BASIC: Towards a Blockchained Agent-Based SImulator for Cities

Luana Marrocco[1], Eduardo Castelló Ferrer[2], Antonio Bucchiarone[3(✉)], Arnaud Grignard[2], Luis Alonso[2], Kent Larson[2], and Alex 'Sandy' Pentland[2]

[1] Ecole polytechnique de Bruxelles, Université Libre de Bruxelles, Brussels, Belgium
lumarroc@ulb.ac.be
[2] MIT Media Lab, Massaschusetts Institute of Technology, Cambridge, MA 02139, USA
{ecstll,agrignar,alonsolp,kll}@media.mit.edu, pentland@mit.edu
[3] Fondazione Bruno Kessler, Via Sommarive, 18, Trento, Italy
bucchiarone@fbk.eu

Abstract. Autonomous Vehicles (AVs), drones and robots will revolutionize our way of travelling and understanding urban space. In order to operate, all of these devices are expected to collect and analyze a lot of sensitive data about our daily activities. However, current operational models for these devices have extensively relied on centralized models of managing these data. The security of these models unveiled significant issues. This paper proposes BASIC, the Blockchained Agent-based Simulator for Cities. This tool aims to verify the feasibility of the use of blockchain in simulated urban scenarios by considering the communication between agents through *smart contracts*. In order to test the proposed tool, we implemented a car-sharing model within the city of Cambridge (Massachusetts, USA). In this research, the relevant literature was explored, new methods were developed and different solutions were designed and tested. Finally, conclusions about the feasibility of the combination between blockchain technology and agent-based simulations were drawn.

Keywords: Blockchain · Smart contracts · Autonomous Vehicles · Data privacy · Multi-agent based simulation · Smart urban mobility

1 Introduction

In less than 50 years, the global urban population increased from 33 to 54%[1], making the economy of several countries concentrate in cities instead of being uniformly distributed. This drastically influenced the activity inside of the urban area, which highly impacts congestion, accidents and air pollution [1]. The most important source of exposure to pollution for humans is created by road vehicles

[1] https://data.worldbank.org/indicator/SP.URB.TOTL.IN.ZS.

© Springer Nature Switzerland AG 2019
D. Lin et al. (Eds.): MMAS 2018, LNAI 11422, pp. 144–162, 2019.
https://doi.org/10.1007/978-3-030-20937-7_10

and there already have been some attempts to estimate the impact of the pollution by changing from car to bicycle journeys [2]. Moreover, because of the high density of cities and the limited space that is available to parking, cars become an unsustainable mode of transportation [3] even if it can be more convenient in term of flexibility, celerity and comfort. In [4], it was showed that during the peak commuting hours, travel delays increased by 41%, making people more stressed in their life. Moreover, correlation was for example found between depression and traffic noise by analyzing a part of the population of Frankfurt international airport [5].

Cities are changing and urban planning became a new challenge for the world. In response to this, different tools like CityScope [6] developed by the CityScience group at MIT Media Lab were created in order to assist novel urban processes and help to visualize and understand complex urban data and interact with it by simulating modifications within the urban scenario. This type of tools help us to understand the urban impact of new technologies in our lives.

Modern cities attempt to flexibly integrate transportation options for residents and visitors to use buses, trains, taxis, bicycles and cars. They play an important role in the economy of the city and the quality of life of its residents. The inadequacy of traditional transportation models is proven by the growth of alternative and social initiatives aiming at a more flexible, customized and collective way of transport. To be collective, a mobility service should offer a way to organize teams of citizens that need to reach equal or closed destinations starting from different locations. In this context, new kinds of transportation are proposed to citizens like Mobility-on-Demand and ride-sharing transportation [7]. By using shared mobility, the notion of owning a car, using it for personal transportation and leaving it in a parking disappears and gives way to the notion of requesting and splitting a service only when it's needed. A lot of different studies aimed to quantify the impact of car sharing on car ownership and CO_2 emission [8,9], proving that this mobility actually decreases the congestion and those emissions.

In order to go even further in the congestion reduction and the mobility paradigm change, new technologies like Autonomous Vehicles (AV) were proposed because they have the potential to impact on vehicle safety, travel behavior and flow distribution [10]. These vehicles are not totally accepted yet in urban areas because some modifications in the legislation are still needed [11], but new methods are currently investigated and developed to make these vehicles more efficient in data analyzing and decision making [12]. While the National Highway Traffic Safety Administration (NHTSA) statistics tell us that human error is the main reason of road crashes, AVs allows users to enjoy their mobility by reducing the time that they have to monitor the dangers of the road [13]. In an ideal world, it seems that this technology only needs time to be accepted as a regular mode of transportation as well as bus, tram or subway. However, despite all these advantages, some barriers still remain and are the major drag for citizens and users.

Despite the fact that personalized services can be proposed to users by analyzing their personal information, the question of data privacy is becoming more and more relevant with new technologies [14]. According to the literature, 20% of the world's data was collected during this last couple of years [15] and people are starting to understand that these data actually have a real economic value [16]. Nowadays, the most common way to store and access this data is to use centralized databases [17]. However, this centralization encounters more and more issues. First, since the server is the entity that can provide the service, if it stops, the entire system will paralyze. The users will thus not be able to access to the service during the failure time. Second, there is the problem of data privacy. In most cases, all data remain unencrypted, therefore, the entity who has it in its possession can breach the privacy of users [18]. Finally, these databases can be easily modified at the server side, which means that the producers (i.e., users) of the data don't have any control over it and don't know how it is being used [19,20].

One of the most promising technologies to tackle both problems of data centralization and privacy is blockchain. More than a mean for exchanging cryptocurrencies without intermediaries, blockchain technology is starting to introduce different methods in order to achieve a secure and accountable way to share data. For instance, [21] outlined a framework for sharing machine learning models between hospitals, [22] described a method to manage byzantine agents in a swarms of robots, [23] introduced a secure architecture for Internet of things (IoT), and [24] proposed a scoring protocol for autonomous systems to increase their reputation. Complementarily, the urban mobility and smart cities fields are also paying an increasing amount of attention to this evolution. In fact, in order to achieve efficient urban mobility models and smarter cities data needs to be collected and processed to improve urban processes. This concern has driven recent works where the problem of communication among AVs was explored [25], by using a blockchain-based solution.

In response to these concerns, this work addresses the use of blockchain in the urban mobility and smart cities fields by proposing a data-sharing framework among different agents based on *smart contracts*. BASIC (Blockchained Agent-based SImulator for Cities) *aims to combine an agent-based simulator with blockchain technology in order to conduct research on urban scenarios where data are involved and needs to securely shared*. The potential of this framework is illustrated in a car-sharing service where a non-negligible number of personal data is usually collected about users with the current proposed applications. The following sections are structured as follow. Section 2 describes background notions of the simulator and the blockchain technologies used in the framework proposed. Section 3 presents the architecture of the framework and the different parts involved in it and how they interact together. Section 4 describes the results of using the proposed tool in a car-sharing scenario. Finally, Sect. 5 concludes this paper with the discussion and the future work of this simulator.

2 Background

2.1 Car-Sharing Scenario

Today, a new kind of mobility is emerging, with the aim of making our city smarter and more connected. Congestion control, autonomy of users, environmental impact and reduction of accidents are several reasons that motivate the use of Autonomous Vehicles (AVs) in urban areas. However, this new kind of mobility needs to be protected, controlled, and managed. For this purpose, we tested BASIC in a car-sharing scenario in the city of Cambridge (MA, USA), where simulated users and AVs interact. When users need to move around the city, they can request an AV, then, the system forms a group, sends the correspondent AV to users' pick-up points and finally drops them off. BASIC adds a blockchain component to achieve a secure data-sharing approach between users and AVs. BASIC should thus be able to support that kind of infrastructure and should also be stable when the number of AVs and users increase. Finally, all AVs and users have to be connected in order to avoid desynchronization issues.

2.2 Agent-Based Simulation

BASIC is based on a generic existing ABM model [26] design to be easily customized for more specific applications [27]. The ABM model is developed using Gama Platform [28][2]. GAMA allows to model and simulate spatially explicit agent-based simulations where real-world maps, streets, buildings, etc. are integrated by using GIS data. Moreover, different types of agents can be programmed each one with their own behavior and attributes. The behavior of each agent is supported by functions, which can represent reflexes (automatically called every step) or actions (executed when another part of the code calls it). To realize the motivating scenario, two agent species were coded. First, *users* were developed in order to recreate the daily activity of citizens moving from one starting location to a certain destination. Second, *AVs* were developed in order to wander around the urban area and fulfill the car-sharing application. The behavior of both species is explained with more details in Sect. 3.1.

2.3 Blockchain and *Smart Contracts*

The most famous application of blockchain is Bitcoin[3]. Bitcoin is a cryptocurrency introduced in 2008 and the idea behind it is to create a new way to make transactions between peers without a third party in a transparent and secure way. In order to send a transaction from one user to another, a peer-to-peer network is used, which allows to delete the central unit from the process.

The blockchain can be seen as a incorruptible ledger of transactions that is decentralized since the information held on it is duplicated in each one of the

[2] http://www.gama-platform.org/.
[3] https://bitcoin.org/.

different computers (nodes) of the peer-to-peer network. In order to add transactions in the blockchain, every nodes must verify and validate the content of the block. This technology allows thus to create *trust between agents* who don't trust in each other. Each block is composed by transactions. In each transaction, we can find the sender, the receiver, the amount but also additional information can be added. Transactions are made from one address to another. Only nodes with access to the private key of the corresponding address can make a transaction from this address (in other words, if you don't have the secret key of the address A, you will not be able to send a transaction from address A). In addition to transactions, each block contains information about previous blocks. Blocks are thus linked in a chain. Therefore, if the content of the previous block changes (for example, if someone tries to attack the system by modifying a block), the value of this information will also change and will create an inconsistency in the blockchain. For this reason, when something is written in the blockchain, it is very difficult to modify it.

A *genesis block*[4] is created as the starting point of the configuration of the chain. With this block for instance, it is possible to initialize accounts with some amount of cryptocurrency inside. This method is useful to generate tokens in the system, and can only be used when the blockchain system is in the design phase. A second way to generate tokens in the system is mining. In order to validate blocks in the blockchain, the content of the block must be first verified. This is the role of *miners*. Miners are nodes of the network. The goal of them, as we just said previously, is to verify and add blocks at the end of the blockchain. To do so, they have to compete with each other. In fact, in order to validate a block, a computational problem needs first to be solved by miners. The first miner who solves it is considered the winner and can add his block at the end of the blockchain. When a miner succeeds, he is rewarded with a certain amount of cryptocurrency (Ether[5] in case of using the Ethereum blockchain).

In our approach, the Ethereum[6] blockchain is used. The ethereum platform provides the additional capability of deploying *smart contracts*[7] in the blockchain. The advantage of *smart contracts* is that Turing-complete code can be added to the blockchain. Due to this functionality, more elaborated and autonomous operations beyond sending and receiving transactions are possible. A *smart contract* can be seen as a digital contract with rules and conditions. This code is thus composed by variables and functions and is deployed within a certain address in the blockchain. Each node of the network has the possibility to interact with the *smart contract* in a peer-to-peer way. First, interacting with the contract implies to call one of its functions. However, such operation is costly since the user needs to register a transaction in the blockchain. Second, when interacting with the *smart contract* the content of this interaction stays secret and it is held by the *smart contract*, however, the proof that this interaction took

[4] https://en.bitcoin.it/wiki/Genesis_block.
[5] https://www.ethereum.org/ether.
[6] https://www.ethereum.org/.
[7] https://en.wikipedia.org/wiki/Smart_contract.

```
contract AskingCar {
    struct Transaction{
        bytes32 idTransaction;
        bytes32 idPassenger;
        bytes32 idCar;
        ...
        }
    string  private idCar;
    mapping (bytes32 => Transaction) private transactions;
    bytes32[] private idsTransaction;

    /* Constructor of the contract */
    function AskingCar(string id) public {
        idCar = id;
        }

    /* Function that will add the info of the passenger */
    function addTransactionInfo(bytes32 idTrans, bytes32
        idPass, bytes32 idCar, bytes32 start, bytes32 end,
        int hour){
        var transaction = transactions[idTrans];
            transaction.idTransaction = idTrans;
            transaction.idPassenger = idPass;
            transaction.idCar = idCar;
            ...
        }

    /* Function that will add the end hour of
     the drive when the drive is finished
    */
    function addEndHour(bytes32 idTransaction, int endHour){
        if(validTransaction(idTransaction)){
            transactions[idTransaction].endHour = endHour;
        }
    }

    /*Check if the transaction is assigned to this contract*/
    function validTransaction(bytes32 idTrans) view public
        returns (bool) {
        for(uint i = 0; i < idsTransaction.length; i++) {
                if (idsTransaction[i] == idTrans) {
                        return true;
                }
        }
        return false;
        }
}
```

Fig. 1. Smart contract used when a user needs an AV.

place is stored in the blockchain and remains public to its participants. Figure 1 shows a portion of a *smart contract*, implemented in each AV. It is composed by two main functions: ADDTRANSACTIONINFO, used to add info related to the user that needs an AV, and ADDENDHOUR, used to add the end hour of the journey when the user reaches her/his destination.

In this research, the ethereum network was simulated by using Docker[8]. Docker provides container-based virtualization and allows to build networks of agents running specific software in an easy way. The code used to build the containers and the simulations described in this research is publicly available in the following github repository[9].

3 The BASIC Architecture

In order to provide a modular framework that can be customized for different urban applications, BASIC's architecture has been specified and is composed by different layers, defined one above the other. A graphical representation of this stack is shown in Fig. 2. In the following sections we give a description of each layer with the aim of giving details on how the BASIC framework works.

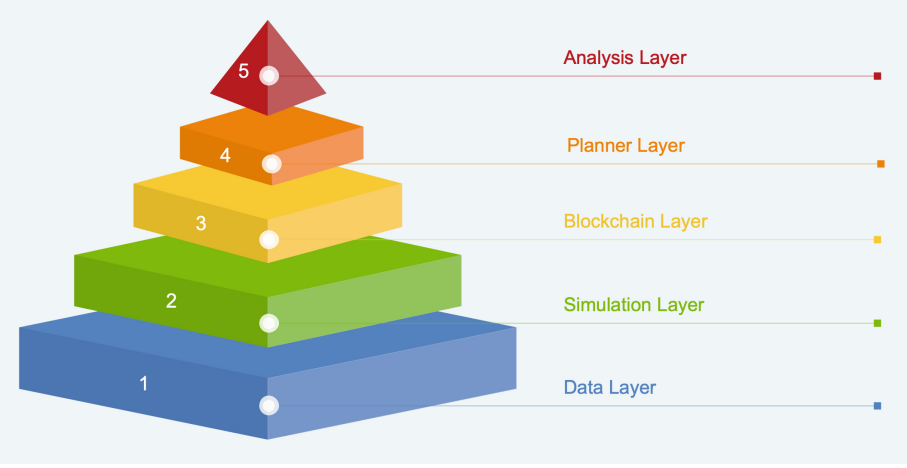

Fig. 2. Multilayer decomposition of BASIC (from bottom to top). First, the data layer represents the actual data used in order to simulate a realistic urban scenario (population information, GIS files, etc). With these data, we were able to build the second layer; the simulation layer was implemented using GAMA which provided different agents such as AVs and users. Third, the blockchain layer creates the infrastructure for the data management in the system. Four, the planner layer aims to guide the system by helping the routing and decision making processes. Finally, the analysis layer sheds some light about the feasibility of the proposed approach.

[8] https://www.docker.com/.
[9] https://github.com/agrignard/Basic.git.

3.1 Data and Simulation Layers

The Data Layer of the BASIC architecture has the objective of creating a virtual environment that replicates a realistic urban scenario.

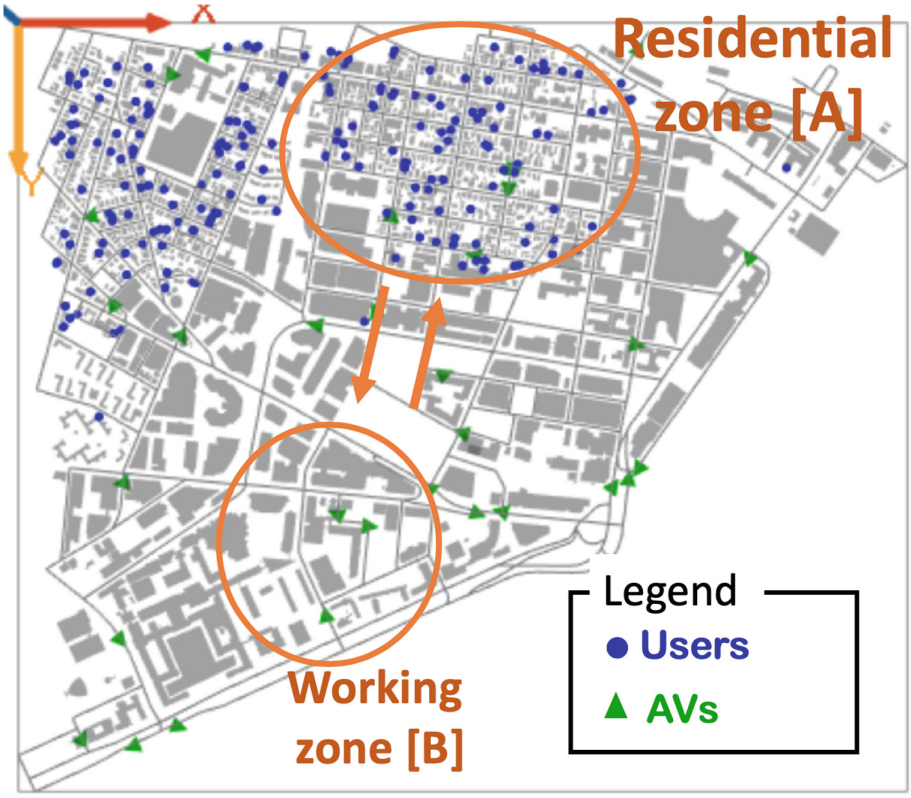

Fig. 3. Simulation of AVs (green triangles) and users (blue dots) in the Kendall urban area, in Cambridge (MA, USA). Two zones are depicted: the residential zone (A) where users live and the working zone (B) where users work. The number of users and AVs were adjusted in the image in order to increase readability. (Color figure online)

The Simulation Layer is an extension of the *CityScope* framework proposed in [26] where *buildings*, *roads* and *citizens* have been already modeled and formed the starting point of our simulations. GIS files have been used in order to replicate the environment and allow us to have a representation of the Kendall area in Cambridge (MA, USA). On top of this, two types of agents have been specified:

Users. These agents represent citizens of the Kendall area. In this simulation, a simple behavior was implemented (see the User Model in Fig. 5). Each user is assigned to a residential (A) and a working (B) zone (see Fig. 3). The only

possibility for the user to go between these two zones is to use an AV. All users are located in zone A when the simulation starts. Then, during the morning hours (i.e., 6–9 AM), at a random point in time, they go to work. After this, during the afternoon hours (i.e., 5–8 PM), at a random point in time, all users leave the working zone to return to the residential zone.

Autonomous Vehicles (AVs). The second class of agents of this system are autonomous vehicles. The unique role of them is to respond to the request of users around the city. Initially, they wander around the map performing a random walk until they receive a user request (see the AV Model in Fig. 6).

As mentioned previously, two types of zones are represented in the simulation. First, we can identify a residential zone (zone [A] in Fig. 3). This zone is composed by houses and apartments. In contrast, the working zone (zone [B] Fig. 3) is composed of company offices and educational institutions.

3.2 Blockchain Layer

This section explains how the BLOCKCHAIN LAYER is built and how it interacts with its previous layer. Figure 4 represents this connection that we describe in the following steps:

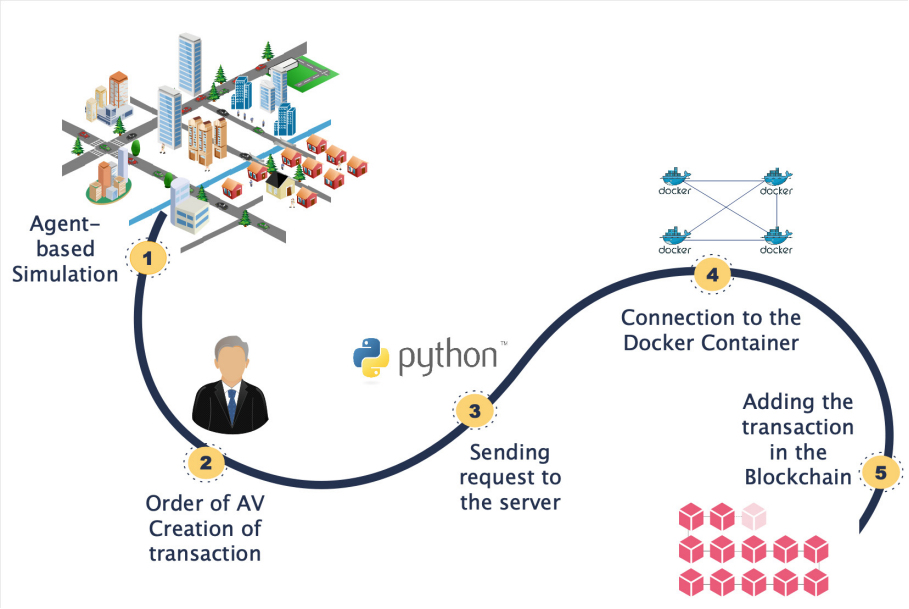

Fig. 4. Process flow of a request of a AV from a user. The starting point is the agent-based simulation. When a user needs an AVs, a transaction is created and sent to the Docker container trough a Python interface. Finally, the container will interact with the blockchain in order to add the transaction in it.

```
species User skills:[moving]{
    building home;
    building work;
    AV myAV <- nil;
    bool waitingForCar <- false;
    ...

     action movement(building start, building end){
          if(currentTransaction = nil){
               do createAndAddTransaction(start, end);
               waitTime <- step*cycle;
          }
          if(myAV = nil){
               askingForCar <- true;
               if(inAGroup = false){
                    copassengers <- findPeople();
                    if(length(copassengers) > 1 or (step*
                        cycle - waitTime) > maxWaitTime){
                          .....
                    }
               }
            }
            if(inAGroup = true and myAV = nil){
               do findCarAndUpdateGroup;
            }
            ......
     }

     action findCarAndUpdateGroup{
          do askAV;
          loop user over: copassengers{
               user.myAV <- self.myAV;
          }
          ask myAV{
               do addPassengers(myself.copassengers);
          }
     }

     AV askAV{
          freeAVs <- AV where(each.isFree = true);
          myAV <- freeAVs closest_to(self);
          return myAV;
     }
     ....
}
```

Fig. 5. User model.

```
species AV skills:[moving, network]{
     list<point> startPoints <- [];
     list<point> endPoints <- [];
     list<User> passengers <- [];
     ...
    action dropOff(User user){
          (user.currentTransaction).endHour <- currentHour;
          ask user{
               ask userClient{
                    string info <- myself.currentTransaction.
                        getStringEndHour();
                    do sendMessage("User;addEndHour;","User",
                         myself.name, info);
               }
          }
         ...
     }
     action addPassengers(list<User> users){
       isFree <- false;
       self.passengers <- users;
       loop user over: users{
          add user.location to: startPoints;
            if(user.nextObjective = "home"){
            add any_point_in(user.home) to: endPoints;
          }
          else{
               add any_point_in(user.work) to: endPoints;
          }
     }
          objective <- "pickUp";
     }

}
```

Fig. 6. Autonomous Vehicle (AV) model.

1 **Agent-based simulation.** This simulation is composed by AVs and users. Each AV is associated to a mining node in the network and deploys its own *smart contract* (as shown in Fig. 1). Each user is associated to a wallet in the blockchain with a certain amount of *ether* pre-filled.
2 **Order of an AV.** When it's time to move from home to work (or vice-versa), the user needs to order a AV. Due to the interaction with the car-sharing algorithm explained in the next section, a AV is assigned to the user.
3 **Sending request to the Python server.** First, a TCP connection is made between GAMA and a Python script. Each user is connected to this code which acts as a client interface. This script acts as an interface between the simulation internals and an external system such as the blockchain. Finally, when an AV is assigned to the user (output of step 2), the same user sends

a message through this interface with the address of the *smart contract* to query (i.e., the one of the requested AV).

4 **Connection to the Docker Container.** To connect with docker, a Docker API[10] is used for the Python language. By using this API, we developed a Python client connected to Docker. Due to this connection, it is possible to enter inside the container and launch code (i.e. script for deploying or querying a contract). For the purpose of this work, we deployed a private blockchain composed of a network docker nodes. Each nodes is associated to one AV and runs geth inside.

5 **Adding the transaction in the Blockchain.** The last step of the workflow is to add the transaction in the blockchain. To this end, the web3-eth JavaScript API[11] is used. This API allows to run geth[12] commands inside of a JavaScript script, which is useful in order to interface with ethereum nodes inside the Docker container (as explained at step 4). By using the Javascript API and a network of interconnected geth nodes, the transaction can thus be sent and added to the network.

3.3 Planner Layer

As mentioned before, the PLANNER LAYER aims to coordinate the fleet of AVs in order to pick up and drop off users. For this purpose, a simple car-sharing model that aggregates users into groups is depicted in Fig. 7.

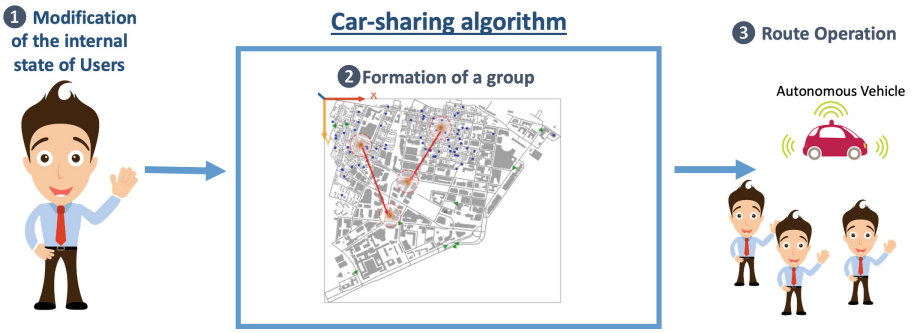

Fig. 7. Workflow of the planner layer. Step 1 consists to change the internal state of the user. By changing this variable, the system knows that this specific user needs an AV. Step 2 is the grouping phase. A car-sharing algorithm will take into account all users that need an AV in order to make groups. Finally, step 3 is route operation phase. Now that a group is formed, the assigned AV is sent to this group in order to pick users up and drop them off.

[10] https://docker-py.readthedocs.io/en/stable/client.html.
[11] https://web3js.readthedocs.io/en/1.0/web3-eth.html.
[12] https://github.com/ethereum/go-ethereum/wiki/geth.

This algorithm is composed of 3 main steps:

1 **Modification of the internal state of users.** In the simulation, each user has a variable which is modified when he/she needs to move. Thus, when a user needs an AV, he/she will change the value of this variable first. This change of internal state takes place at a random point in time during the morning/afternoon shifts explained in Sect. 3.1.

Algorithm 1. Formation of group for the planner layer.

```
1: for user in userWithChangedState do
2:     group ← emptyList
3:     add user in group
4:     remove user from userWithChangedState
5:     while user.waintingTime < maxTime AND group.length = 1 do
6:         for other in userWithChangedState do
7:             if group.length < 5 AND dist(user.start,other.start) < ThresholdStart
               AND dist(user.stop,other.stop) < ThresholdStop then
8:                 add other in group
9:                 remove other from userWithChangedState
10:            end if
11:        end for
12:    end while
13: end for
```

2 **Formation of a group.** The next step is the creation of groups. This process is described in Algorithm 1. In line 1, we can see that this algorithm is executed for each user with a changed state (the state is changed when a user needs an AV). A group will be assigned to this user and the idea is to add other users in this group. To do so, the block (from line 5 to 12) is executed. However, there are two conditions for the execution of this part. First, a limit of time is expressed in line 5. In fact, if a user is looking for a group but no one fits in this group, he will be able to take an AV. Second, when the group is composed of more than one user (the initial one), there is no need to execute again this loop.
Inside this block, the formation of a group is done as follows: Every other users will be taken into account (as shown in line 6), and if he meets some condition, he is added to the group. These users need to be within a threshold distance (*ThresholdStart*) from each other. Moreover, these user's destinations need to be in a place within a certain distance (*ThresholdStop*) from each other. This is represented in line 7.

3 **Route Operation.** If the conditions are met, a group is formed and the closest AV is assigned to the group. This AV picks up and drops off users by always going to the closest stop. If it is not possible to form a group, a maximum waiting time (line 5) was included to prevent lockout periods. If, after this time, the algorithm doesn't find any group, an AV is assigned to a single user as it was explained before.

4 Experiment, Results and Analysis

This section describes the last layer of the BASIC system architecture: the Analysis Layer. As described previously, during the experimental phase of this framework, the feasibility of the integration of blockchain in urban scenario was tested by implementing a car-sharing model. This model focused on the population that uses car and ride-sharing to go work. Some parameters were fixed during the experiments and are explained below:

- **Number of AVs.** In our system, the number of vehicles available in the city was fixed at ten. Because each car is a node of the blockchain network, we have thus ten mining nodes in the system. The first idea was to test the tool and its feasibility with a small amount of AVs. After doing that, extrapolations will be used to draw conclusions about the scalability of the system.
- **Period of time.** During these experiments, we analyze the behaviour of the simulation representing seven days (one week). Each day is the same and the agents have the same behaviour (i.e., go to work and go back home).
- **Distance thresholds for the grouping phase.** During the grouping phase, the distance between starting points of users was fixed to one kilometer. The exact same value was fixed for the ending points of users.
- **Maximum waiting time.** During the grouping phase, it was decided to put a limit on the time that a user can spend for finding a group. This limit was fixed at 15 min.
- **Difficulty of the blockchain.** The difficulty fixes the time needed to mine a block and therefore include new transactions. A too high difficulty value could provoke a slow-down of the system while a too low difficulty value might impact the security of the system. Therefore, in the experiments conducted in the research, we decided to fix the difficulty level.
- **Gas used.** When a transaction is made in the blockchain, it implies to pay a fee for the miner. Each user can choose the fee he/she would like to pay for the transaction by tuning the gas parameter. If the user selects a high value for the gas, the transaction will be mined faster. On the contrary, when the value of gas is low, it might take more time for the transaction to be mined. In this case, the gas was fixed at 25000000 for all transactions. This value remained fixed throughout the experiments.
- **Genesis block.** In order to initialize the blockchain, a genesis block was created. This genesis block contains accounts that were pre-filled with Ether. For the experiments described in this work, ten accounts were initially created and filled up with twenty Ether.

By using a blockchain solution for storing data, we know that a full copy of the ledger is kept in all nodes (AVs). It is important to note that, even though, recent literature suggests that storing data directly into the blockchain might impact the scalability of the system leading to bloat [29], the aim of our experiments is to analyze how much memory is needed in each AV according to the number of users in the simulation and whether a realistic projection of these requirements might exceed current state-of-the-art specifications in the AV field.

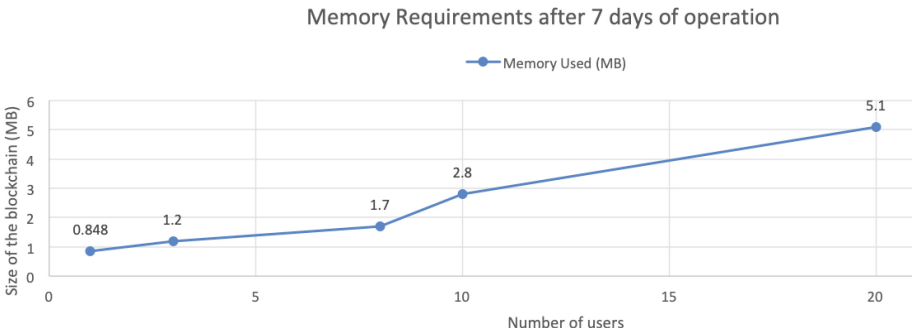

Fig. 8. Amount of memory needed for one AV (node) after seven days of operation in relation to the number of users in the simulation.

Figure 8 shows the amount of memory needed per node (AV) in relation to the number of users active in the simulation. During this test, the number of users was increased from one to twenty. This allowed us to see the evolution of the memory needed per AV. Let's remember that, each day, users need an AV to go to work in the morning, and to come back home during the afternoon. The AV is thus requested two times per day per user. The maximum number of users in an AV is fixed at five, which also correlated to the maximum capacity for user groups introduced in the previous section. According to Fig. 8, when the number of users increase, the size of the blockchain also increases. This phenomenon can be explained by the fact that the size of the blockchain is proportional to the number of transactions. If more users are present in the system, more users will need to travel and thus, more transactions will be created. Let's remember that the only way for users to move, is to use car-sharing. Everyday, each user needs two AVs. However, each request for having an AV corresponds to two transactions. The first transaction represents the request itself. Each user queries the AV by providing all the needed information like the starting and destination points, user address (public key), hour, etc. When the drive is completed, there is the second transaction. The second transaction aims to validate the drive by adding the ending hour when the user finally arrived at his destination point. In conclusion, because users need to travel two times each day, four transactions are added per user per day.

Let's now analyze the memory needed for such a system. As we can see in Fig. 9, after seven days, the size of the blockchain for this system when there is 20 users is 5.1 MB. This amount of data will be stored in each AV. By using this information, we can extrapolate what will be the size of the blockchain after a year. After 52 weeks, by making the assumption that this growth will be linear, the size will be around 265 MB. Due to previous research [6], the population in the Kendall area was roughly estimated to 10.000 people. Moreover, in 2016, 3.5% of the population in Cambridge (where Kendall belongs to) used carpooling

as method of travel[13]. By using this, we can estimate the number of carpoolers to 350. By making the same linear assumption than before and by considering these 350 citizens, the amount of needed memory is thus 4641 MB (4.641 GB). Today, storing that amount of data in an AV is feasible according to current AV specifications.

5 Discussion and Future Work

During this work, we tested the feasibility of the combination between blockchain, agent-based modelling, and urban mobility by proposing a tool named BASIC. This tool was validated by implementing a car-sharing simulation within the city of Cambridge (MA, USA). The blockchain component was introduced in our work to store and share data among a distributed system of AVs and users in order to avoid a centralized controlling entity. First, this study suggests that the memory needed for each AV increased when the number of users increased. However, the simulation process was feasible and fully operational with 30 agents (20 users and 10 cars).

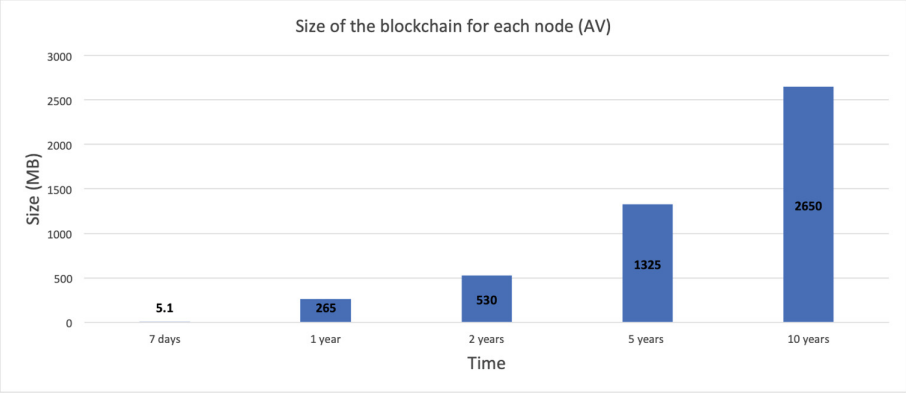

Fig. 9. Extrapolation of the memory needed for one AV (node) with 20 users.

Today, mobility in urban areas is still a big challenge and it remains complicated to test new infrastructures in real life. Due to BASIC, it is now possible to simulate different kinds of urban scenarios where agents interact with a blockchain layer. In the future, other implementation works are possible in order to increase the performance of the system. Improving the car-sharing algorithm is a first idea and will allow to have less cars in the city. The impact on the traffic will also be an interesting feature to analyze in this case. Since this research suggests that storing the data in a decentralized way is feasible, continuing with

[13] https://datausa.io/profile/geo/cambridge-ma/.

the idea of data privacy is an interesting direction to make citizens control their own data. Along those lines, people are realizing the real economic value of their data. Allowing them (by using the *smart contract* technology for example) to choose who can see this information, for how long, and for what purposes, can be another interesting next step that can be implemented by using BASIC. Finally, since a modern city needs to flexibly integrate transportation options including different means (i.e., cars, buses, trains, taxis, and bicycles), the combinatorial complexity of all these possibilities negates the options of a single monolithic control system. How would a grouping, or ensemble of hierarchies perform in this situation? In the near future, we want to extend our approach to deal with Collective Adaptive Systems (CAS) [30][14] able to emerge and continuous adapt in a changing environment.

Acknowledgments. This project has received funding from the European Unions Horizon 2020 research and innovation programme under the Marie Skodowska-Curie grant agreement No. 751615.

References

1. Żak, J., Hadas, Y., Rossi, R. (eds.): Advanced Concepts, Methodologies and Technologies for Transportation and Logistics. AISC, vol. 572. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-57105-8
2. Johansson, C., et al.: Impacts on air pollution and health by changing commuting from car to bicycle. Sci. Total Environ. **584–585**, 55–63 (2017)
3. Fiedler, D., Certický, M., Alonso-Mora, J., Cáp, M.: The impact of ridesharing in mobility-on-demand systems: simulation case study in Prague. CoRR, abs/1807.03352 (2018)
4. Schrank, D., Eisele, B., Lomax, T., Bak, J.: Urban mobility scorecard. Technical report, Texas A&M Transportation Institute (2015)
5. Seidler, A., et al.: Association between aircraft, road and railway traffic noise and depression in a large case-control study based on secondary data. Environ. Res. **152**, 263–271 (2017)
6. Alonso, L., et al.: CityScope: a data-driven interactive simulation tool for urban design. Use case volpe. In: Morales, A.J., Gershenson, C., Braha, D., Minai, A.A., Bar-Yam, Y. (eds.) ICCS 2018. SPC, pp. 253–261. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96661-8_27
7. Chen, X., Zheng, H., Wang, Z., Chen, X.: Exploring impacts of on-demand ridesplitting on mobility via real-world ridesourcing data and questionnaires. Transportation, August 2018
8. Nijland, H., van Meerkerk, J.: Mobility and environmental impacts of car sharing in the Netherlands. Environ. Innov. Societal Transit. **23**, 84–91 (2017)
9. Giesel, F., Nobis, C.: The impact of carsharing on car ownership in German cities. Transp. Res. Procedia **19**, 215–224 (2016)
10. Fagnant, D.J., Kockelman, K.: Preparing a nation for autonomous vehicles: opportunities, barriers and policy recommendations. Transp. Res. Part A: Policy Pract. **77**, 167–181 (2015)

[14] http://www.focas.eu/manifesto/ - FoCAS Manifesto: A roadmap to the future of Collective Adaptive Systems.

11. BBC New: Who is responsible for a driverless car accident? BBC News Online (2015). http://www.bbc.com/news/technology-34475031
12. Millard-Ball, A.: Pedestrians, autonomous vehicles, and cities. J. Plann. Educ. Res. **38**(1), 6–12 (2018)
13. Haboucha, C.J., Ishaq, R., Shiftan, Y.: User preferences regarding autonomous vehicles. Transp. Res. Part C: Emerg. Technol. **78**, 37–49 (2017)
14. Serra, M.: An exploratory paper of the privacy paradox in the age of big data and emerging technologies. Master's thesis, KTH, School of Electrical Engineering and Computer Science (EECS) (2018)
15. Zyskind, G., Nathan, O., Pentland, A.: Decentralizing privacy: using blockchain to protect personal data. In: 2015 IEEE Symposium on Security and Privacy Workshops, SPW 2015, San Jose, CA, USA, 21–22 May 2015, pp. 180–184 (2015)
16. Oyola, J.O., Hoffman, W., Schwab, K., Marcus, A., Luzi, M.: Personal data: the emergence of a new asset class. In: An Initiative of the World Economic Forum (2011)
17. Uber's big data platform: 100+ petabytes with minute latency (2019). https://eng.uber.com/uber-big-data-platform/
18. Former employees say Lyft staffers spied on passengers (2019). https://techcrunch.com/2018/01/25/lyft-god-view/
19. Fan, L., Ramon Gil-Garcia, J., Werthmuller, D., Brian Burke, G., Hong, X.: Investigating blockchain as a data management tool for IoT devices in smart city initiatives. In: Proceedings of the 19th Annual International Conference on Digital Government Research: Governance in the Data Age, DG.O 2018, pp. 100:1–100:2. ACM, New York (2018)
20. Michelin, R.A., et al.: SpeedyChain: a framework for decoupling data from blockchain for smart cities. In: Proceedings of the 15th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services, MobiQuitous 2018, New York City, NY, USA, 5–7 November 2018, pp. 145–154 (2018)
21. Castelló Ferrer, E., Rudovic, O., Hardjono, T., Pentland, A.: RoboChain: a secure data-sharing framework for human-robot interaction. CoRR, abs/1802.04480 (2018)
22. Strobel, V., Ferrer, E.C., Dorigo, M.: Managing byzantine robots via blockchain technology in a swarm robotics collective decision making scenario. In: Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2018, Stockholm, Sweden, 10–15 July 2018, pp. 541–549 (2018)
23. Alphand, O., et al.: IoTChain: a blockchain security architecture for the Internet of Things. In: WCNC, pp. 1–6. IEEE (2018)
24. Alowayed, Y., Canini, M., Marcos, P., Chiesa, M., Barcellos, M.P.: Picking a partner: a fair blockchain based scoring protocol for autonomous systems. In: Proceedings of the Applied Networking Research Workshop, ANRW 2018, Montreal, QC, Canada, 16 July 2018, pp. 33–39 (2018)
25. Singh, M., Kim, S.: Branch based blockchain technology in intelligent vehicle. Comput. Netw. **145**, 219–231 (2018)
26. Grignard, A., Alonso, L., Taillandier, P., Gaudou, B., Nguyen-Huu, T., Gruel, W., Larson, K.: The impact of new mobility modes on a city: a generic approach using ABM. In: Morales, A.J., Gershenson, C., Braha, D., Minai, A.A., Bar-Yam, Y. (eds.) ICCS 2018. SPC, pp. 272–280. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96661-8_29
27. Alfeo, A.L., et al.: Urban swarms: a new approach for autonomous waste management. CoRR, abs/1810.07910 (2018)

28. Grignard, A., Taillandier, P., Gaudou, B., Vo, D.A., Huynh, N.Q., Drogoul, A.: GAMA 1.6: advancing the art of complex agent-based modeling and simulation. In: Boella, G., Elkind, E., Savarimuthu, B.T.R., Dignum, F., Purvis, M.K. (eds.) PRIMA 2013. LNCS (LNAI), vol. 8291, pp. 117–131. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-44927-7_9
29. Castelló Ferrer, E.: The blockchain: a new framework for robotic swarm systems. CoRR, abs/1608.00695 (2016)
30. Bucchiarone, A., De Sanctis, M., Marconi, A., Martinelli, A.: DeMOCAS: domain objects for service-based collective adaptive systems. In: Drira, K., et al. (eds.) ICSOC 2016. LNCS, vol. 10380, pp. 174–178. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68136-8_19