A New Solution to the Traffic Managing System for Autonomous Vehicles

(Demonstration)

Rodrigo Rodrigues Novaes Júnior[1], Daniel de Sousa Santos[1],
Gabriel Martins Franco Santiago[1], Sandro Renato Dias[1]
[1]Computing Department, CEFET-MG, Belo Horizonte, Brazil
{rodrigo.novaes.jr, daniel.sousasantos01, gasantiago2012}@gmail.com, sandro@decom.cefetmg.br

ABSTRACT

One of the most famous IOT applications, the vehicular ad hoc networks, VANETs, are emerging new possibilities for a more efficient traffic system. This work intends to simulate a VANET using San Andreas Multiplayer platform as a simulation environment, replacing the usual traffic lights for automatic centrals, aiming to reduce the time spent on a travel and cease with unnecessary stops. The centrals were distributed throughout the streets intersections on the virtual city of Blueberry. Using UDP sockets, we adapted the vehicles to exchange messages with the centrals. Therefore, the central manages the crossing of each vehicle on its respective intersection, assuring efficiency and safety. The results demonstrate a gain with this approach to avoid traffic jams and unnecessary stops in front of traffic lights.

Keywords

Wireless communication, autonomous vehicle, internet of things, vanet

1. INTRODUCTION

As the recent advances in the IOT keep increasing, it's important to think in better solutions for daily problems. The current managing system in streets intersections is based on traffic lights, which allows the passage through a determined route, while blocking the others that might interfere in the first one. However, this solution can cause traffic jams and unnecessary stops if a route is blocked without any need.

With the brand new technologies for autonomous vehicles, it's possible to use telecommunication as an improvement to the traffic. As a result from the application of these technologies, the vehicular ad hoc networks, VANETs, consist in a set of routing protocols among vehicles, which can be applied to increase the comtemporary traffic efficiency [6].

There are several studies emerging on this area, such as Google Self-Driving Car Project, which is already found on streets along USA [5], and MIT's revision on streets intersections using slot-based systems, that challenges the traffic control using traffic lights [8]. Besides, Dubai's government has a goal that, until 2030, 30% of their local fleet would be composed by smart vehicles. This proposal substantiate the benefits a smart fleet provides, such as reduction in number of accidents, less emission of pollutants and the increasing in the traffic efficiency [1].

This work used San Andreas Multiplayer [7], a platform for editing features in the game Grand Theft Auto: San Andreas, to implement a communication protocol between autonomous vehicles (here are the agents) and centrals in streets intersections, aiming to cease with unnecessary stops in front of traffic lights. The algorithms developed for this work presented good results, as we could reduce the total waiting time for a vehicle from its source point to its destination. The demonstration video of this project can be found at `https://youtu.be/aRO8viGEzks`.

Appears in: *Proc. of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2017), S. Das, E. Durfee, K. Larson, M. Winikoff (eds.), May 8–12, 2017, São Paulo, Brazil.*

2. OUR SYSTEM

San Andreas Multiplayer is a platform, developed by game fans, as a modification for the game Grand Theft Auto: San Andreas. It permits the creation of scripts to control elements of the game, such as objects, characters, temporal events and so on [7]. The language adopted is PAWN, a simple, robust and typeless language, with a C-like syntax [2]. Besides, it's possible to develop plug-ins in other programming languages, such as C++ and Java, allowing a larger number of dynamic applications to run with the modifications, and to make use of the basic support to database management systems such as SQLite [7].

2.1 The simulation environment

The platform features a virtual city, named Blueberry, where the simulations took place. We created a structure, called *node*, composed by an identifier, a spacial position, with coordinates x, y and z, and four references, representing which nodes are reachable from the current one. There are 1,277 nodes distributed throughout Blueberry, used to create a routes table and the car courses along the nodes. However, one of the difficulties faced in the project was the assignment of each node's references. Since it's a manual process, it turned hard the mapping of other cities in the game. We used the platform's native support to SQLite for storing the nodes information [3].

2.2 Creating routes

Graphs are common models for problems involving routes. Dijkstra's algorithm for shortest path calculates routes from a vertex to every other vertex of a connected graph. If there are costs assigned to each pair of vertexes, then the routes calculated by the algorithm will have minimal costs [4]. We implemented a plug-in for the platform which uses the stored nodes information in order to create a directed graph, which serves as an input of Dijkstra's algorithm for

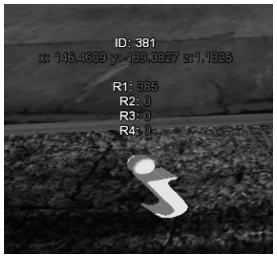

Figure 1: The representation of node 381 in the simulation environment, with its positions and references.

Input: s: node, G: set<node>
Output: L: list<node>
Data: c(node u, node v): function, d: list<node>, A: list<node>
begin
  forall the $u \in G$ **do**
    $d.update(u, \infty)$, $L.update(u, \textbf{null})$, $A.insert(u)$
  end
  $d.update(s, 0)$, $A.make\_heap()$
  while $|A| \geq 1$ **do**
    $u \leftarrow A.pop()$
    forall the *reference* $v \in u$ **do**
      if $d.find(v) > d.find(u) + c(u, v)$ **then**
        $L.update(v, u)$, $d.update(v, d.find(u) + c(u, v))$
        $A.make\_heap()$
      end
    end
  end
  return L
end

Figure 2: Dijkstra's algorithm for shortest path adapted to determinate a best route along the nodes. We have, as input, the node s for departure and a set G of all nodes in the environment. As output, we have a list L which stores the direct ancestor of a given node. The function $c : G \times G \rightarrow \mathbb{R}$ is the Euclidean distance between two nodes, considering their position in the space. d is a list which stores the distance to a node starting from s. A is a heap with minimal priority based in d.

shortest path. The output provides a best route for this vehicle, allowing us to simulate the autonomous vehicles behavior [3]. See Figure 2 for the Dijkstra's algorithm adapted to determinate a best route along the nodes.

2.3 The virtual central

We created virtual centrals, distributed along all sorts of streets intersections throughout Blueberry, that behave as an automatic traffic management device, which control each vehicle's speed in order to prevent collisions. We developed a plug-in in C++, containing a queue that stores the current vehicles accessing an intersection. A central has a signal reach of radius r_1, and a critical region, susceptive to collisions, of radius r_2. Henceforth, R_1 and R_2 will represent the regions with radius r_1 and r_2, respectively. The plug-in is executed via a PAWN script, where an event is triggered by a vehicle accessing R_1, storing it into the queue. As it reaches R_2, the vehicle is popped out from the queue, then we make a regular verification of collisions among all other vehicles at the intersection. If a collision is detected, the speeds of these vehicles are delayed, so the first one can cross towards its destination. We implemented all centrals for each intersection in the city map. The difficulty of collision detection increased for a more complex shape of intersection [3].

Besides, we created a simpler algorithm to prevent collisions beyond the intersections. Since every vehicle has a traffic way and a spacial position, then the same idea was used. If A and B are two vehicles in the same lane of the street, and A is just behind B, coursing through the same traffic way, then we have transparent spherical areas R_A and R_B around A and B, respectively. If region R_A intersects R_B, then A's speed is reduced in order to guarantee a safe breaking distance. This should be used if, and only if, A and B are in the same traffic way and in the same direction [3].

The communication between vehicles and centrals used a message exchange system via UDP sockets. All vehicles in the simulation had their network layer address set to the local interface of the machine running the simulation. The application port of each side of the connection was set as the object identifier, guaranteeing an unique port for each vehicle [3].

3. INITIAL RESULTS AND CONCLUSION

In a 15 minutes simulation with 45 vehicles, we calculated the total waiting time, defined as the time a vehicle spent in the reach of a central. We simulated the behavior of traffic lights using the centrals to block the access of a route for a period of 8 seconds. We ran the simulation using both approaches to compare their performances. The graph in Figure 3 shows the total waiting time calculated from the same simulation using these approaches. Therefore, the first one, using centrals, had better results.

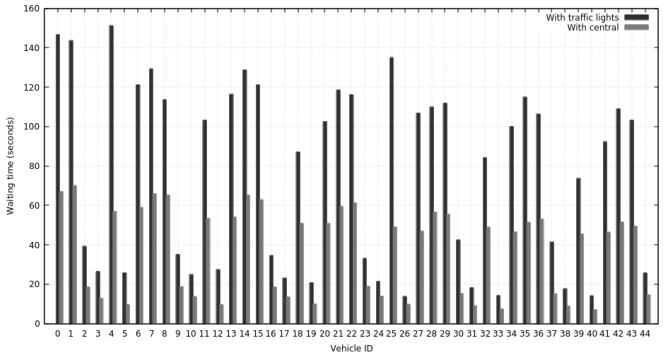

Figure 3: Results comparison between the approach using the automatic centrals (gray bars) and traffic lights (black bars).

We had several benefits in using San Andreas Multiplayer as a simulation environment. For example, we didn't need to implement any vehicles functions or traffic objects, besides the ones we propose to create. Also, we had a three-dimensional environment, which provides a richer amount of details. However, there are some differences we need to indicate: i) the spatial dimensions of Blueberry, compared to a real-life scale, makes 45 vehicles be a massive traffic in the city; ii) we assumed that all vehicles in the city are autonomous. Therefore, we limited the scope of problems that might interfere in the simulations.

Moreover, this opens a row of possibilities using this as a simulation platform. For example, we could use priority lists instead of queues, assuring that some vehicles should have prioritary access to their destination streets. Also, we could introduce pedestrians in the simulation, creating a set of more complex protocols that consider accidents in the environment and the interruption of the street traffic flow with a pedestrian crossing abruptly.

Acknowledgments: Our thanks to CEFET-MG for the support to develop and present this research.

REFERENCES

[1] B. Brown. Dubai sets a high goal for driveless cars. *Digital Trends*, Apr. 2016.

[2] CompuPhase. PAWN: an embedded scripting language. `http://www.compuphase.com/pawn/pawn.htm`, 2016.

[3] R. R. de Novaes Júnior, D. de Sousa Santos, G. M. F. Santiago, and S. R. Dias. Autonomous vehicles networks with GTA San Andreas Mutliplayer. In *13th International Conference on Applied Computing*, Mannheim, Germany, Oct. 2016. International Organization for Development of the Information Society.

[4] E. W. Dijkstra. A note on two problems in connection with graphs. *Numerische Mathematik*, 1:269–271, June 1959.

[5] Google. Google self-driving car project. `https://www.google.com/selfdrivingcar/where/`, 2009.

[6] M. Sá and S. Gorender. Um protocolo de roteamento para redes veiculares. In *Anais do 4° Workshop de Sistemas Distribuídos Autonômicos*, Salvador/BA, Brazil, 2014. Brazilian Symposium of Computing Network.

[7] SAMP. Sa-mp San Andreas Multiplayer mod for Grand Theft Auto: San Andreas. `http://www.sa-mp.com/`, 2016.

[8] R. Tachet, P. Santi, S. Sobolevsky, L. I. Reyes-Castro, E. Fazzoli, D. Relbing, and C. Ratti. Revisiting streets intersections using slot-based systems. *Plos*, 2016.