Torstein Meyer
Fernando Monje
Carlos Villa

Simulation of a Decentralized Network of Autonomous Cars

Decentralized Systems Engineering

1. Introduction

As the development of automated, self-driving car technology advances rapidly, the need for robust intercommunication between cars grows as well. There are many reasons one would want to send messages through a network of self-driving cars - whether it be to warn far-away cars about accidents so that they can reroute, or for close-range communication in order to synchronize traffic stops. Certainly, one could look at this as a decentralized system of independent nodes. We wanted to create a simulation of self-driving cars that operate in a decentralized fashion. To do this we had created a basic movement scheme for a set of cars, and as the cars move around the map, they have to communicate between each other in order to achieve certain objectives - e.g. avoid collisions, negotiating between each other, alerting of accidents on the road and managing parking spots, while they try to reach their target location

2. Related work

There is a lot of related work to this project. Applying decentralized technologies in car networks has been approached to solve the most common problems in transport nowadays: Traffic jams, car sharing, package deliveries… This related work can be found both as simulations (like our project is) and as real implementations that are in early phases. Nevertheless, this related work can be useful for taking ideas. About collision avoidance there is also something out there, but it involves physical sensors like cameras, which we will not use, for obvious reasons.

- **The DAV network [1]:** This is the biggest project we have found related to our topic. They are building a decentralized network “to revolutionize the transportation industry”. In the DAV network, they are approaching a lot of different problems related to transportation. Some of them are not related to our project, like package delivery or some services that involve payments. The common addressed problems between the DAV project and ours are private communication between nodes, services (like parking) offering and news subscription. For service announcement and news sharing, they are using the Quasar algorithm. This enables large scale of nodes communication, but the main drawback is that it has some latency issues. We will not use this technology in our project. For providing anonymity, they use what they call “a DAV identity” that is a pseudorandom identity uniquely identifiable and non-ephemeral that still provides anonymity to the users. These identities are created with private keys of the nodes and stored and tracked using blockchain. We may use this approach in our project for anonymity and privacy insurance. It may be useful so that users cannot be tracked in our network. In general, from this project we can extract a lot of ideas and technologies/protocols useful for us.

- **Scalable decentralized solution for secure Vehicle-to-Vehicle communication [2]:** This is a research paper that uses decentralized solution to protect cars communication from common attacks like DOS attacks, man in the middle… In this paper they prevent these attacks using the Blockchain technology, where each transaction is logged in a decentralized immutable Blockchain ledger. They also present a scalable decentralized platform called "IOTA". In the platform, public keys stored in blockchains are used to solve the above-mentioned attacks. These public keys will be used by users every time a message is sent. The main drawback is that a certifying authority is required, and this against the decentralized principles. From this paper we can obtain strategies for making our network more secure. The public key solution may not be useful in some cases, but we will explore where it is indeed useful and applicable. We will also look for a way to avoid using a certificate authority for the authenticity of the public keys.

- **Decentralized collision avoidance for large teams of robots [3]:** In this paper they address the collision avoidance using decentralized technologies, peer-to-peer connections and trajectory changes. However, this paper does not offer us any interesting approach we can use, as they are based in other physical technologies that are not used in our network.

- **BitCar [4]:** BitCar is a decentralized car sharing platform. It is similar to our solution in the sense that it uses a decentralized architecture to communicate users together with private and public messages in the network. Like the previous solutions, it uses blockchain in the system for things like payments (something we do not have). It would be interesting to see how they approach different issues related to privacy, but the project is not open source, and the technologies used are not available for the public.

3. System Goals & Functionalities

In this project, we have implemented a decentralized network of simulated autonomous cars moving around a grid-like road map. Every car is able to move avoiding collisions with other cars, as well as communicating several pieces of information throughout the network related to different events.

The map that can be seen in Fig.1 where the cars move in, is a grid of 10x10 cells split into 4 areas. The following elements can be found in the map:

- **Cars:** They are the nodes forming the decentralized network. They always have a specific destination (a cell in the map) and their objective is to reach it. Once they do that, they stop.

- **Police:** The police is a special kind of car who is stopped in the same position by default. When it receives an alert of an accident, it moves to that spot to “solve” the accident, meaning that the accident is cleared from the square.
- **Buildings:** They are cells in the map that cannot be entered by any car, they are seen as obstacles.
- **Events:** There are two types of events. A car accident, which is a simulated road accident. When a car gets into a cell where there is an accident, it has to inform the police about it. The second event is a free parking spot. There are some cars in the network searching for parking spots, so when a car finds one it will inform the other car about it.

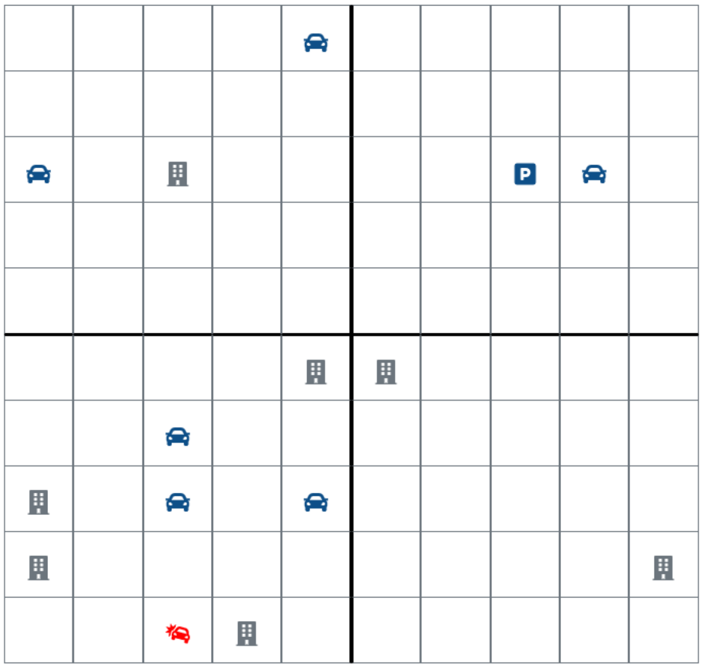


Fig. 1. Image of the city map with cars, buildings, an accident and a parking spot.

Functionalities

A list of concrete system goals with related functionality (indents in the bullet list) follows:

- Simulate a decentralized, automated car that can drive on a simulated road map without colliding with other cars.
    - A road map needs to be created. We will represent the road map as a 2D matrix of “squares” that the cars can move between.
    - The cars must be able to move between squares.
    - The cars must be able to avoid colliding with each other (occupying the same square).
    - The cars must be able to send critical information to other cars in the same area of the city, such as the car’s position, to avoid collisions.
- Cars should be able to subscribe to the news groups the are interested in.

    - Cars will propagate, but not save, the messages that don’t belong to any newsgroup the car is interested in.
    - Cars will subscribe to the news group of spots in they are looking for a parking spot.
    - Cars will subscribe to the area news group they are currently in by default.
- Cars must be able to send emergency messages to the police car, with files attached.
    - The police car must be able to download the file, for example a photo of an accident.
    - The emergency messages must be encrypted.
- Create a communication protocol that allows for cars far away from each other to communicate, without flooding the network.
    - The road map needs to be divided into equal-size sections, or grids, that represent different areas of the city.
    - Cars will communicate at a high frequency with cars in the same area, but with lower frequency to the entire network, to minimize the network traffic.
    - The cars must notify the entire network of cars when they arrive in a new section of the city. When a car does this, other cars in the same section must reply with their IP address.
- Ensure that cars can navigate successfully between two points in the road map
    - Cars must be able to pathfind from point A to B in the network
- Ensure that the network is safe from fraudulent car nodes
    - To ensure that cars in the network are authentic, a web-of-trust mechanism must be implemented, in which cars start the simulation being “aware of” a few cars, gradually building up a trusted network.
- Show a visualization that shows the cars moving in the road map in real-time
    - The road map, divided into city sections, must be shown in a visualization.
    - The cars currently driving on the road map must be shown in real-time in the visualization.
    - Individual tracing information, such as each car’s position, destination and other important data must be visible in the user interface.

4. Architecture

The architecture of the system is seen in Fig.2.

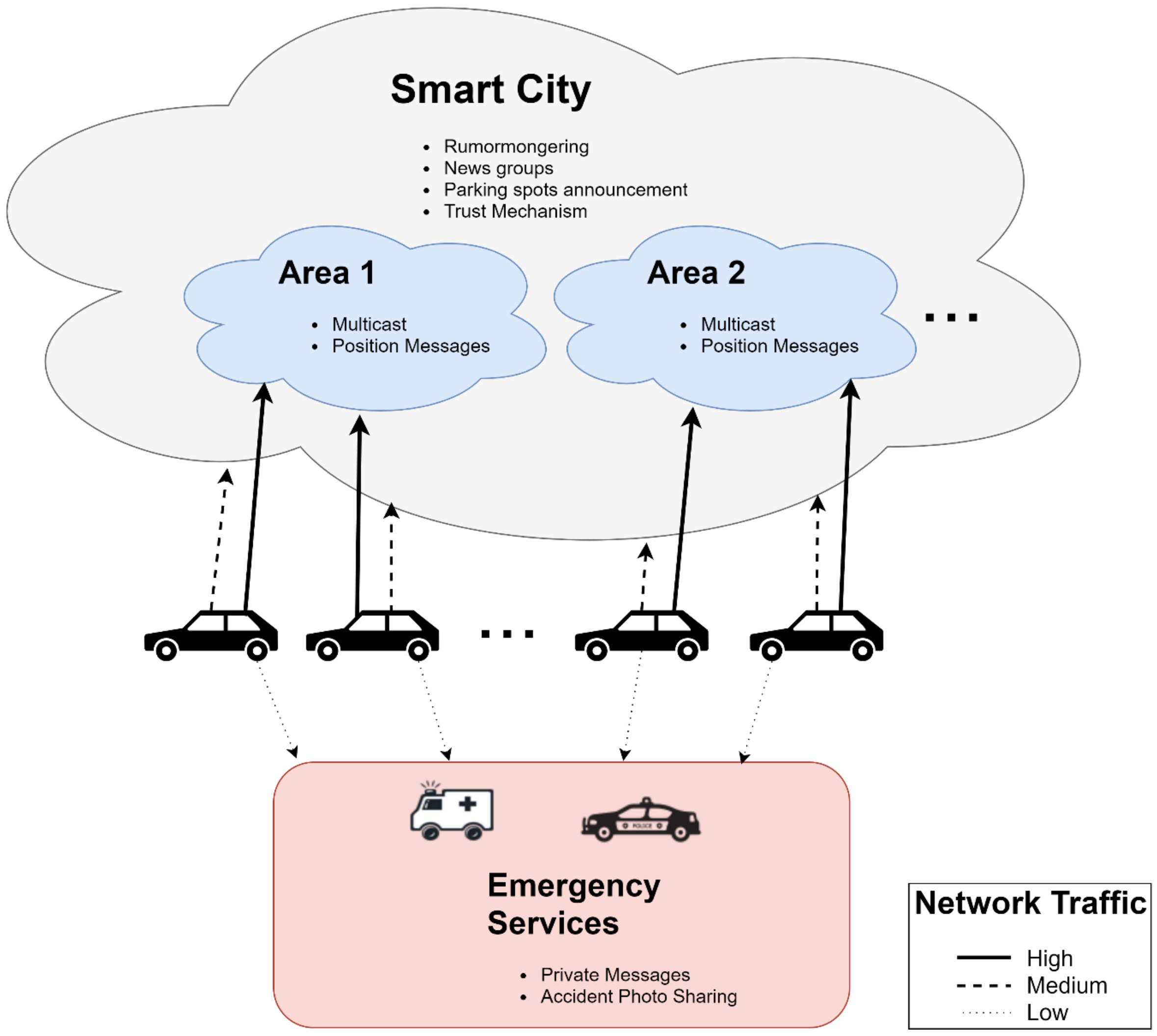


Fig. 2. Architecture of the proposed solution, it is divided in layers using different communication schemes.

Communication stratification

As a city can have a huge number of peers, for efficiency purposes the communication is going to be layered. The smart city is going to be divided in areas. The mechanism is the following:

In the **first layer,** all the cars are only going to be posting area's changes to avoid overloading the network with messages. For this, all cars in an area will be subscribed to the news group of that area, this way, they can get informed about a new member entering the area. There is an additional news group for parking spot, in which parking spots are announced, and cars looking for one are subscribed. These messages are going to be spread with a rumormongering protocol.

The **second layer** consists of the cars in a certain area, and only the cars in that area listen to it. In it, position messages are published every time a car moves so that all cars know the traffic status. The difference between this layer and the news group of the third layer, is that the messages are spread in a multicast way to all cars of the area and their frequency is greater than in the former layer. This layer of communication is faster and more trustworthy than the first one, thus used to avoiding crashes between cars.

Finally, the **third layer** consists of peerster private messaging to inform the police of any accident (in this case the information encrypted), alongside with an image of the accident. Private messaging is also used to request a parking spot that has been announced in the first layer, and to assign it to the winner.

We have used several functionalities of Peerster: The rumormongering protocol from Peerster has been used as the base for the communication in layer 1. In layer 1 we have added the news group intelligence. Regarding layer 3 we have used the private messages of Peerster with a modified file sending scheme from Peerster in layer 3. We have also added encryption capabilities to private messaging. The communication in layer 2 is nothing related to Peerster, and thus has been done from scratch. In brief, we have used the first two homeworks of Peerster as a base for the communication, not using anything from the third one.

Central server

As we are working with a simulation, the are several events real autonomous cars detect with the use of sensors, but the cars of our system will not be able to detect. That is why we have included a central server in our system. This server is not a node of the network but does the function of these sensors informing the cars about an accident or parking spot they should be detecting due to their position.

The communication between the cars and this server is relatively simple. Each time the position of a car changes, it sends the new position to the server. The server checks if there is an event in that position, and if there is, it informs the car about it. The fact that the central server is not a node of the network is what maintains the decentralized paradigm in our system.

Visualization

For the visualization of the simulation, we have developed a user interface in ReactJS. Taking advantage of the fact that there is already a central server with the information of events and car positions, this UI use it for getting all the needed information. The user interface has two states, the setup and the simulation.

In the setup phase, the user can design the map adding buildings in a grid. Then, the user can also add cars, selecting the start position and the destination. Events (accidents and parking spots) can be also added around the map. Finally, the user starts the simulation, informing the central server of the setup who launches the cars.

In the simulation phase, the user interface shows the movement of the cars around the map being able to visualize the traces of each car individually. The traces can be filtered by type, in case the user wants to be only the logs of specific actions. During this phase, the user can also add new parking spots and accidents.

5. Protocol

In this part we explain the network protocol that allows the system to match the requirements set.

Pathfinding

The first thing a car does is define an appropriate path to reach its destination avoiding buildings, which is the information it has at the beginning. This path will be redefined only to avoid a collision with another car, or in the case of the car looking for a spot and obtaining one, this process is explained later in this section. The pathfinding is implemented using Dijkstra's shortest-path algorithm.

Moving inside an area

Cars move following their path, advancing at a defined rate, when a certain time has passed between each step. Every time they move, using the second layer of communication, they inform the rest of the cars of the area about their new position. This way, each car knows the position of everyone else in the same area.

A conflict arises when a car wants to move into a cell where there is another car. In this situation the first car will wait a defined number move turns (typically 2) for the other car to move. If the other car does not move, it means they are both in the position the other wants to be in. In this situation they both start a negotiation process, choosing a random number between a defined range. The one with the lowest number will recalculate its route to avoid the position of the other car, therefore releasing the current position. The one with the highest number will wait for the other car to move and then follow its original path.

Changing area

The process of changing area is more complex. Let's say a car A, that is in area 0, wants to move to a new area 1. In order to move to another area, the car A has to discover all the cars that are already in the new area 1, to communicate with them in layer 2 (which has low latency). For this, it uses a rumor monger message with the news group of area 1 to inform them about its intention to move to the area. All cars in area 1 will add car A to their layer 2 of communication and will also inform it about their positions. Then car A will move into area 1 in its desired position and the cars of area 0 will stop processing his messages as he is in another area, eventually forgetting him. It will also subscribe to the news group of the new area and unsubscribe to the old one.

Like in the movement between squares inside one area, if there is a car in the desired position of the new area, car A will wait for it to move and then follow the changing area protocol. If the car remains in the desired position of A does not move, it means that they are both in the desired position of the other, trying to switch areas. When this happens, a similar negotiation process is followed to decide which car must repath and move from the position and which one takes the released position

Accidents

At the beginning of the simulation, the police car is in the position (0,0) of the map. It will remain there until it receives an alert of an accident. When a car discovers the accident (this happens when the car is in the same cell as the accident). Once this happens, the car will “make” a photo of the accident and upload the photo. Then, it will send a private message to the police car, with the position of the accident and the hash of the photo, to allow the police car to download it.

Finally, once the police car has downloaded the photo it will calculate the shortest path to the accident and help the victims until a new alert appears.

Message encryption

The communication between the police and other cars are encrypted. For this, the police has a private-public key pair, and every car in the network has a copy of the public key. This is a simulation, but in real life, autonomous cars could have this key embedded in the production. As the message encrypted is a private message transmitted through rumormonger, only the content of the message is encrypted. This way it can be transmitted through peerster without any problem.

For the implementation we have used the golang library “crypto/rsa” for the key generation, encryption and decryption. We have used a key length of 2048 bits.

Parking spots

There will be some cars looking for parking spots. If during movement in the map, a car (for example car A) finds a spot, it will announce it in the layer 1 in the newsgroup defined for spots announcements. Only the cars looking for spots are subscribed to that group. Cars interested in the spot will notify so to the car A. This car, after waiting a little bit of time to allow everyone interested in the spot to request it, will randomly select a winner of the parking spot.

Finally, car A will notify the winner and send a private message to him. When receiving the message this car will recalculate its path to end up in this spot, parking there.

This is how the process is carried out without the Web of trust functionality. If the flag to activate the web of trust is set, the system will have some variations that are explained in the next section.

Web of trust

For the web of trust, we have used a private-public key system. Each car of the network has a key pair from the beginning of the simulation. We have divided the cars into 3 groups representing 3 different autonomous cars manufacturers. We make the assumption that autonomous cars from the same manufacturer trust each other, and so, they have each others public key from the start. Let’s see how the web of trust works in more detail.

If a car A trust a car B, car A stores the public key of car B, and signs “car B” with its own private key sending the resulting signature. In our simulation, a car trusting another one means that it thinks it works correctly, and so, if a car you trust announces a parking spot, you believe in the existence of that parking spot. This way, when car B publishes an announcement of a parking spot, it will send all the signatures it has stored attached. This way, if a car C, that does not trust B but trusts A, checks the signatures attached and can verify that car A trusts car B. In our web of trust, by extension, car C will trust car B. The diagram in Fig. 3. explains visually the situation.

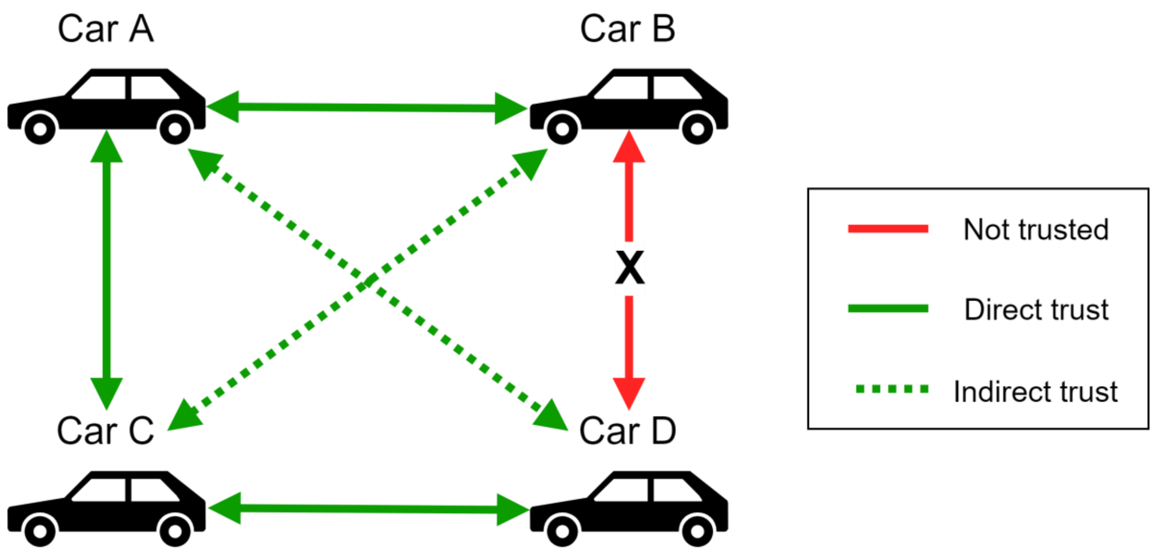


Fig. 3. Example of a web-of-trust with 4 cars

In a direct trust, two cars have each other’s public key and signatures. In an indirect trust, two cars do not have each other’s public key and signature, but they both have a signature the other can verify (from their list of trusted cars).

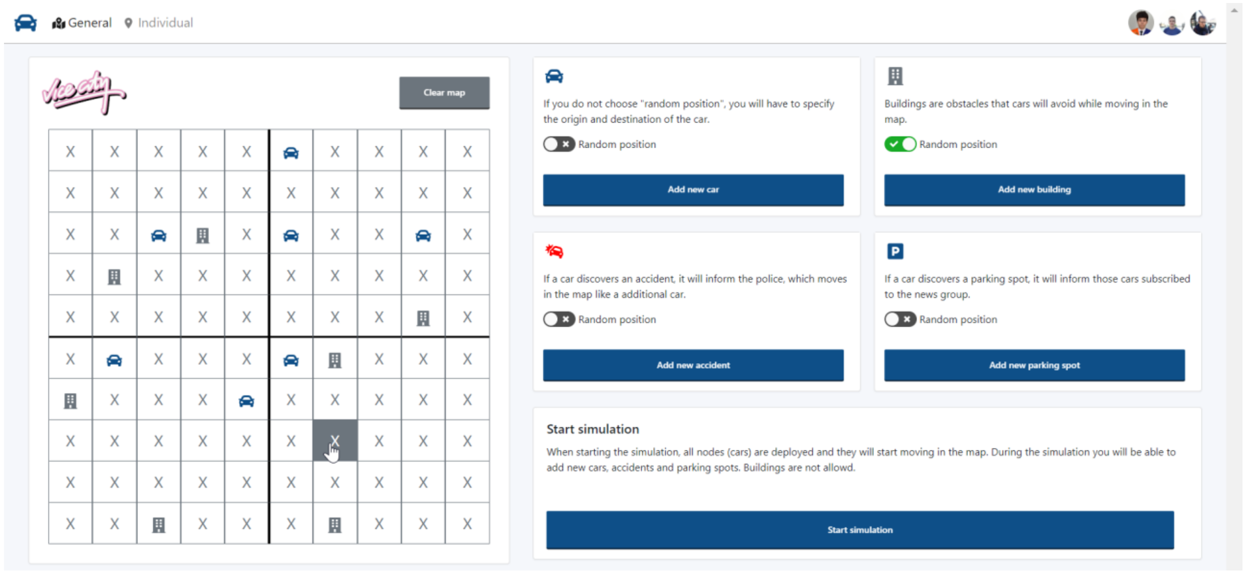
If only cars from the same manufacturer trust each other, there would be only direct trusts. To solve this, we have made that when to cars encounter each other following the “crash avoidance” protocol (see moving inside an area above), and finish successfully, they start trusting each other. This simulates two autonomous cars verifying the well-functioning of each other. They will exchange each other public key and sign each other’s id.

6. Results

In this project, we have created the visualization tool that we have mentioned before. It can be served as a web page over the Internet and is capable of running the simulations in Golang with the parameters introduced by the user. As you can see in Fig.4, a user can select the following features:

- Initial position of cars and their destination
- Position of buildings in the city
- Accidents that have happened.
- Free parking spots to be announced.

Once the simulation has been started accidents and free parking spots can be notified in real time.



If you do not choose "random position", you will have to specify the origin and destination of the car.

Random position

Buildings are obstacles that cars will avoid while moving in the map.

Random position ✓

If a car discovers an accident, it will inform the police, which moves in the map like a additional car.

Random position

If a car discovers a parking spot, it will inform those cars subscribed to the news group.

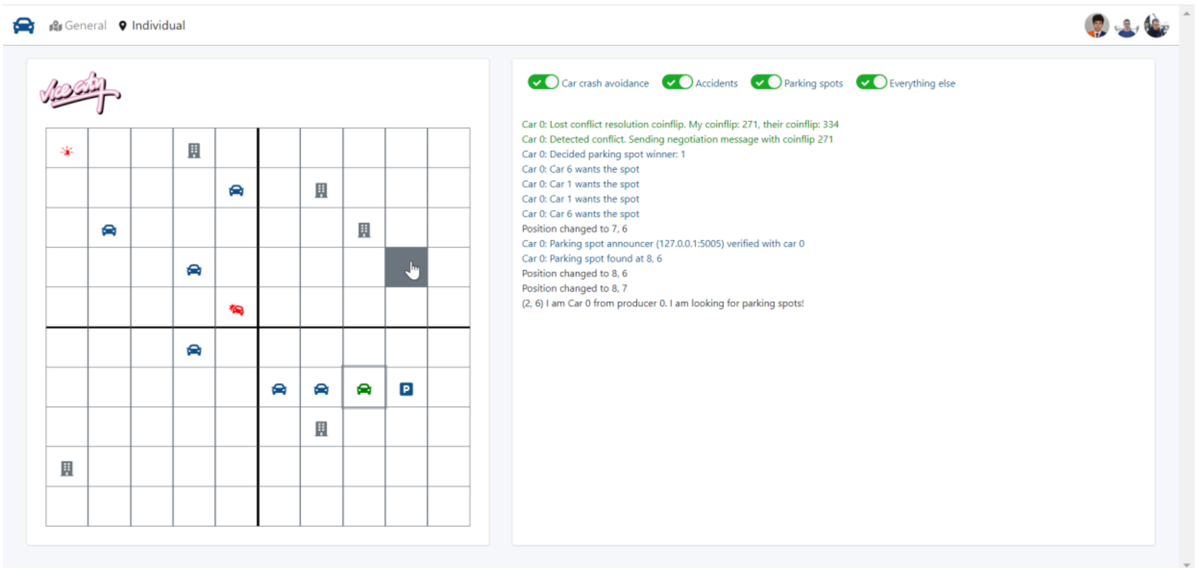
Random position

Start simulation

When starting the simulation, all nodes (cars) are deployed and they will start moving in the map. During the simulation you will be able to add new cars, accidents and parking spots. Buildings are not allowd.

Fig. 4. Graphical User Interface that can be used to configure and run the simulations.

Once the simulation has started, individual information of a car can be seen in real time by clicking on it, opening a new view as can be seen in Fig.5. In this view the user can use different filters to visualize only certain information, such as collision negotiations, accidents and spots requests and assignments.



```
Car 0: Lost conflict resolution coinflip. My coinflip: 271, their coinflip: 334
Car 0: Detected conflict. Sending negotiation message with coinflip 271
Car 0: Decided parking spot winner: 1
Car 0: Car 6 wants the spot
Car 0: Car 1 wants the spot
Car 0: Car 1 wants the spot
Car 0: Car 6 wants the spot
Position changed to 7, 6
Car 0: Parking spot announcer (127.0.0.1:5005) verified with car 0
Car 0: Parking spot found at 8, 6
Position changed to 8, 6
Position changed to 8, 7
(2, 6) I am Car 0 from producer 0. I am looking for parking spots!
```

Fig. 5. View of the Graphical User Interface where information about the cars movement and communications can be seen and filter by different events.

To assess the performance of our solution, we have measured the number of errors that appear based on the number of cars in the network. The number of errors increases with the number of cars as a result of an overload in the network and/or in the peersters making them unable to process the positions of the surrounding cars. We think this happens as a result of our testing computer not having enough processing power. The end result is that two cars can move to the same spot without them realizing in time. In Fig.6 we can see a graph with the number of cars in the simulation and the errors that appear.

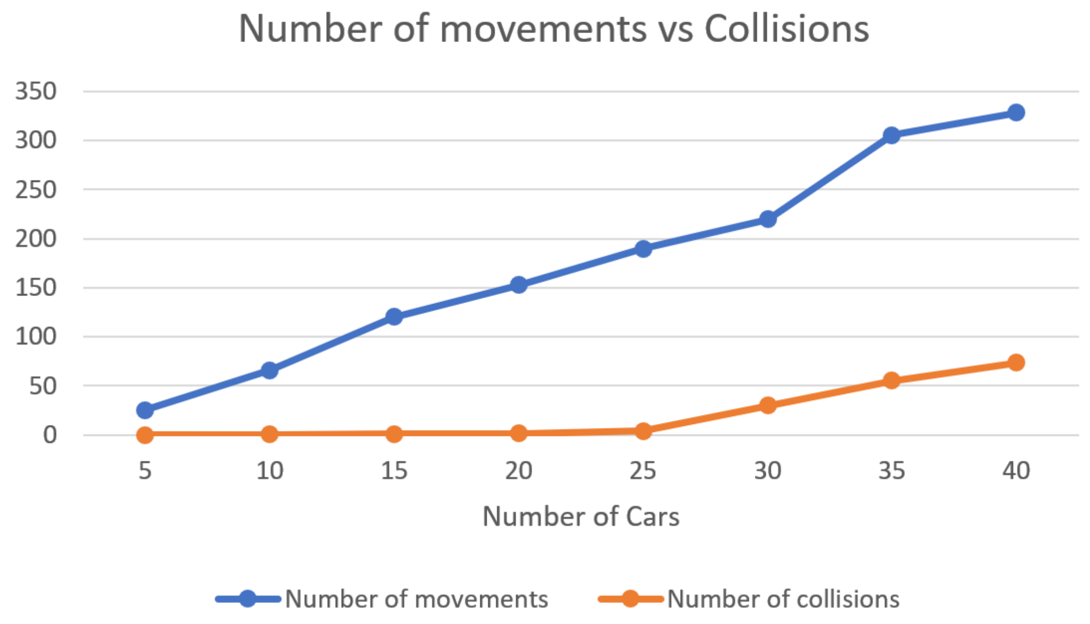


Fig. 6. Graph containing the total number of movements and the movements that were performed incorrectly.

The blue line represents the total number of movements made by the cars. The orange line represents the total number of incorrect movements done i.e. the number of collisions to peers have.
As can be seen the network starts having problems with more than 25 cars at the same time. This means 25 peerster instances communicating in a grid of 100 available positions, meaning 1/4th of the space is already occupied.

In Fig.7 we can see the percentage of errors in respect to the total movements performed. As we can observe the number of errors does not increase linearly with the total number of movements. An increase in movements does not have the same increase in number of collisions.

The information that we present in the plots is the average of the results obtained with 10 measurements per number of cars tag.

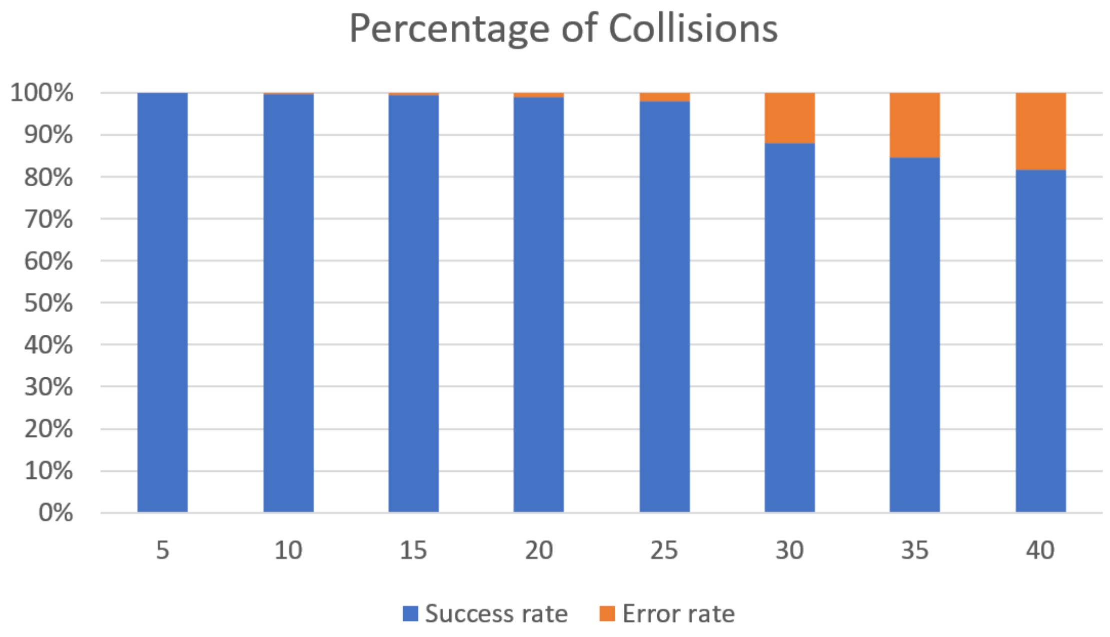


Fig. 7. Graph containing the rate between correct and incorrect movements for different number of cars in the network.

7. Conclusion

In this project, we have attempted to simulate a network of autonomous cars, using a layered networking protocol to avoid collisions in a performant way. From our observed results, we see that in a 10x10 grid network, including various obstructions and hindrances, the protocol works reliably with over 20 active cars. However, we see that as we add a significant amount of cars to the network, collisions start to occur. This is likely due to computer performance issues and/or congestion in the network.

The 3-layer communication protocol also fulfills its purpose by limiting the amount of communication across the entire network. While the 10x10 grid network with 4 areas is little more than a toy example, we still see that by separating a city into discrete areas where the bulk of the network traffic takes place, we can achieve a scalable network.

The implementation of a web-of-trust mechanism in the network also provides a valuable security aspect to the simulation, so that cars only request spots published by a trusted car, to avoid a malicious car publishing a spot that doesn't exist.

Future work

As this is a final project of a subject and not a semester project or a master thesis, there are some aspects it that could have been done in a better way or some improvement that could have been added. To finish this report we will address some of them.

The first issue is that with the coin flip method in the crash avoidance (see "moving inside an area"), a car can lie on the result of the random number to win, and not move. To avoid this we propose a solution. Each car select a number and send the hash to the other. One of them is even and the other odds. Then they reveal their number, sum it and depending on the number parity, one of them will move.

The second issue is the same concept, but lying with the random number in the parking spot contest. A similar solution can be used to solve this, but instead of using even and odd for the winner, a number from 0 to N-1 is used, being N the number of cars applying for the spot. The result will be the sum of all random numbers mod N. To make sure there is no car lying in the result of the sum, a verification by all the car in the network can be done, so that only owning 51% of them you can achieve the cheating.

In the GUI it would be nice to be able to add cars during the simulation. Also, it will be nice to have a higher control on which car wants a parking spot (now it is done randomly). It would also be interesting to have the option of a grid with a desired size, to test how the protocol works in bigger cities.

We have to keep in mind that this is a simulation of autonomous cars. In real life, some of the communications of the simulation are replaced by sensors of the cars (crash avoidance), so the number of errors would drop to 0. Also for security, other practices could be used, like secure hardware with attestation, for example, for trusting each other in the coin flipping process.

REFERENCES

[1][https://dav.network/]
[2][https://www.researchgate.net/publication/333388814_Scalable_decentralized_solution_for_secure_Vehicle-to-Vehicle_communication]
[3][https://ieeexplore.ieee.org/document/6766474]
[4][https://medium.com/causys/bitcar-decentralized-car-sharing-3d19abad6528]