# Energy and Profit-Aware Proof-of-Stake Offloading in Blockchain-based VANETs

Vincenzo De Maio, Rafael Brundo Uriarte and Ivona Brandic
Vienna University of Technology, Austria
{name.surname}@ec.tuwien.ac.at

## ABSTRACT

In Vehicular Ad-hoc NETworks (VANET) users do not necessarily trust each other and in some cases they may introduce dubios information in the network. Centralized approaches to improve the credibility of information do not easily scale, require trusting the service provider and have higher network delay. Blockchain solutions are promising in the area but they need consensus on the credibility of new information, which requires the participants solve computational puzzles in a competitive manner. To cope with vehicles' limited computational resources and mobility, we propose brokerage mechanism to decide whether to execute the validation locally, or to offload it to an edge or cloud infrastructure. We define a Satisfiability Modulo Theories (SMT) method to enable participants to decide whether to take part in the validation and whether to offload it considering the state of the infrastructure, energy efficiency, the offloading cost and the computation reward. Our method obtains 77.7% higher profit and consumes 39.2% less energy in comparison with the case where no offloading is allowed.

## 1 INTRODUCTION

VANET (Vehicular Ad-hoc NETworks) are used in a wide spectrum of applications in traffic management [38], ranging from collision avoidance [4], traffic lights management [26]. However, users of these networks do not necessarily know each other and in some cases they may misbehave and introduce dubious information in the network, which can lead to, e.g., accidents or favour malicious users. Therefore, new approaches to promote trust and improve information are necessary. Solutions relying on centralized approaches [23] do not easily scale, require the users to trust the service provider and have a higher network delay, which can hinder timely decisions, such as, a change of route. Hence, secure distributed solutions can improve trust and the scalability of VANETS.

Blockchain has emerged as a promising solution for enabling secure distributed systems, such as peer-to-peer electronic payment systems [18], smart grids [20]. Several works propose blockchain-based solutions for vehicular networks [30, 37]. However, trust in distributed environments requires the validation and consensus on new information about the environments (e.g., a report of an accident). In blockchain normally the participants need to solve computation puzzles in a competitive manner; broadcasting the solution to other miners; reach consensus; and receive a reward for the correct solutions as incentives (see the Section 2 for details). Despite the fact that validation is feasible in wired blockchain networks, it is still challenging for vehicles in the VANET because of the limited computational resources and the mobility of the vehicles.

Due to this limitation, the existing blockchain-based VANET solutions either avoid the validation, reducing the trust in the networks, or develop a trust management systems that rely on roadside units (RSUs), where vehicles only provide the necessary data for this validation. In this work, we address the limitation by creating a brokerage algorithm that handles the control of validation to the network participants. More specifically, the main contribution of this work is the definition of a brokerage mechanisms that evaluates where and when to execute the validation, i.e., if it is more convenient to execute locally or to offload to edge or cloud computing[1] nodes.

The use of edge nodes, especially, allows exploitation of resources in nodes' close proximity to increase trust in blockchain-based VANETs, reduce latency and energy consumption. Integrating offloading in blockchain-based VANET allows validating a higher number of transactions in a smaller time, with limited overhead for the nodes in the network.

We propose ECbroker, a Satisfiability Modulo Theories (SMT)-based method to help VANET nodes in finding offloading trade-off solution between energy efficiency and profit when offloading the validation to cloud/edge infrastructures. ECbroker allows IoT devices to decide whether to take part on the mining game and where to offload the validation, based on the current state of the infrastructure, the economical cost for offloading and the computation reward. Results show that ECbroker allows to obtain a 77.7% higher profit and a 39.2% lower energy consumption in comparison with the case where no offloading is allowed.

The paper is organized as follows. Section 2 describes background approaches. In Section 3 we define our theoretical model. Section 4 describes our method. In Section 5, we describe our simulation framework and our experimental setup, while Section 6 presents the evaluation. Related works are described in Section 7 and future work and conclusion are discussed in Section 8.

## 2 BACKGROUND

In VANET networks, vehicles receive information about the road status both from sensors (e.g., cameras, noise, pollution sensors) and other vehicles in the network. This information needs to be validated and delivered to vehicles in a timely manner, allowing them to take decisions based on, e.g., traffic condition or weather. However, due to the open nature of VANET networks, new solutions are necessary to confirm the credibility of the participants and of the messages exchanged in these environments.

Blockchain is a distributed, immutable and growing series of records, divided into blocks and linked using cryptography. All

*Conference'17, July 2017, Washington, DC, USA*
2019. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

[1] Although clouds are centralised, offloading the validation process to clouds would not increase the centralisation of blockchain since every validator is independent and they choose their cloud provider, software base, virtual machine, etc.

participants have a copy of the records and new records are broadcasted to the other participants that validate it before adding it to the ledger. The validation, also known as mining, requires the consensus among the participants, which enables a system without a central authority. We adopt this paradigm to create distributed and trusted VANET networks.

Figure 1 illustrates the VANET architecture, briefly described below, which is based on the solution proposed in [37].

- The vehicles and road sensors exchange messages about the environment, such as, traffic congestion and incident reports;
- Each vehicle evaluates the credibility of all the received messages with a positive or negative rating (e.g., +1 if credible and -1 if not);
- The trust value of every vehicle is calculated by summing the offset (positive/negative feedbacks) for all messages from each involved vehicle and a candidate block is created;
- The participants concurrently try solve computation challenges to validate the messages in the network. The winner of the validation process is rewarded by the network.

The last step of the algorithm, also known as blockchain validation, is modelled as a competitive game [16] where each network user (miner) willing to participate in the validation attempts to solve a computation challenges before the other miners in the network. The main steps of blockchain validation are (1) receipt and selection of messages to be validated, which are put together in a *candidate* block; (2) validation of the block; (3) broadcast of the validated block to all the nodes in the network for verification; and (4) the block is added to the blockchain, if it is the first verified solution.

The validation merges concepts from Proof-of-Work (PoW) and Proof-of-Stake (PoS) (for details we refer to [37]). With this algorithm vehicles need to solve computational challenges which difficulty is defined according to the sum of absolute values of messages' offsets in the candidate block as the stake. In this case, larger trust values are more quickly reflected in the blockchain, which ensures the timely update of the recorded data. In other words, the more the participants trust the messages, easier is for miners to solve the computation problem and validate the messages.

While requesting the participants to solve computational challenges by themselves works well with wired blockchain networks, applying it to mobile networks raises several issues, due to the high computational cost of the validation. Another issue is the connectivity of the nodes, which can be variable in this context. Such variability in the connection could significantly affect the broadcast phase, resulting in energy wastage. Some works propose to do the validation in the roadside units (RSUs). However, this approach creates a dependence on RSUs, requires a second blockchain (one between vehicles and one between RSUs), increases centralization and reduces the level of trust and control of the participating vehicles, since a single institution may own multiple RSUs.

Our goal is to allow the vehicles participating in the VANET to take control of the validation and increase trust. In this work, we focus on identifying a method for deciding where to perform validation, while optimising selected objectives. More specifically, we propose brokerage algorithm that handles the control of validation to the network participants, who decides whether to execute the validation or to offload it to cloud or edge in order to ensure a faster delivery of the information.

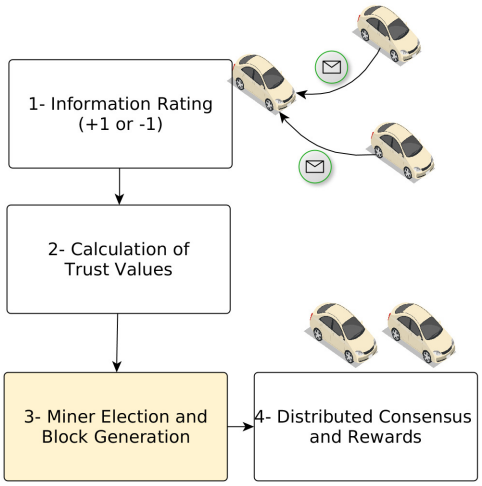



**Figure 1: Blockchain-based VANET**

## 2.1 VANET Information Validation

We describe the main steps of the validation step in Figure 2. In Step 1, each node concurrently looks for new blockchain transactions and check if new blocks are available. Each new transaction is added to the transaction pool. Once a new block is available, the transactions included in that block are removed by the transaction pools. On regular intervals, nodes generate a *candidate* block to be validated (Step 3). In this work, we assume for simplicity that all transactions have the same length, therefore the problem that we want to solve is to identify the number of transactions to be mined. The number of transactions, as well as the edge node selected in this phase should guarantee in the long-term the node will profit from the validation. Once the node selection is performed, the PoS is computed (Step 5) and the result is sent in broadcast to other nodes in the network (Step 6). The PoS is then verified by the other nodes in the network (Step 7) and, if the node is the first one to mine the candidate block, the node obtains its reward (Step 8).

## 2.2 Business Model

The advent of self-driving cars, boats and drones has renewed interest in VANETs. These networks enable communication between its participants, who can exchange valuable current and historical information about, e.g., traffic, weather condition and routes. There are several business models for the implementation and use of such networks, here we focus on blockchain-based distributed VANETs.

In these networks, it is necessary create incentives to the participants to provide credible information and to validate the information received, which can be computation intensive. A valid business model in this case is the similar to other blockchain scenarios: the creation and association of a crypto-currency. In particular, the system could mint new coins to reward the transaction validation.

Nevertheless, the value of these coins is a function of demand and offer. To sustain the demand and the value currency, which

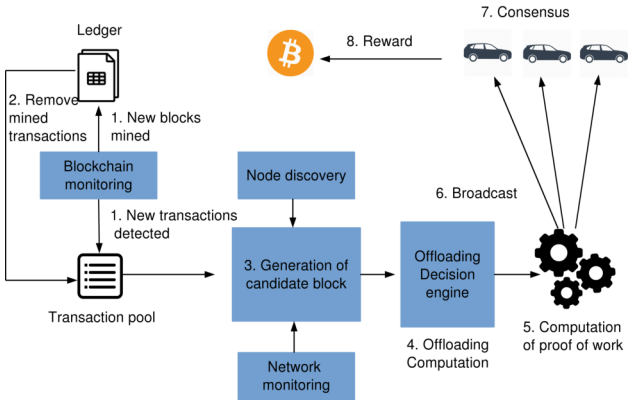



**Figure 2: First-hop block validation overview**

attracts and stimulate participants to generate and validate data, the system could require the use of coins for multiple purposes. For instance, they might be required for the payment of fixed membership fee or fixed fees for on-demand information; for integrating the network with other service providers, such as, GPS providers or traffic information broadcasters; to pay for requesting specific information to participants; and for purchasing general statistics related to traffic.

# 3 THEORETICAL MODEL

## 3.1 VANET model

We model the VANET as a directed graph $\mathcal{V} = (\mathcal{N}, \mathcal{L})$, where $\mathcal{N}$ is the set of computational nodes in the VANET and $\mathcal{L}$ is the set of network connections between nodes. Also, we define a map $\mathcal{M}$, containing a set of geographical coordinates, in terms of latitude and longitude, and models the area where $\mathcal{V}$ is deployed. The state of a computational node $n$ at a instant $\tau$ is defined as in Equation 1

$$n = (\texttt{THREADS}, \texttt{MIPS}, pos, \tau), \tag{1}$$

where $\texttt{THREADS}(n, \tau)$ represents the amount of threads available on node $n$ at time $\tau$, $\texttt{MIPS}(n)$ represents the computational capabilities, expressed in MIPS (millions of instructions per second) of each core in vehicle $n$, while $\texttt{pos}(\texttt{n}, \tau)$ represents the position of node $n$ over map $\mathcal{M}$ at time instant $\tau$. We distinguish three types of computational nodes: On-board units (OBU), edge nodes and cloud nodes. On-board units are the communication and computation enabled devices mounted on each vehicle, that allow to perform simple computations; edge and cloud nodes, are, respectively, edge and cloud data centers. Edge nodes model computational facilities deployed close in the urban area, as in [5], while cloud nodes model cloud data centers. For simplicity, we divide OBUs, edge and cloud nodes in three different sets, namely $\mathcal{O}$, $\mathcal{E}$ and $\mathcal{C}$. The main differences are that (1) $\mathcal{O}$ nodes are mobile, while $\mathcal{E}$ and $\mathcal{C}$ and (2) $\mathcal{E}$ nodes hardware capabilities are significantly less than nodes in $\mathcal{C}$, as stated by [5]. Therefore, we define the set $\mathcal{N} = \mathcal{O} \cup \mathcal{E} \cup \mathcal{C}$, with $\mathcal{O} \cap \{\mathcal{E} \cup \mathcal{C}\} = \emptyset$, $\mathcal{E} \cap \{\mathcal{C} \cup \mathcal{O}\} = \emptyset$ and $\mathcal{C} \cap \{\mathcal{O} \cup \mathcal{E}\} = \emptyset$.

Concerning the set of network connections $\mathcal{L}$, it contains the network connections between nodes in $\mathcal{N}$, namely, $\mathcal{L} \subseteq \mathcal{N} \times \mathcal{N}$. For each $l_{i,j} = (a_i, a_j) \in \mathcal{L}$ we define $\texttt{latency}(l_{i,j})$ and $\texttt{bbw}(l_{i,j})$, respectively the latency and the bandwidth available on the network connection between $n_i$ and $n_j$.

## 3.2 PoS model

*3.2.1 PoS validation time.* The validation time for a PoS, as defined in Section 2, is the time that a node in the blockchain requires to decrypt each transaction in the block. The decryption time varies for each PoS $p$, therefore cannot be determined in advance. However, it depends mostly on two parameters: the *difficulty* value, specified in the configuration of the blockchain network, and the threads that each node in the blockchain assigns to the decryption task. Based on this, we define the PoS MI as in Equation 2,

$$mi_{pos}(p, n, \tau) = X(\texttt{params}(\texttt{THREADS}(n, \tau), diff)) \tag{2}$$

where $p$ is the PoS, $n$ is the node where the PoS is validated, $\tau$ is the time instant where PoS is validated, $X$ represents a random variable, following a specific distribution, and $\texttt{params}$ are the parameters of distribution of $X$. Concerning $\texttt{params}$, we assume that the parameters are a function of the number of threads available on node $n$, $\texttt{THREADS}(n)$ and on the value $diff$, modelling the difficulty value of the blockchain. The validation time of a PoS $p$, excluding offloading, is then equal to

$$t_{pos}(p, n, \tau) = \frac{mi_{pos}(p, n, \tau)}{\texttt{MIPS}(n) \cdot \texttt{THREADS}(n, \tau)}, \tag{3}$$

as we assume that each thread gets the full CPU over the node $n$ which executes the validation of $p$. The distribution used in this work is described in Section 5.5.

*3.2.2 Offloading model.* We have two possibilities for offloading: cloud or edge nodes. We assume edge infrastructure to be organized as a Small Cell cloud infrastructure with a double coordinate system, as in [11]. For each cell, we have $x$ and $y$ coordinates. The distance between each cell is defined as in Equation 4.

$$d(n_1, n_2)) = |x_1 - x_2| + \max(0, \frac{|x_1 - x_2| - |y_1 - y_2|}{2}). \tag{4}$$

If at least one between $n_1$ or $n_2$ is a cloud node, we assume $d(n_1, n_2)$ to be equal to a random value that models internet latency, $K_c$. In Figure 2, offloading a PoS requires to (1) offloading the PoS data, (2) computing the PoS, (3) broadcasting to all vehicles in the blockchain network. Therefore, we define the offloading time from OBU $o$ to target node $n$ as

$$t_{val}(p, o, n) = d(o, n) \cdot latency(o, n) + \frac{data(p)}{bw(o, n)} + t_{pos}(p, n, \tau) + broadcast(result(p), o). \tag{5}$$

where $data(p)$ is the amount of data of transaction $p$ and $result(p)$ is the hash of the transaction $p$. The time for broadcasting the result of validation to other participants in the blockchain is instead defined

as $broadcast(result(p), o)$ in Equation 6.

$$broadcast(result(p), o) = \sum_{o_i \in O \setminus \{o\}} d(o, o_i) \cdot latency(o, o_i) + \frac{result(p)}{bw(o, o_i)} \tag{6}$$

## 3.3 PoS cost model

Offloading on a cloud/edge infrastructure has a economical cost for the users. This cost is for the use of the computational resources on the cloud/edge layer, that are rented to the users in form of containers. We define a set of container instances available on a cloud/edge infrastructure, namely $\mathcal{I}$. The price for execution on a node $n$ $p(i, n)$ depends on two main parameters: the type of container instance $i$ selected and node $n$ where the instance is running. We define it in Equation 7,

$$p(i, n) = \begin{cases} 0 & \text{if } n \in \mathcal{O} \\ c_i & \text{if } n \in \mathcal{C} \\ c_i + c_e(i, n) & \text{if } loc(i) \text{ is an edge node} \end{cases} \tag{7}$$

where $c_e(i, n)$ is an additional quantity for execution on edge nodes, defined according to [11]. Based on this function, we define the cost of validating a single transaction $p$, $c(p, i)$ as in Equation 8

$$c(p, n) = p_i(i) \cdot t_{val}(p, v, n). \tag{8}$$

We also define as $\mathcal{P}(o)$ the set of all transactions validated and/or offloaded by OBU $o$. Finally, we define the cost paid by a OBU $o$ as

$$C(o) = \sum_{p \in \mathcal{T}(o)} c(p, n). \tag{9}$$

*3.3.1 Reward.* OBUs get a reward as an incentive to perform validation, proportional to the number of transactions that they validate. We assume that reward is fixed to a quantity $r$. For this reason, reward for a OBU $o$ is defined as in Equation 10

$$\mathcal{R}(o) = r \cdot \mathcal{T}(o). \tag{10}$$

We define the profit for a OBU $o$, $Pr(o)$, as

$$Pr(o) = \mathcal{R}(o) - C(o) \tag{11}$$

## 3.4 Energy model

Validation process consumes energy on the on-board unit of the vehicle. Energy consumption is the integral of power consumed by the on-board unit $P_{obu}$ over time. We define the instantaneous power for validating a transaction $p$ on the OBU $o$ as the sum of the power of processing and the in Equation 12

$$P_{obu}(p, o, \tau) = P_{cpu}(o, \tau) + P_{net}(p, o, \tau) + P_{idle}(o). \tag{12}$$

For the CPU power, we employ the model of [3], described by

$$P_{cpu}(p, o, \tau) = \sum_{i=0}^{i < \mathsf{THREADS}(o)} \beta_{\mathtt{freq}}(i, \tau) \cdot \mathcal{U}_{cpu}(v, \tau) + \beta_{\mathsf{base}}, \tag{13}$$

where $\beta_{\mathtt{freq}}(i, \tau)$ is a constant dependent on the frequency of CPU used by thread $i$ at the instant $\tau$, $\beta_{base}$ is a hardware dependent constant and $\mathcal{U}_{cpu}(o, \tau)$ is the CPU utilization of the OBU $o$ at time instant $\tau$, as defined by Equation 14.

$$\mathcal{U}_{cpu}(o, \tau) = \frac{\mathsf{THREADS}(o, \tau)}{\mathsf{THREADS}(o)}, \tag{14}$$

where $\mathsf{THREADS}(o, \tau)$ is the number of threads available at time instant $\tau$. Concerning the power for offloading the PoS p, we define the power consumption of network transfer by expanding the model proposed by [1], from which we derive our instantaneous power function, defined in Equation 15

$$P_{tx}(n_1, n_2, t) = P_{amp}(n_1, n_2) \cdot bw(n_1, n_2) \cdot d^{\upsilon} \tag{15}$$

where $P_{amp}$ is the power for amplification between the two nodes, $bw(n_1, n_2)$ is the bandwidth available between the two nodes, $d$ the distance between the two nodes, $\upsilon$ the path loss component, $t$ the instant when transmission is happening. When no transmission is happening, $P_{amp}(n_1, n_2) = 0$. Energy consumption is defined instead as in Equation 16

$$E_{tx}(p, n_1, n_2) = \int_{t_{start}}^{t_{end}(n_1, n_2)} P_{tx}(n_1, n_2, t)\, dt \tag{16}$$

where $t_{start}$ is the instant where transmission starts and $t_{end} = \frac{data(p)}{bw(n_1, n_2)}$ is the time when transmission ends. All energy coefficients are in Table 2.

## 3.5 SMT encoding of the problem

Every time a set of transactions $\mathcal{T}(\tau)$ is available, each OBU has to decide if it is worth to take part to the validation, and if so, create a suitable transaction block and take a offloading decision which satisfies its energy consumption and profit constraints. We model this constraint satisfiability problem in terms of Satisfiability Modulo Theories (SMT). These formulas contain various operations, ranging from booleans, arithmetic, arrays and recursive datatypes, combining the power of constraint programming with first-order logic to obtain an assignment that satisfies infrastructure and users' constraints [19]. The reasons for using SMT lie in its capabilities to express infrastructure and QoS constraints and its applicability to near real-time scheduling problems [7]. Also, SMT can be solved by standard SMT solvers [12], extensions of typical SAT solvers that check satisfiability of formulas made of Boolean variables and operations. Solver is described in Section 5.6.

*3.5.1 Basic definitions.* First of all, we define the matrix $R(\mathcal{T}, \mathcal{I})$ with dimentions $|\mathcal{T}| \times |\mathcal{O}|$, where each row is a transaction $p_i \in \mathcal{T}$ and each column is a OBU $\{o\} \in \mathcal{O}$. Each cell $R_{i,j}$ contains the time required to perform validation of transaction $p_i$ on node $j$, namely

$$R_{i,j} = t_{val}(p_i, o_j, o_j). \tag{17}$$

similarly, we define the matrix $P(\mathcal{T}, \mathcal{I})$, with $P_{i,j}$ equal the profit of validating transaction $p_i$ on OBU $o_j$ and $E(\mathcal{T}, \mathcal{I})$, with $E_{i,j}$ equal to the energy consumption for validating transaction $p_i$ on OBU $o_j$. Assignment of transactions to OBUs is modelled by matrix assignment $A = |\mathcal{T}| \times |\mathcal{O}|$. Each cell $A_{i,j}$ is defined as

$$A_{i,j} = \begin{cases} 1 & \text{if } p_i \text{ is assigned to OBU } o_j \\ 0 & \text{otherwise.} \end{cases} \tag{18}$$

In our definition, we want each transaction to be assigned at most to one OBU. Therefore, we impose the constraint

$$\forall i \in [0, |\mathcal{T}|] \sum_{j \in [0, |O|]} A_{i,j} \leq 1. \tag{19}$$

Also, we want that each OBU does not get more transactions than its block size. Therefore, we also impose the constraint

$$\forall j \in [0, |O|] \sum_{i \in [0, |\mathcal{T}|]} A_{i,j} \leq \mathcal{B}. \tag{20}$$

In our formulation, we also have to consider offloading of block validation from OBUs to computational (edge or cloud) nodes. For this reason, first, we define the offloading runtime, profit and energy matrices $R^o$, $P^o$ and $E^o$. These matrices have dimensions $|O| \times |C \cup \mathcal{E}|$, and at each cell $R^o, P^o, E^o{}_{i,j}$ contain, respectively, the runtime, the profit and the energy consumption for offloading the transaction block from OBU $i$ to computational node $j$. Then, we define the offloading matrix $Off$ as matrix $A$, namely

$$Off_{i,j} = \begin{cases} 1 & \text{if } o_i \text{ offloads to node } n_j \in \mathcal{E} \cup C \\ 0 & \text{otherwise.} \end{cases} \tag{21}$$

Similarly to Equation 19, we impose that a block cannot be offloaded to multiple nodes, then

$$\forall i \in [0, |O|] \sum_{j \in [0, |\mathcal{E} \cup C|]} Off_{i,j} \leq 1. \tag{22}$$

For simplicity, we also define a vector $\omega$ with length $|O|$, defining whether a OBU $i$ offloads a PoS to any computational node.

$$\omega_i = \begin{cases} 1 \iff \exists j | Off_{i,j} = 1 \\ 0 \text{ otherwise} \end{cases} \tag{23}$$

*Problem definition.* Our SMT can be seen as finding an assignment for $Off$ and $A$ such that, for each OBU in $O$, the values for runtime, profit and energy respect the constraints defined by each OBU. Constraints defined by each OBU are modelled by a vector $Goal(o_i) = \langle pr_i, ec_i \rangle$, namely, cost and energy consumption desired by OBU $o_i$. Also, we have to consider that each block has to be mined respecting a given deadline $d$. Each OBU can choose to offload its block to a computational node. Based on the previous definitions, we define the constraints of our SMT formulation in the following equations:

$$(1 - \omega_j) \sum_{i \in [0, |\mathcal{T}|]} \sum_{j \in [0, |\mathcal{E} \cup C|]} A_{i,j} R_{i,j} + \omega_j Off_{i,j} R^o_{i,j} \leq d \tag{24}$$

$$(1 - \omega_j) \sum_{i \in [0, |\mathcal{T}|]} \sum_{j \in [0, |\mathcal{E} \cup C|]} A_{i,j} P_{i,j} + \omega_j Off_{i,j} P^o_{i,j} \leq Goal_j(pr_i) \tag{25}$$

$$(1 - \omega_j) \sum_{i \in [0, |\mathcal{T}|]} \sum_{j \in [0, |\mathcal{E} \cup C|]} A_{i,j} E_{i,j} + \omega_j Off_{i,j} E^o_{i,j} \leq Goal_j(ec_i) \tag{26}$$

Respectively modelling runtime, profit and energy constraints for each vehicle $j$. Each To obtain the constraints of our SAT formulation, we combine Equations 24, 25 and 26 using a AND logical connector and adding also Equations 20, 22 and 19. We omit capacity constraints of computational nodes for brevity.

## 4 ECBROKER

The goals of ECbrokerare to determine (1) whether or not to participate to the validation, and (2) how to perform the validation. For each OBU, the algorithm goes through the following phases: (1) *Reading transaction pool*, where each OBU access the transaction pool to check non-validated transactions; (2)*Estimation phase*, where each OBU determines if it can participate to the validation

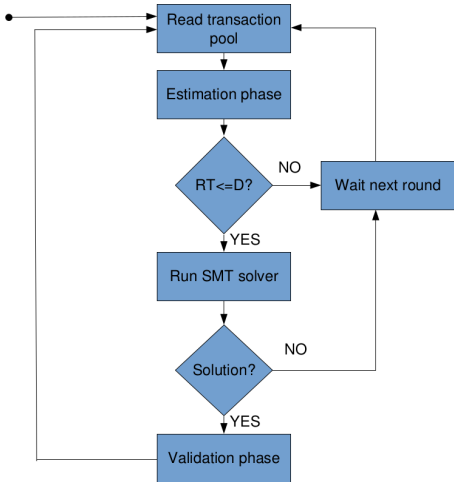



**Figure 3: ECbroker flowchart**

satisfying the deadline imposed by the blockchain; (3)*SMT Solver*, where, based on results of estimation phase, OBU executes the solver to identify an assignment satisfying the OBU's constraints; (4)*Creation of candidate block*, where, according to solver's results, the OBU creates a candidate block of transactions; (5)*Validation phase*, where the OBU performs the validation of the candidate block. ECbroker is detailed in Algorithm 1.

**Algorithm 1** ECbroker Algorithm

```
1:  function ECBROKER(o, T, Goal(o), P, d)
2:      nT ← |T|
3:      nThreads ← getAvailableThreads(o)
4:      q ← estimateRuntime(P)
5:      bSize ← calculateBlockSize(q, d)
6:      if bSize ≥ 1 AND q · bSize ≤ d then
7:          cNodes ← getCandidates(o, dist, threshold)
8:          mapping ← SOLVE − SMT(T, bSize, q, Goal(o), cNodes)
9:          (rt^r, pr^r, ec^r) ← validate(mapping)
10:     elsereturn
11:     end if
12: end function
```

The input includes the current OBU $o$, the transaction pool $\mathcal{T}$, the goal vector for OBU $o$, $Goal(o)$, the deadline $d$ and a probability value $P$ that is used for the estimation phase. $P$ is used because it is not possible to know in advance the time the validation will take. However, in our model we assume that this time is modelled by a probability distribution, whose parameters depend on the difficulty and the number of threads utilized for validation. Therefore, first, we obtain the number of threads available on OBU $o$ (line 3). The estimation phase, described by Algorithm, starts at line 2.

**Algorithm 2** Estimation phase of ECbroker

```
1: function ESTIMATERUNTIME(o, T, P, d)
2:     nThreads ← getAvailableThreads(o)
3:     X ← selectDistribution(nThreads, diff)
4:     param ← parametersFor(X, nThreads, diff)
5: return quantile(X, P)
6: end function
```

Due to the nature of PoS computation, it is impossible to know in advance the time they will take for their execution. However, since we model them accordingly to a probability distribution, we can identify an upper bound for the execution. By definition, a p-quantile $q$ of a probability distribution is a value such that all the values coming from a random variable $X$, following a given probability distribution with specific parameters, are lower than $q$ with probability $P$, namely, $P(X \leq q) = P$. We assume that each node contains a repository of probability distributions that can be selected according to the number of available threads and the *diff* value. Once we have the upper bound $q$, we can determine a suitable block size at line 5. By our assumptions, all transactions have the same data size, and difficulty value is fixed. Therefore, we assume that upper bound that we have for the validation time is equal to $bSize \cdot \alpha$. Since we want upper bound to be lower than deadline $d$, we set $bSize \cdot \alpha \leq d$. Also, to this time, we have to add the time it takes to compute the mapping for the validation, coming from the SMT. We call this quantity $t_{map}$. Therefore, the block size is calculated according to Equation 27

$$bSize = \left\lfloor \frac{d - t_{map}}{\alpha} \right\rfloor . \tag{27}$$

Afterwards, we check on line 6 if it is possible to include at least one transaction in the newly generated block. If that is the case, the algorithm proceeds by solving the SMT and performing the validation. The SMT can be solved by using a standard SMT solver. To accelerate computation of SMT solver, we restrict the evaluations only to a restricted set of nodes, determined by the function *selectCandidateNodes*, described by Algorithm 3.

**Algorithm 3** Selection of candidate nodes

```
1: function selectCandidateNodes(o, dist, threshold)
2:     cNodes ← ∅
3:     pNodes ← E ∪ C
4:     for all n ∈ pNodes do
5:         if d(o, n) ≤ distAND|cNodes| ≤ threshold then
6:             cNodes ← cNodes ∪ {n}
7:         end if
8:     end for
9: end function
```

In this case, selection is performed based on the distance between the nodes and the OBU and on a threshold on the number of nodes that we consider. Once the mapping is calculated, we validate it by applying the mapping obtained by the SMT solver and checking the values obtained by it, at line 9.

# 5 EXPERIMENTAL SETUP

## 5.1 Simulator

Preliminary evaluation is performed using simulations. After investigating different edge simulators, like iFogSim [15] and edgecloudSim [36], we decided to base our simulation on SLEIPNIR[2], the extended version of FogTorchPi described in [11]. SLEIPNIR is a edge simulator running on Apache Spark, which allows it to easily scale according to underlying computational resources. Moreover, it provides validated models for Edge/Cloud infrastructure.

[2]https://github.com/vindem/sleipnir

| $n \in \mathcal{N}$ | CPU | MIPS | $c_i$ |
|---|---|---|---|
| cloud-* | 16 | 10000 | 0.03 |
| edge-* | 8 | 2500 | $0.03 + c_e(i)$ |
| obu-* | 4 | 1000 | 0 |

**Table 1: Hardware configuration.**

| Coefficient | Value |
|---|---|
| $\beta_{\texttt{freq}}$ | 6.9320 |
| $\beta_{\texttt{base}}$ | $625.25e-6$ |
| $P_{amp}^{3g}$ | $0.025e-6$ |
| $P_{amp}^{wifi}$ | $0.007e-6$ |

**Table 2: Energy coefficients**

| Connection | Availability | QoS profile | | Probability |
|---|---|---|---|---|
| | | Latency (ms) | Bandwidth (Mbps) | |
| 3G | 0.9 | 54 | 7.2 | 0.9957 |
| | | ∞ | 0 | 0.043 |
| WiFi | 0.1 | 15 | 32 | 0.9 |
| | | 15 | 4 | 0.09 |
| | | ∞ | 0 | 0.01 |

**Table 3: Network availability distribution.**

We extend this version by adding support for (1) OBUs, (2) VANET networking and (3) PoS modelling and (4) mobility modelling.

## 5.2 Infrastructure model

We assume that CPU specifications of cloud and edge nodes, do not change during each different run of the simulation. We assume that edge nodes have fewer capabilities than cloud nodes in terms of cores and MIPS [5]. Hardware setup for each node type is summarized by Table 1. We assume 6 cloud locations and a Edge node for each cell. Concerning the connections inside cloud and edge layer, we set up each link bandwidth and latency according to the specifications in Table 3. For connections intra-layer, we distinguish two main layers: (1) connections between OBUs and edge/cloud layer and (2) the connections between edge and cloud layer. For the first, due to the unreliability of connections in mobile data distribution services (caused by mobility, environmental factors and reduced availability of edge nodes), we need to accurately model the unreliable connections between OBU and edge/cloud nodes to perform an accurate simulation. The QoS provided by each link $l_i \in \mathcal{L}_{\mathcal{I}}$ as a random variable $r'(l_i) = \langle \texttt{latency}(l_i), \texttt{bw}(l_i) \rangle$. By our assumption, OBU can use two types of connections: 3G and WiFi. Availability of connection is determined by a random variable. If both are available during the execution, the faster is selected. For available latency and bandwidth we use the probability distribution of [11] which is summarized in Table 3. `latency` $= \infty$ and `bw` $= 0$ means that the two computational nodes are not connected. If a cloud node is selected for placement, the Internet transmission delay $K_c$ between 100 and 300 milliseconds [11] is considered. We model $K_c$ as a Gaussian random variable with $\mu = 200$ and $\sigma = 33.5$ [11]. Concerning the connections between OBUs and edge nodes, bandwidth and latency are set according to Table 1, multiplying the latency by the distance between two nodes (see Equation 4). Distance depends on the coordinates of each node, as defined in Section 5.3.

## 5.3 Node distribution

We simulate the execution of ECbroker on real-world urban areas selected among the neighborhoods in the city of Vienna. We imagine that edge node are deployed over a urban area $\mathcal{M}$ in a way similar to cellular network, according to a small-cell setup with cells of $2km^2$, with a single edge node per cell, as in [11]. The number of vehicles is calculated as $2 \cdot \frac{AREA}{2}$, where $AREA$ is the size of the area we are considering in $km^2$ and 2 is the area of each cell in $km^2$, which is typical in works targeting SCC [2]. We extract the areas' maps from OpenStreetMaps[3], and use them also for mobility simulations. Number of cloud nodes is set as 6 to simulate the geographical distribution of [33].

## 5.4 Mobility Simulation

To simulate OBU's mobility, we employ mobility traces generated using SUMO simulator [32]. SUMO is a state-of-the-art mobility simulator that allows to simulate vehicular and pedestrian traffic on OpenStreetMap maps. To perform our simulation, we extract maps of the areas described in Table 4. Then, we use the maps extracted from OpenStreetMaps as input for the SUMO mobility simulator [6], which is used in many works on VANET, such as [34]. First, we collect logs about relative coordinates of each vehicle moving in the aforementioned areas, with sampling interval of 1 second. For each vehicle, we generate different logs. Once the simulation is complete, we use the generated logs to simulate mobility of subscribers, updating the coordinates of each subscriber according to the mobility traces as long as our simulation advances.

## 5.5 PoS modelling

As described in Section 5.5, the MI for the transaction validation are generated using a random variable. Since the difficulty is set at the moment of initialization of the blockchain, we perform our modelling by fixing the difficulty value and varying the amount of threads that perform the mining, to simulate the variation of computational capabilities. We collect data about 5 hours of mining for different number of threads in the $[1, 16]$ interval (the number of cores available on the selected machine), which gives us more than 10000 total readings of validation runtimes. Mining is performed over a Intel Xeon E5-2623. By analyzing the validation times, we observe that (1) time is higher with lower of threads, and (2) the shape of the runtime distribution is very similar between different thread numbers. (see Figure 4). The *diff* value is set to `0xffffff` in all the nodes of our blockchain, which is a typical test value used in experimental blockchain setup. To identify the best fit for this distribution, first, we draw a Cullen-Frey graph to identify the squared-skewness and the kurtosis of the data, using R package `fitdistrplus`. Then, once observed the similarity of these values with the ones of $\beta$ distribution, we run a goodness-of-fitness $\chi^2$-test with 0.01 confidence, using R `EnvStats` package. $\chi^2$ and the $p - value$ results are summarized Table 5.

The test is passed when a $p - value$ higher than the specified significance value is obtained. We use these preliminary results to simulate the validation time, according to the number of threads available on the nodes.

[3]https://www.openstreetmap.org

| Area | Extension ($km^2$) | # OBU | # edge nodes | # cloud DC |
|---|---|---|---|---|
| Hernals | 11.35 | 24 | 36 | 6 |
| Leopoldstadt | 19.27 | 40 | 100 | 6 |
| Simmering | 23.23 | 48 | 144 | 6 |

**Table 4: Data about deployment area.**

| Threads | $\alpha_{thr}$ | $\beta_{thr}$ | $\chi^2$ | $p - value$ |
|---|---|---|---|---|
| 1-3 | 0.6554 | 4.2105 | 13.7933 | 0.1826 |
| 4-6 | 0.6694 | 3.3925 | 28.0288 | 0.0210 |
| 7-9 | 0.6991 | 3.4636 | 38.1903 | 0.0240 |
| 10-12 | 1.1231 | 9.2888 | 31.1597 | 0.2223 |
| 13-14 | 1.1011 | 211.24 | 28.9226 | 0.2671 |
| 15-16 | 0.7889 | 1.9633 | 24.7511 | 0.4100 |

**Table 5: Statistical analysis.**

## 5.6 SMT Solver

For compatibility with SLEIPNIR, we evaluated different solvers offering a Java API, such as SMTInterpol [8], STP [14] and Z3 [12]. Among them, we selected Z3, due to its wide applications to real-time scheduling problems [7]. Also, Z3 solver allows us to set a timeout on the solution, which makes it suitable for near real-time scenarios. When the timeout of the solver expires, it returns the best solution that he could find until then. To simulate a Ethereum blockchain, we set its timeout to 1.5 minute. The used Java API is available online[4].

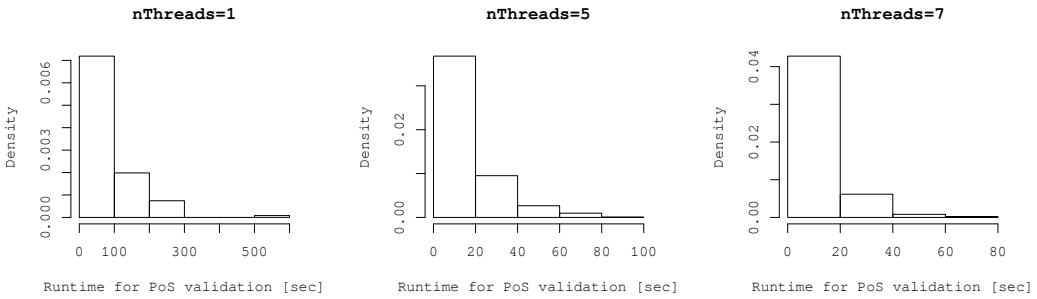
# 6 EVALUATION

In this section we evaluate ECbroker in terms of profit and energy consumption. To calculate the profit, we assume $r = 1$ for each transaction for which a OBU completes the validation. We simulate 1 day of execution and generate $n$ new transactions in rounds of 2 minutes, where $n$ is a uniform random variable with range $[1, |O|]$, which models the fact that at most one transaction per OBU is generated at each round.

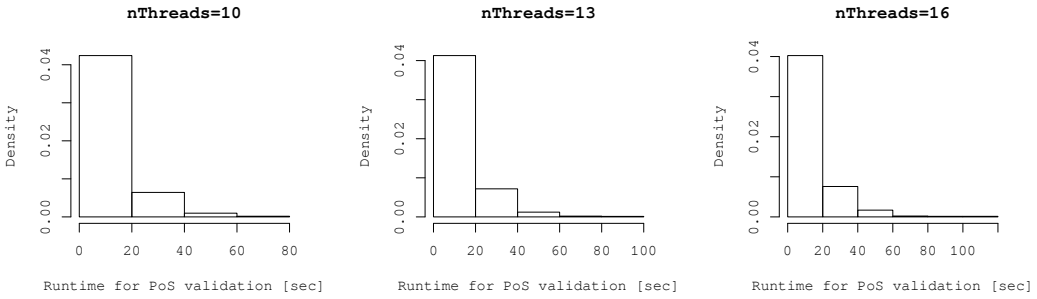
## 6.1 Quantile evaluation

We compare ECbroker performance using different $P$ values, as this affects the upper bound used to take our offloading decisions. We variate $P$ between $[0.05, 0.95]$ and see how profit and energy consumption are affected. We simulate a edge/cloud infrastructure deployed over areas in Table 4. Results are shown in Figure 5. "H" lines refer to `HERNALS` area, "L" to `LEOPOLDSTADT` and "S" to `SIMMERING`. Each value is the average value over each simulation round. We see that lower $P$ implies higher energy consumption. This is because a lower $P$ causes runtime underestimation, which causes the vehicles to perform more local processing. This also causes a higher profit due to the reduced offloading, as we see from Figure 5(c) which show the average percentage of offloaded transactions per round by each OBU. By increasing $P$, we observe instead an opposite tendency, due to the overestimation of the PoS runtime which triggers more offloading to edge nodes. The choice of this parameter allows user to adjust algorithm behavior, depending on what he/she prefers between profit and energy consumption.

[4]https://github.com/Z3Prover

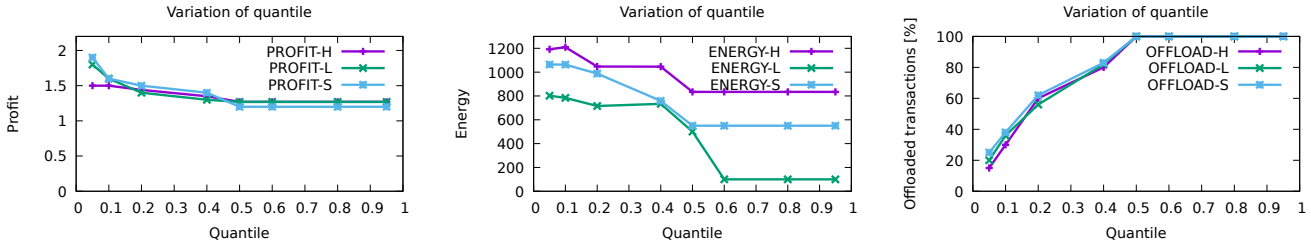



(a) Runtime distribution, 1 – 3 threads (b) Runtime distribution, 4 – 6 threads (c) Runtime distribution, 7 – 9 threads




(d) Runtime distribution, 10 – 12 threads (e) Runtime distribution, 13 – 14 threads (f) Runtime distribution, 15 – 16 threads

**Figure 4: Mining runtime distributions on Intel Xeon E5-2623, 5 hrs execution.**



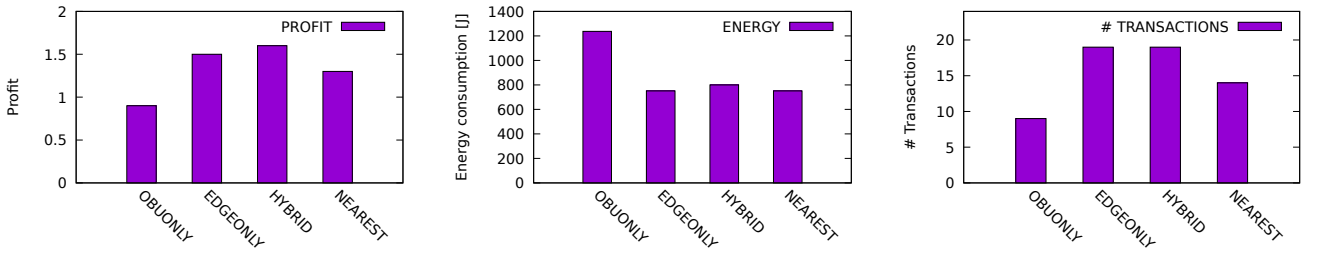

(a) Average profit per OBU (b) Average energy consumption per OBU (c) Percentage of offloaded transactions

**Figure 5: Quantile variation**

## 6.2 Infrastructure comparison

We compare performance of ECbroker in four different scenarios: OBUONLY, where no offloading is used, EDGEONLY, where only edge nodes are used for offloading, HYBRID, where offloading is performed on a hybrid cloud/edge infrastructure, and NEAREST, which uses the same setup of EDGEONLY but allows offloading only on the node that is closer to the OBU. The $P$ value for the quantile is set to 0.5, as it provides the best compromise between energy and profit. For brevity, we show only results for HERNALS area, since the other areas show similar trends. We also showthe average number of transactions per round that OBUs are capable to validate.

We see from Figure 6 that the use of hybrid cloud/edge infrastructures allows a 77.7% improvement for profit and a reduction of energy consumption by 39.2% in comparison with OBUONLY. The higher profit is due to the increased number of transactions that can be evaluated, as shown in Figure 6(c). Also, the energy consumption for offloading transactions is significantly less than energy for processing, with a positive effect for the average energy consumption. Concerning the NEAREST approach, while energy consumption is comparable to the EDGEONLY case, profit is 18.7% less than the HYBRID case and 13.3% less than EDGEONLY case. Results shows the benefits of having multiple offloading possibilities, which however requires a brokerage mechanism to select the best trade-off solution.




(a) Average profit per OBU (b) Average energy consumption per OBU (c) Average number of transactions validated per round

**Figure 6: Infrastructure comparison**

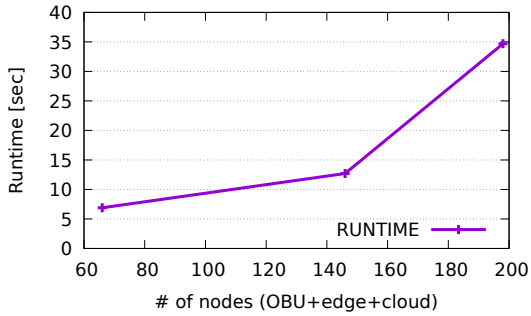



**Figure 7: Scalability evaluation**

### 6.3 Scalability evaluation

Finally, we analyse the runtime of ECbroker to show its applicability to blockchain-based VANETs. We measure the average runtime over 1000 simulations of a execution round of ECbrokerover all areas described in Table 4. We run the simulation using JVM 1.8 in a Intel(R) Core(TM) i7-7500U laptop. The simulated runtime is shown in Figure 7. 95% confidence intervals for `HERNALS`, `LEOPOLDSTADT` and `SIMMERING` runtimes are, respectively, [4.8, 14.7], [23.4, 59.9] and [33, 63.2]. We see that over each area ECbroker delivers a timely validation solution that satisfies users' constraints. In average, the block validation time obtained by placement calculated by ECbroker is 55.6 seconds, which is adequate to the deadline of 2 minutes set by typical blockchain protocols, such as Ethereum. It is also of note that, due to the difficulties in deploying a node in each cell, we expect a number of nodes closer to 60, even in bigger urban areas, which results in a runtime around 5 seconds. Moreover, this time could also be significantly reduced by performing an embedded non-Java implementation, e.g., using C++ library for Z3, and by performing further code optimization for different target devices. Nevertheless, these preliminary results are promising and show that using SMT solver for offloading decisions of PoS validation is a promising research direction.

## 7 RELATED WORKS

Blockchain has emerged as a promising solution for enabling secure distributed systems, such as peer-to-peer electronic payment systems, smart grids, and vehicular networks. Blockchain-based VANET are discussed in [27]. In VANET context, several works focus on reward model [29, 35], without considering offloading.

In [24] the authors study the impact of mobility the validation process in VANETs. A particular interesting aspect of this study is the probability winning the mining process, which could be adopted in our solution as a future work. Yet, the authors do not consider the possibility of offloading this process.

In [22] an authentication mechanism for vehicular fog infrastructure is proposed. The work focus the cross data centre authentication but does not mention the message validation. In [37] the authors develop a blockchain-based VANET architecture. However, their approach depends on third-party RSUs and requires two blockchain networks. Similarly, Kang et. al [21] propose a blockchain-based solution for vehicular networks focusing on the definition of reputation of vehicles since it is a good indicator of data quality. Moreover, they consider RSUs as edge nodes, which participate on the validation of the messages. However, like similar approaches, the RSUs and a centralised cloud are used to validate messages, which takes control from the vehicles and reduces trust in the network. In our approach, we also use edge and cloud infrastructures but the participants of the network can choose where to execute the validation (including locally) and the edge and cloud infrastructure are not owned by a single entity.

The benefits of mobile offloading for enhancing capabilities of mobile devices has been shown by [25]. Frameworks like [9, 10] have been proposed for mobile workloads, but without considering latency critical applications or blockchain context. Offloading capabilities of VANET have been analytically investigated by [13]. Other works target computation offloading on cloud infratructure [28] or on hybrid cloud/edge infrastructures [17], but without without considering requirements of PoS offloading. Similar to our work, [31] focuses on blockchain offloading for VANET, but they rather focus on PoW computations, rather than on PoS, therefore having less latency requirements in comparison to our scenario. Plus, their focus is more on security and content caching, rather than on computation offloading.

## 8 CONCLUSION AND FUTURE WORK

In this paper, we presented a preliminary study about a method for PoS offloading over VANETs based on SMT. First, we identify the theoretical background about VANETs, blockchain and SMT solvers. Then, we define a theoretical model for PoS offloading problem, and we define it in terms of SMT, then we design ECbroker, a algorithm which allows each node in the VANET to decide where to offload their transaction block in order to improve both energy

efficiency and profit. Results show improvement of 77.7% for profit and a reduction of energy consumption of 39.2%, between the cases with and without offloading. Also, we perform an evaluation of different upper bounds for ECbroker, modelled by parameter $P$, and identify the best value for energy consumption and profit. Finally, we evaluate the runtime of our simulation, showing the possible applicability of SMT to the VANET scenario.

In future work, we plan to provide a real-world implementation of ECbroker, showing its applicability in a real world scenario. Also, we will investigate methods to accelerate the computation of a solution for SMT, in order to apply such technique to a wider spectrum of mobile applications, with stricter latency requirements. We also plan to consider other aspects in our Broker, such as, the business model and other Quality of Service terms.

## ACKNOWLEDGMENTS

This research was funded by the Rucon project (Runtime Control in Multi Clouds), FWF Y 904 START-Programm 2015 and by the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie Grant Agreement No.838949.

## REFERENCES

[1] S. Agarwal, A. Das, and N. Das. 2016. An efficient approach for load balancing in vehicular ad-hoc networks. In *2016 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*. 1–6.

[2] A. Ahmed and E. Ahmed. [n. d.]. A survey on mobile edge computing. In *2016 10th International Conference on Intelligent Systems and Control (ISCO)*. 1–8.

[3] Farhan Azmat Ali, Pieter Simoens, Tim Verbelen, Piet Demeester, and Bart Dhoedt. 2016. Mobile device power models for energy efficient dynamic offloading at runtime. *Journal of Systems and Software* 113 (2016), 173 – 187.

[4] D. Anadu, C. Mushagalusa, N. Alsbou, and A. S. A. Abuabed. 2018. Internet of Things: Vehicle collision detection and avoidance in a VANET environment. In *2018 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*. 1–6. https://doi.org/10.1109/I2MTC.2018.8409861

[5] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. 2012. Fog Computing and Its Role in the Internet of Things. In *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing (MCC '12)*. ACM, 4.

[6] Shafinaz Buruhanudeen, Mohamed Othman, and Borhanuddin Mohd Ali. 2007. Mobility models, broadcasting methods and factors contributing towards the efficiency of the MANET routing protocols: Overview. *2007 IEEE International Conference on Telecommunications and Malaysia International Conference on Communications* (2007), 226–230.

[7] Z. Cheng, H. Zhang, Y. Tan, and Y. Lim. 2016. Scheduling overload for real-time systems using SMT solver. In *2016 17th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*. 189–194.

[8] Jürgen Christ, Jochen Hoenicke, and Alexander Nutz. 2012. SMTInterpol: An Interpolating SMT Solver. In *Model Checking Software*, Alastair Donaldson and David Parker (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 248–254.

[9] Byung-Gon Chun, Sunghwan Ihm, Petros Maniatis, and Mayur Naik. 2010. CloneCloud: Boosting Mobile Device Applications Through Cloud Clone Execution. *CoRR* abs/1009.3088 (2010).

[10] Eduardo Cuervo, Aruna Balasubramanian, Dae-ki Cho, Alec Wolman, Stefan Saroiu, Ranveer Chandra, and Paramvir Bahl. 2010. MAUI: Making Smartphones Last Longer with Code Offload. In *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services (MobiSys '10)*. ACM, 49–62.

[11] Vincenzo De Maio and Ivona Brandic. 2019. Multi-Objective Mobile Edge Provisioning in Small Cell Clouds. In *Proceedings of the 2019 ACM/SPEC International Conference on Performance Engineering (ICPE '19)*. ACM, New York, NY, USA, 12.

[12] Leonardo de Moura and Nikolaj Bjørner. 2008. Z3: An Efficient SMT Solver. In *Tools and Algorithms for the Construction and Analysis of Systems*, C. R. Ramakrishnan and Jakob Rehof (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg.

[13] G. e. m. Zhioua, H. Labiod, N. Tabbane, and S. Tabbane. 2014. VANET Inherent Capacity for Offloading Wireless Cellular Infrastructure: An Analytical Study. In *2014 6th International Conference on New Technologies, Mobility and Security (NTMS)*. 1–5. https://doi.org/10.1109/NTMS.2014.6814060

[14] Vijay Ganesh and David L Dill. 2007. A decision procedure for bit-vectors and arrays. In *International Conference on Computer Aided Verification*. Springer.

[15] Harshit Gupta, Amir Vahid Dastjerdi, Soumya K. Ghosh, and Rajkumar Buyya. 2016. iFogSim: A Toolkit for Modeling and Simulation of Resource Management Techniques in Internet of Things, Edge and Fog Computing Environments. *CoRR* abs/1606.02007 (2016).

[16] Nicolas Houy. 2014. The Bitcoin mining game. *Available at SSRN 2407834* (2014).

[17] C. Huang, M. Chiang, D. Dao, W. Su, S. Xu, and H. Zhou. 2018. V2V Data Offloading for Cellular Network Based on the Software Defined Network (SDN) Inside Mobile Edge Computing (MEC) Architecture. *IEEE Access* 6 (2018).

[18] G. Hurlburt. 2016. Might the Blockchain Outlive Bitcoin? *IT Professional* 18, 2 (Mar 2016), 12–16. https://doi.org/10.1109/MITP.2016.21

[19] E. Incerto, M. Tribastone, and C. Trubiani. 2016. Symbolic Performance Adaptation. In *2016 IEEE/ACM 11th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*. 140–150.

[20] J. Kang, R. Yu, X. Huang, S. Maharjan, Y. Zhang, and E. Hossain. 2017. Enabling Localized Peer-to-Peer Electricity Trading Among Plug-in Hybrid Electric Vehicles Using Consortium Blockchains. *IEEE Transactions on Industrial Informatics* 13, 6 (2017), 3154–3164.

[21] Jiawen Kang, Rong Yu, Xumin Huang, Maoqiang Wu, Sabita Maharjan, Shengli Xie, and Yan Zhang. 2018. Blockchain for secure and efficient data sharing in vehicular edge computing and networks. *IEEE Internet of Things Journal* (2018).

[22] Kuljeet Kaur, Sahil Garg, Georges Kaddoum, François Gagnon, and Syed Hassan Ahmed. 2019. Blockchain-based Lightweight Authentication Mechanism for Vehicular Fog Infrastructure. *arXiv preprint arXiv:1904.01168* (2019).

[23] Jung-Yoon Kim and Hyoung-Kee Choi. 2012. An Enhanced Security Protocol for VANET-Based Entertainment Services. *IEICE Transactions* 95-B (2012).

[24] Seungmo Kim. 2019. Impacts of Mobility on Performance of Blockchain in VANET. *IEEE Access* (2019).

[25] K. Kumar and Y. H. Lu. 2010. Cloud Computing for Mobile Users: Can Offloading Computation Save Energy? *Computer* 43, 4 (2010), 51–56.

[26] S Kwatirayo, J Almhana, and Z Liu. 2013. Adaptive traffic light control using VANET: A case study. *2013 9th International Wireless Communications and Mobile Computing Conference, IWCMC 2013*, 752–757. https://doi.org/10.1109/IWCMC.2013.6583651

[27] Benjamin Leiding, Parisa Memarmoshrefi, and Dieter Hogrefe. 2016. Self-managed and Blockchain-based Vehicular Ad-hoc Networks. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct (UbiComp '16)*. ACM, New York, NY, USA, 137–140.

[28] Bo Li, Yijian Pei, Hao Wu, Zhi Liu, and Haixia Liu. 2014. Computation Offloading Management for Vehicular Ad Hoc Cloud. In *Algorithms and Architectures for Parallel Processing*, Xian-he Sun, Wenyu Qu, Ivan Stojmenovic, Wanlei Zhou, Zhiyang Li, Hua Guo, Geyong Min, Tingting Yang, Yulei Wu, and Lei Liu (Eds.). Springer International Publishing, Cham, 728–739.

[29] L. Li, J. Liu, L. Cheng, S. Qiu, W. Wang, X. Zhang, and Z. Zhang. 2018. CreditCoin: A Privacy-Preserving Blockchain-Based Incentive Announcement Network for Communications of Smart Vehicles. *IEEE Transactions on Intelligent Transportation Systems* 19, 7 (2018), 2204–2220.

[30] H. Liu, Y. Zhang, and T. Yang. 2018. Blockchain-Enabled Security in Electric Vehicles Cloud and Edge Computing. *IEEE Network* 32, 3 (2018), 78–83.

[31] M. Liu, F. R. Yu, Y. Teng, V. C. M. Leung, and M. Song. 2018. Computation Offloading and Content Caching in Wireless Blockchain Networks With Mobile Edge Computing. *IEEE Transactions on Vehicular Technology* 67, 11 (2018).

[32] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y. FlÃűtterÃűd, R. Hilbrich, L. LÃijcken, J. Rummel, P. Wagner, and E. WieBner. 2018. Microscopic Traffic Simulation using SUMO. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. 2575–2582.

[33] Drazen Lucanin and Ivona Brandic. 2016. Pervasive Cloud Controller for Geotemporal Inputs. *IEEE Trans. Cloud Computing* 4, 2 (2016), 180–195.

[34] Francisco J Martinez, J-C Cano, Carlos T Calafate, and Pietro Manzoni. 2008. Citymob: a mobility model pattern generator for VANETs. In *ICC Workshops-2008 IEEE International Conference on Communications Workshops*. IEEE, 370–374.

[35] M. Singh and S. Kim. 2018. Trust Bit: Reward-based intelligent vehicle commination using blockchain paper. In *2018 IEEE 4th World Forum on Internet of Things (WF-IoT)*. 62–67.

[36] C. Sonmez, A. Ozgovde, and C. Ersoy. [n. d.]. EdgeCloudSim: An environment for performance evaluation of Edge Computing systems. In *2017 Second International Conference on Fog and Mobile Edge Computing (FMEC)*. 39–44.

[37] Z. Yang, K. Yang, L. Lei, K. Zheng, and V. C. M. Leung. 2019. Blockchain-Based Decentralized Trust Management in Vehicular Networks. *IEEE Internet of Things Journal* 6, 2 (April 2019), 1495–1505. https://doi.org/10.1109/JIOT.2018.2836144

[38] R. Yugapriya, P. Dhivya, M. M. Dhivya, and S. Kirubakaran. 2014. Adaptive traffic management with VANET in V to I communication using greedy forwarding algorithm. In *International Conference on Information Communication and Embedded Systems (ICICES2014)*. 1–6.