We researched possible ways of communicating with traffic lights and chose to use the compatibility of our system with the NTCIP protocol, being the most used worldwide and the most well documented, in addition to being based on IP protocol.

Chapter 2 System Description

To improve the traffic light system, iTraffic performs a dynamic check of average speeds from queries to geolocation and navigation applications such as: Google Maps and Waze, which have been shown to be effective tools to indicate congestion due to the most diverse factors (rain, road works, agglutination of cars, etc.) and most diverse forms (when using the application the users report voluntarily or involuntarily their speed). These applications provide public APIs (Application Programming Interface) that can be consumed by servers, returning, among various information, the average speed in a part of the runways.

Receiving the information from the APIs, the servers perform calculations to reduce congestion on the traffic lanes by changing the semaphore time duration.

In order to reflect the changes on the traffic lanes, the embedded system installed in the traffic lights would receive from the server the duration in which the traffic lights should be opened, dynamically, according to the current traffic situation.

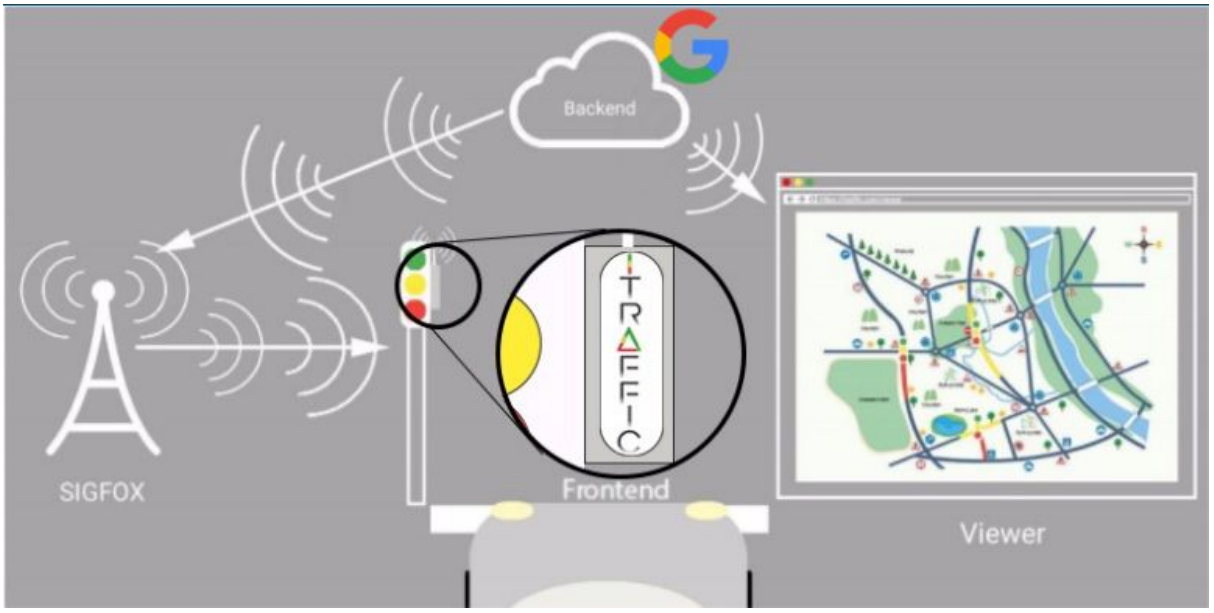


Figure 1: The iTraffic project is subdivided into 3 subsystems: backend, frontend and viewer.

2.1 Modules Description

2.1.1 Backend

In the **backend** subsystem, there is a server divided into sub modules: the query module, the timing estimation module, the communication module, the database module (storage). The query module consume data from Google API, getting all average speeds around each traffic light in a net of traffic lights. Then, the query module checks if already has a previous optimized solution in this net with all same condition: average speeds and timing of traffic lights cycles, if has a solution, it will be sent to the communication module, otherwise, the timing estimation module will be call. This module detects the corresponding flow for each line giving all average speeds, applying a divide and conquer algorithm to test all possible flow numbers at SUMO simulator, using always the actual

times from each traffic light and resetting the environment at each test. After detected, this module will run the genetic algorithm that gives a better timing for each traffic light cycle, It is important to fix that the algorithm gives a rate between 0 to 10 based on a variable called saturation, which defines how good the traffic jam is. Finally, these times are send by the communication module to the respective iTraffic-Device, placed at each traffic light, and the result will be stored at the database module. Besides that, SIGFOX communication module is responsible for receive the current cycle times from each iTraffic-Device and store at the database module and send data to iTraffic Viewer module.

2.1.2 Frontend

In the **frontend** or the iTraffic-Device, we have three submodules: the sniffer module, the actuator module and the SIGFOX communication module. The first one constantly listens to the timings sent by the Transit Center, originally sent to the traffic light using a protocol, and sends them to the actuator module. The SIGFOX communication module constantly send uplink messages asking to backend for new timings, when receives, send it to actuator module. The actuator module will communicate with the traffic light sending these times acquired through SIGFOX when available and when not, will send the time received from sniffer module.

2.1.3 Viewer

In the **viewer** we have a website that communicate with iTraffic backend through an API and receive current iTraffic-devices status, like: speed (km/h), duration of green light and red light (seconds), and if It is working. Any changes at an iTraffic-device is reported to viewer, through backend. This data is shown on a map and notifications of problems with traffic lights are received in real time.

Chapter 3 System Implementation

3.1 Backend Subsystem

The backend module was implemented using an Amazon Web Server (AWS) with a virtual private server (VPS) EC2 computer running Ubuntu 16.04. At the EC2 It was installed SUMO Simulator version 0.31.0 and Python 2 to run ours scripts for flow detection and to find the optimized time for a net of traffic lights.

The main design code, i.e the genetic algorithm, was based on Wellington Cruz's graduation work entitled Aplicação de Algoritmos Genéticos em Semáforos Inteligentes [7]. The algorithm was made in the programming language Python 2 and as it is a simulation algorithm, it makes a direct communication with SUMO Simulator through an API called TraCI. Although it is perfect for the idea that was proposed by the author, it is not perfect for iTraffic, of which it was necessary to make several alterations.

One of them and very relevant is that the algorithm is based on the flow of vehicles that is present in the simulation environment, while the Google API sends the average speed of cars at the required time. Thus, it was necessary to implement an algorithm, called as Flow Detection Algorithm, which receives as input the average speed of the cars and returns the flow of vehicles such that, with the standard semaphore timing of the environment (in a real case it would be the current static timing), would return the same average speed. The Flow Detection Algorithm uses the SUMO Simulator to, with divide logic to conquer, test flow values, until it converges to find an average speed value, the convergence value has been chosen such that it is within a range of 0,7-1.3 of the average real speed, by mere convention and error analysis.

Then, the genetic algorithm code was made for a standard environment, as iTraffic sought to be more general than a simple case, it was necessary to apply the algorithm to several environments, for this, SUMO Simulator provides the possibility of creating an environment from the outset, but also provides, along with a web application, Open Street Map (as shown in Figure 2 below), a user-selected base environment, but, as recommended, the environment is not very true to reality, so we needed to adapt to make it more faithful.

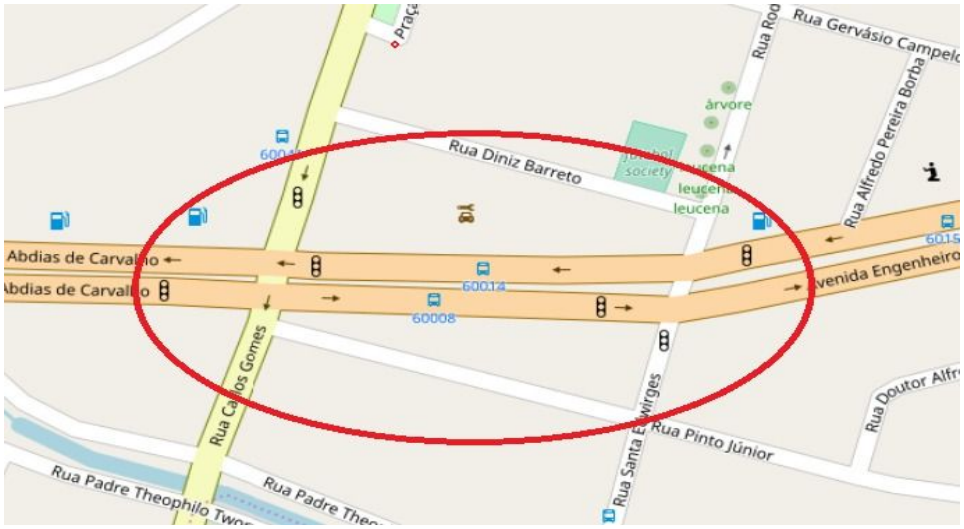
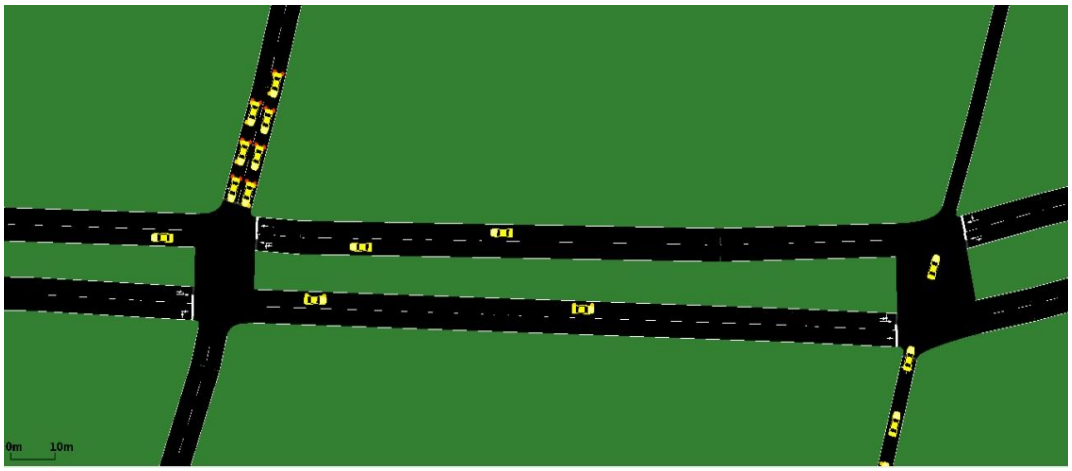


Figure 2: Example of a base environment (Open Street Map)

In Figure 3 below it is possible to see the same environment selected in Figure 2 after the conversion to the SUMO simulator.

Figure 3: Example of the same base environment in SUMO



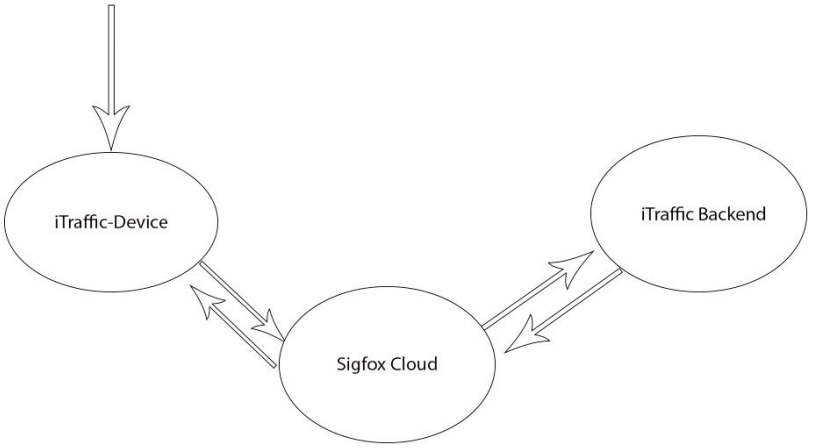
The Google API used, called Google Maps API Get Distance, given an input with its geographic coordinates (latitude and longitude), provides a block with data: distance between the two points (in km), time taken to cross the two points without dealing with the traffic (in seconds), time it takes to cross the two points, now considering the traffic (in seconds) and finally, the "speed of the traffic jam", that is, the average speed of the vehicles that are between the two points (in km/h), as shown in Figure 4.


```
{
    "destination_addresses": [
        "Av. Caxangá, 1366 - Madalena, Recife - PE, Brasil"
    ],
    "origin_addresses": [
        "Av. Prof. Estevão Francisco da Costa, 1 - Torre, Recife - PE, 50711-200, Brasil"
    ],
    "rows": [
        {
            "elements": [
                {
                    "distance": {
                        "text": "0,8 km",
                        "value": 837
                    },
                    "duration": {
                        "text": "2 minutos",
                        "value": 146
                    },
                    "duration_in_traffic": {
                        "text": "3 minutos",
                        "value": 191
                    },
                    "speed_in_traffic": {
                        "text": "16 km/h",
                        "value": 15.78
                    },
                    "status": "OK"
                }
            ]
        }
    }
```


Data

Figure 4: Example of an API shown data

The communication between the server, which now has the data of a possible better timing, and the iTraffic-Device, is carried out through the SIGFOX network, where it can be schematized in the following diagram:



1. In the first step, each iTraffic-Device sends an uplink message with a semaphore timing update request. In the uplink message, the device sends 2 bytes indicating the green timing, and 2 bytes for the red timing, it was imposed that a semaphore would not be implemented with a timing above 255 seconds, which is quite reasonable, so it was defined 2 bytes for each timing (remembering that the communication is done in hexadecimal by default of the SIGFOX network).

2. In the second step, the message is received by at least one base station and is transmitted to the SIGFOX Cloud.

3. In the third step, the message is processed and the callback defined is called, after that, the cloud waits for the callback response with the timing update message.

4. In the fourth step, the backend subsystem receives the callback and returns a response with the date being sent to the device.

5. In the fifth, the last step, the SIGFOX Cloud transmits the message to the base-station that will send the timing update message to the device, in the established protocol for this communication, it has 2 bytes for the green timing, 2 bytes for the red timing, 2 bytes for the hours and 2 bytes for the minute that such timing will be put into practice (the choice of the 2-byte size was made analogously to step 1).

Finally, the iTraffic backend storage module, as its name says, has utility to store relevant data to create a history. This module is subdivided into an iTraffic database in the tables: trafficlights, nets and solutions.

1. In the nets table, the signal networks are shown, each network is identified by an id and name, also having network creation and deletion information.

2. In the traffic lights table, the semaphores are defined. In the name column there is the semaphore code established by the CTTU, in net_id there is the id of the semaphore network to which the semaphore is present, in the columns lat and lng it has the geographical coordinates, respectively latitude and longitude, at reference_range, it has the input that is given to the Google API to calculate the average speed, at reference_speed, it has the average speed of the cars along the reference_speed at the given time, in m/s, in sigfoxID we have the id of the SIGFOX board, in the temp column, there are two values, the first is the current timing of green, the second is the current timing of red and the last column informs whether the semaphore is active or not.

3. In the solutions table, it is find old solutions, with the intention of, for example, repeat previous situations without having to re-calculate the values. There is, at cod, average speeds, in net_id there is which semaphoric network was realized such a calculation of solution, in the temp column, the timing for the respective average speeds of the cod column, and in the calc_at column, we have the date at which such solutions were calculated.

3.2 Frontend Subsystem

In the frontend subsystem, we have the sniffer, communication and actuator modules. For the implementation of the sniffer module, an Arduino Uno with an ethernet shield coupled was used, the main function of the sniffer module is to be listening to the information sent to the traffic light by the Transit Center and sends to the actuator module this information through communication serial.

Figure 5: Frontend iTraffic-Device Prototype

Figure 6: Frontend iTraffic-Device Connections

The communication in the frontend subsystem aims to receive the timing changes proposed by the backend and sent through the SIGFOX system, and was implemented using the SIGFOX development kit with the ARM microcontroller and shield antenna.
The actuator aims to communicate with the traffic light through the NTCIP communication protocol by sending the timing received from the communication module or by sending the timing received by the sniffer module, and was implemented using a mega arduino with an ethernet shield.

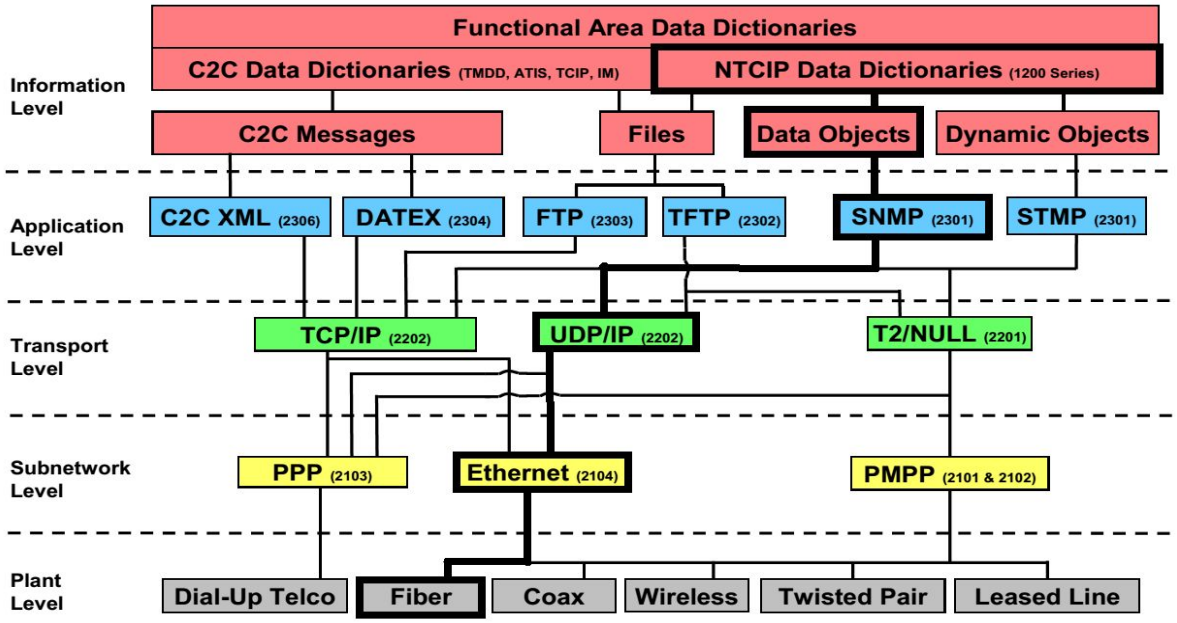


Figure 7: NTCIP - Protocol

The communication protocol used in the project was NTCIP, as it is one of the standards required by law for traffic lights, and also the most adopted in the world. For our system we used: ethernet, UDP / IP and SNMP for communication, with the sniffer module as client and the actuator module as server.

3.3 Viewer subsystem

To implement the web application, it was used: javascript, CSS, HTML and NodeJS with socket.io for server-client communication. In addition, to draw the maps and place the traffic lights was used LeafletJS[8] with the maps of the Open Street Map.

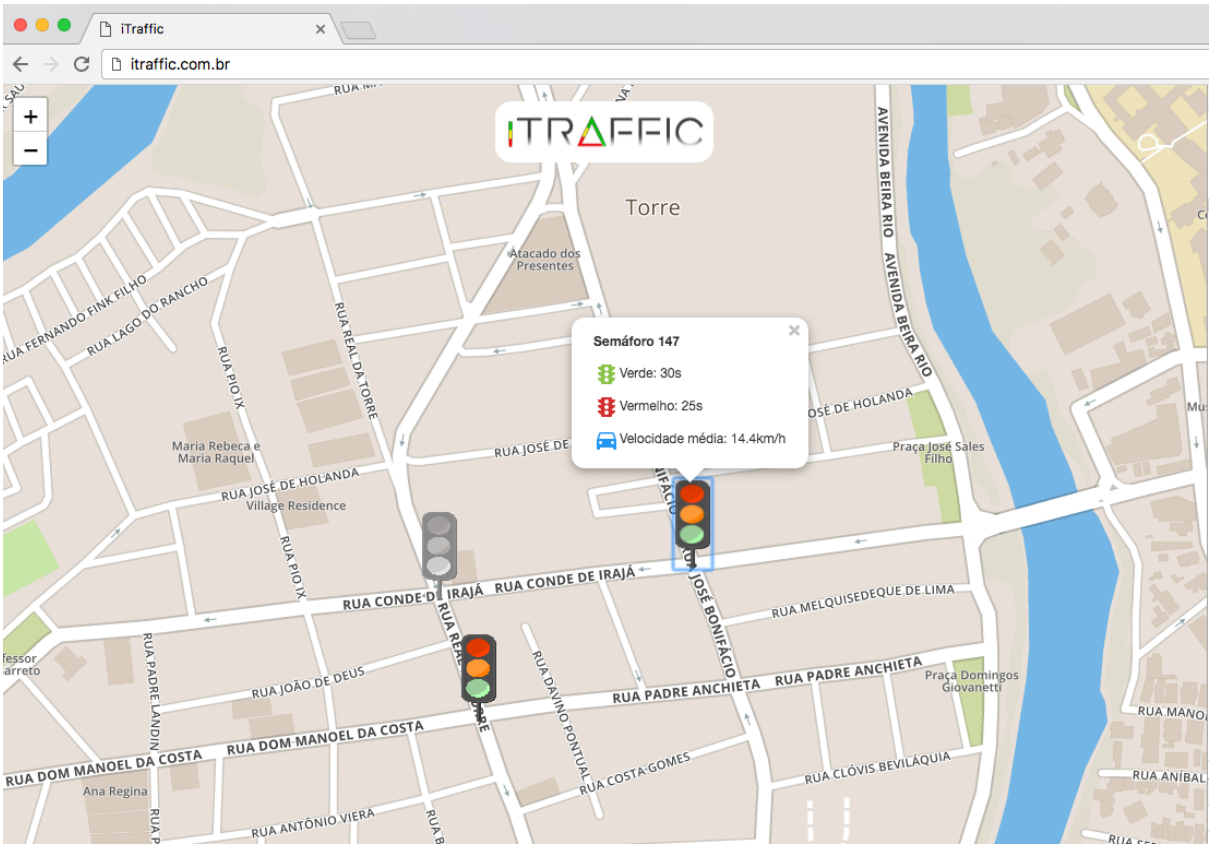


Figure 8: iTraffic Viewer, grey traffic lights means device error.

Chapter 4 Results

4.1 Flow Detection Algorithm

In order to validate if the algorithm made for flow detection works correctly, a comparative analysis of data provided by CTTU was made, in this way it was possible to obtain Tables 1 and 2 where the accuracy of the algorithm and its fidelity to reality in four periods of Test can be observed.

In the first column, there is the time period analyzed in the day, in the second column, the average speed in the analyzed section of Av. Abdias de Carvalho, composing the simulation environment illustrated in figure 3, with Table 1 for the street in one direction, and Table 2 for the street in the opposite direction. In the third column, there is the flow of vehicles obtained by the data provided by CTTU, in the fourth column the flow of vehicles obtained by the flow detection algorithm, both related to a period of 15 minutes. And in the fifth column the percentage relative error, calculated through the expression below:

$$RelativeError = 100 * \frac{|Reference - Analyzed|}{Reference}$$

Where the reference value was taken as the value of the vehicle flow given by CTTU, and the value analyzed is the vehicle flow obtained by the flow detection algorithm.

Table 1: First example for analysis

Period of Time	Average speed (km/h)	Vehicles Flow (CTTU)	Vehicles Flow (iTraffic)	Percent Relative Error
9-12 h	18	356	375	5,34%
12-14 h	24	267	300	12,36%
14-17 h	20	356	300	15,73%
17-20 h	14	534	600	12,36%

Table 2: Second example for analysis

Period of Time	average Speed (km/h)	Vehicles Flow (CTTU)	Vehicles Flow (iTraffic)	Percent Relative Error
9-12 h	23	251	275	9,56%
12-14 h	25	188	225	19,68%
14-17 h	27	251	300	19,52%
17-20 h	24	376	425	13,03%

4.2 Genetic Algorithm

In order to validate if the genetic algorithm works correctly, a different simulation environment was used to cover a more complex situation than that found in Av. Abdias de Carvalho, with 6 car entrance ways. Shown below.

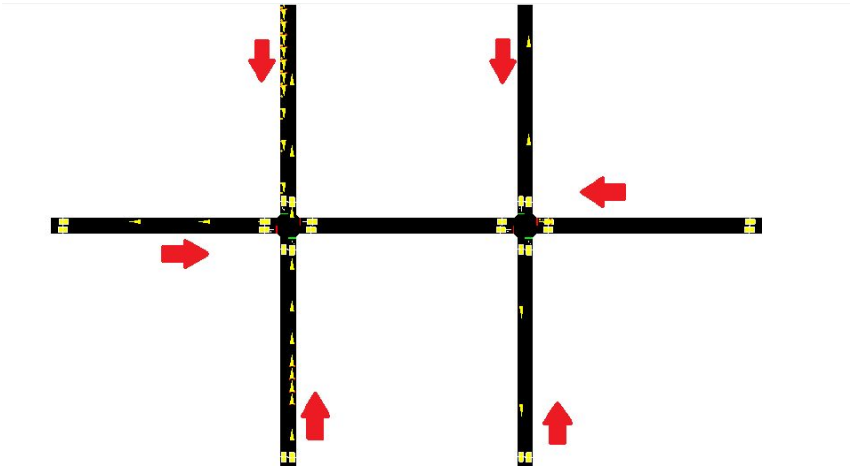


Figure 9: Simulation environment used for the validation of the genetic algorithm.

In order to validate if the genetic algorithm works correctly, graphs have been made to analyze the average speed of the vehicles that pass through the section fixed before the traffic lights after and before the performance of the algorithm. For the comparative purposes of the two situations we also use the same simulation duration: 700 seconds and the same flow obtained by the flow detection algorithm. The graph below shows the average speed in meters per second during the simulation time in the six entrance roads simulation before the calculation performed by the genetic algorithm.

As it can be seen below, before the calculation of the new timing, the average speed of the 6 flows were very irregular and a congestion was formed during the simulation.

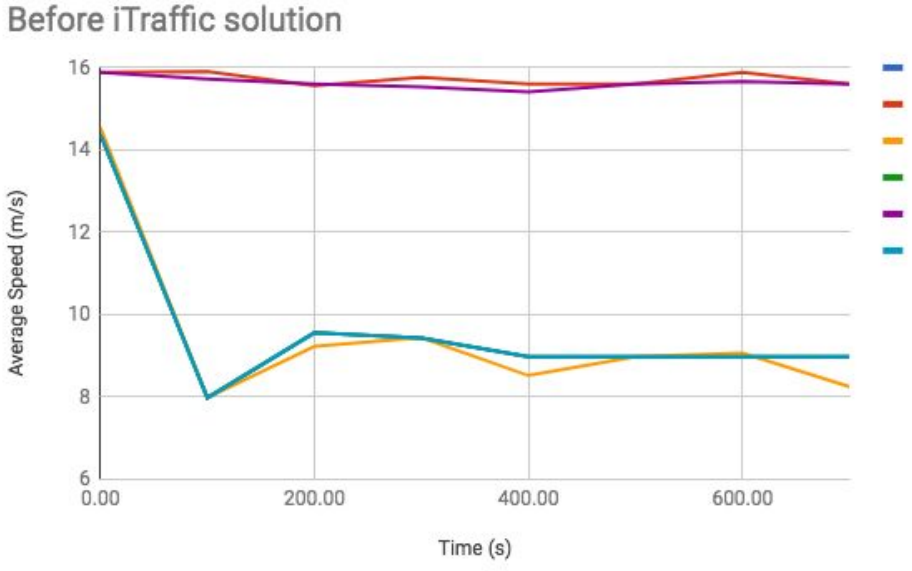


Figure 10: Average Speed at each flow before iTraffic Solution

After calculating the timing by the genetic algorithm, it can be noticed that the average speeds remained higher and more regular than in the previous environment, reflecting the improvements achieved by the system. In addition, we can see in the table below that more vehicles were able to cross the traffic light in the same simulation time interval and the average total speed during the simulation for the six lanes increased.

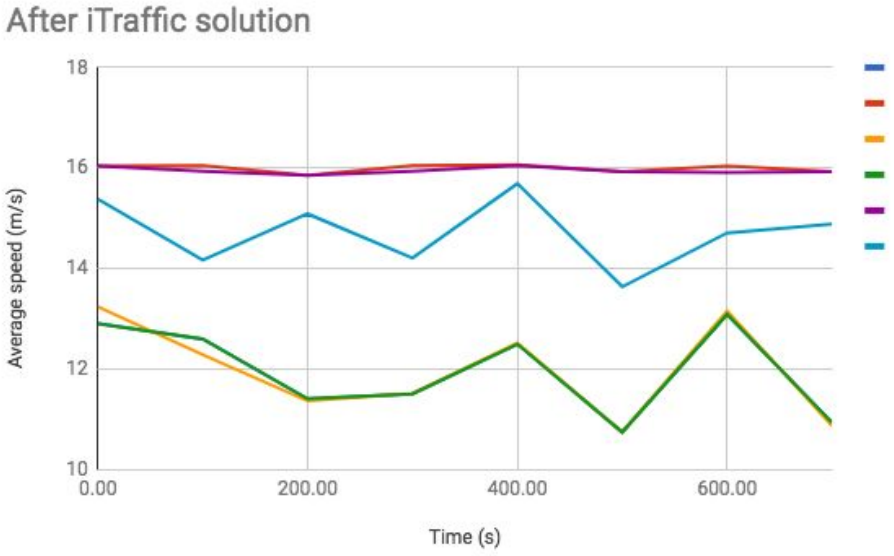


Figure 11: Average Speed at each flow after iTraffic Solution

Table 3: Average Speed and Vehicles Count results.

//	Average Speed	Vehicles Count
Before iTraffic Solution	5,53 m/s	689 vehicles
After iTraffic Solution	12,70 m/s	1106 vehicles

Chapter 5 Conclusion

The main objective of iTraffic is to dynamically reduce congestion by modifying the timing of traffic lights using as basis for calculations the data obtained by the Google API and consists of three subsystems that communicate through SIGFOX's communication infrastructure.

The iTraffic proved to be an excellent solution to the complex problem that is the traffic congestion, potentialized by the desynchronization of the traffic lights, since it was efficient, with answers in acceptable time, besides not needing external sensors and for being an intelligent system which learns over time, allowing better responses and in less time. To achieve this goal, the iTraffic-Device is installed at each traffic light of a traffic light network, and this device sends the current traffic timings to the iTraffic Backend. Backend constantly analyzes the average speeds of the traffic lights network and calculates the best solution through the genetic algorithm in conjunction with the SUMO simulator, sending this best solution to each iTraffic-Device associated with the network through the SIGFOX network. In the meantime iTraffic-Viewer allows you to view all changes and disturbances in the iTraffic network, being notified if any iTraffic-Device stops working.

References

[1] Estadão. (2016). *Três cidades do Brasil estão no top 10 de congestionamentos - Brasil - Estadão*. [online] Available at: http://brasil.estadao.com.br/noticias/geral,tres-cidades-do-brasil-estao-no-top-10-de-congestionamentos,10000022561 [Accessed 29 Oct. 2017].

[2] O Dia. (2013). *Tempo perdido em engarrafamentos triplica nos últimos 10 anos - O Dia*. [online] Available at: http://odia.ig.com.br/portal/rio/tempo-perdido-em-engarrafamentos-triplica-nos-%C3%BAltimos-10-anos-1.536523 [Accessed 29 Oct. 2017].

[3] Anon, (2017). *Paulistanos gastam em media 2h38min no transito para realizar suas atividades diarias*. [online] Available at: http://www.ibope.com.br/pt-br/noticias/Paginas/Paulistanos-gastam-em-media-2h38min-no-transito-para-realizar-suas-atividadesdiarias.aspx [Accessed 29 Oct. 2017].

[4] Ecommercebrasil.com.br. (2015). *Trânsito de São Paulo gera prejuízo de R$ 80 bilhões ao ano*. [online] Available at: https://www.ecommercebrasil.com.br/artigos/transito-de-sao-paulo-gera-prejuizo-de-r-80-bilhoes-ao-ano/ [Accessed 29 Oct. 2017].

[5] Blog do Eliomar. (2017). *Semáforos dessincronizados causam engarrafamento na avenida Antonio Sales*. [online] Available at: http://blogdoeliomar.com.br/semaforos-dessincronizados-causa-engarrafamento-na-avenida-antonio-sales/ [Accessed 29 Oct. 2017].

[6] Portal Alô. (2016). *Detran desvenda mistério de engarrafamento de três horas em Águas Claras | Portal Alô*. [online] Available at: http://www.alo.com.br/noticias/detran-desvenda-misterio-de-engarrafamento-de-tres-horas-em-aguas-claras-379025 [Accessed 29 Oct. 2017].

[7] Cruz, W. (2011). Aplicação de algoritmos genéticos em semáforos inteligentes. São Paulo: UNIVERSIDADE PRESBITERIANA MACKENZIE.

[8] Leaflet. (2017). *Leaflet — an open-source JavaScript library for interactive maps*. [online] Available at: http://leafletjs.com/ [Accessed 30 Oct. 2017].

Google Developers. (n.d.). *Google Maps Distance Matrix API | Google Developers*. [online] Available at: https://developers.google.com/maps/documentation/distance-matrix/?hl=pt-br [Accessed 1 Jul. 2017].