Querying Large Vehicular Networks: How to Balance On-Board Workload and Queries Response Time?

Romaric Duvignau[1], Bastian Havers[1,2], Vincenzo Gulisano[1] and Marina Papatriantafilou[1]

***Abstract*— Data analysis plays a key role in designing today's Intelligent Transportation Systems (ITS) and is expected to become even more important in the future. Connected vehicles, one of the main instantiations of ITS, produce large volumes of data that are difficult to gather by centralized analysis tools. The even larger volumes of data expected from autonomous driving will further exacerbate the bottleneck problem of data retrieval. When analysts issue queries that seek data from vehicles satisfying certain criteria (e.g. those driving above a certain speed or in a certain area), the problem can partially be overcome by pushing to vehicles themselves the job of checking and reporting the compliance of their local data (e.g. recorded GPS positions or CAN data), hence avoiding a costly data retrieval phase. The problem we tackle in this work consists in spreading a set of such queries over a vehicular fleet while balancing the time needed to resolve the queries and the computational overhead induced on the vehicular network. We present in this work efficient and configurable query-spreading algorithms tailored for vehicular networks. Our tunable algorithms, which we evaluate on two large sets of real-world vehicular data, outperform baseline solutions and are able to balance the trade-off between the overall on-board workload and the response time needed to receive all answers for a set of queries.**

I. INTRODUCTION

One of the key triggers for further deployment of Intelligent Transportation Systems (ITS) relies on their communication capacity and on an efficient use of their available bandwidth. For instance, in the automotive industry and in academic research, Vehicular Ad Hoc Networks (VANETs) [14] have been established in the recent decade as a key-enabling technology for providing a wide range of applications such as vehicle road safety, enhanced traffic, entertainment, infotainment, and improved comfort for drivers and passengers. A central component of many applications in vehicular networks is data gathering (see [24], [9] for applications and services in VANETs), i.e., the process of collecting sensed data from a fleet of vehicles to a central point (e.g. a company's data center). Since evolving vehicles can produce several gigabytes of data per hour [4], analysis of data sensed from a large fleet becomes practically infeasible if raw data is transferred to the analysis center. Hence there is a strong need to leverage processing power on-board the vehicles [11], [22], [23], [26], or use some form of compression mechanisms (e.g. [7]) in order to decrease the volume of data on the communication network (hence reducing monetary cost associated with the analysis) and utilize possibilities for continuous, stream processing [5].

[1]Authors are with Chalmers Univ. of Technology, Gothenburg, Sweden {duvignau,havers,vinmas,ptrianta}@chalmers.se.

[2]Bastian Havers is also with Volvo Cars, Gothenburg, Sweden bastian.havers@volvocars.com.

Two types of communication are most often discussed in association with VANETs: namely Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I). V2V relies on some form of short-range communication whereas V2I can be further subdivided into either communicating with *roadside units* [16] or to some *base stations* via mobile broadband. Short-range communication is a promising technology that has been greatly exploited to efficiently solve a wide range of information gathering in vehicular networks, but even though the technology is mature enough it is not yet readily available [28]. Cellular connectivity, on the other hand, is becoming standard on modern cars (now using 4G/LTE mobile infrastructure and soon 5G [8]) and all vehicles are projected to be equipped with it by 2020, while 5G connectivity is expected to be widely used in vehicles in the next couple of years. 5G will allow huge amounts of data to be transferred but at predictably higher monetary costs [28].

In this context, pushing the analysis on-board participating vehicles, thus leveraging their distributed computational power, can reduce communication overheads by avoiding central data gathering, which is of utmost importance for big data analysis to be feasible within large fleets of vehicles. This work challenges a recurring assumption in the literature about VANETs, differentiating them from other sensor networks [15], [20]: it is assumed that vehicles have "no strict limitation for processing power and storage capabilitities" contrary to other typical sensor networks made of small computing units. As more computationally heavy on-board tasks will be performed on *intelligent* vehicles and considerable amounts of data must be stored or transferred, we cannot assume any longer that those vehicles have unlimited computing power and storage space; moreover, the need for smart vehicles to run priority security services and applications [14] other than the analysis task, is an additional reason why we cannot assume unbounded computational power. Hence, solutions tailored to reducing computing resources will soon be of key importance.

Contribution: In this exploratory work, we present algorithms relying on broadband communication to spread a set of *queries*, a particular instance of pre-processing for big data analysis, on a fleet of vehicles. The aim in this context is to minimize the time and computational overhead needed to check whether the queries can be fulfilled. In our work, queries consist of performing some task locally on a subset of the fleet satisfying some criteria (e.g. having available data on the requested time interval, or restricted geographical position, speed, driving mode, etc.) and collecting to a centralized point a certain amount of answers from vehicles

matching the query's many criteria. For example, to perform some traffic flow analysis, one can retrieve 100 vehicles' average speed among those in the fleet driving above 50 km/h within a city center during rush hour. We present here algorithms that distribute a batch of such queries to vehicles in a fashion that balances the response time to resolve all queries and the total workload induced on the vehicular network. We perform an extensive evaluation on two large real-world sets of vehicular data that allows us to simulate both the time needed to resolve different realistic ad-hoc queries and the processing time needed on-board all participating vehicles. Our main contribution is a set of tuned algorithms that produce significantly improved trade-offs compared to baseline solutions.

The paper is organized as follows. Section II introduces the system model we are using and provides definitions for queries and the vehicular network. In Section III, different algorithms are provided for spreading a set of queries over a fleet of vehicles. Section IV is dedicated to a thorough evaluation of the proposed algorithms on real vehicular datasets. Related work is discussed in Section V, while Section VI presents our conclusions and future research directions.

II. SYSTEM MODEL AND PROBLEM DEFINITION

A. System Model

In this study we consider the following model: a *fleet* V of k vehicles, referred to also as nodes, is equipped with multiple sensors $S = \{\text{GPS}, \text{steer}, \text{break}, \dots\}$ where each sensor $s_i \in S$ records a timestamped (multi-valued) stream of data $s_i(v) = (t_0, x_0), (t_1, x_1), \dots$ for each vehicle $v \in V$. Furthermore, all vehicles are connected to a central entity C, thought of as a datacenter, via a two-way communication channel. From data analysts, C receives queries $q_1, \dots, q_r$, each focusing on some subset of the possible sensors for some time span of recorded data.

In more details, queries carry a specific *condition* that needs to be fulfilled by vehicles to participate in the analysis and every query specifies some number of *answers* (response to the query from distinct vehicles) that need to be collected to complete the analysis task. Hence, not every vehicle may answer every query because of lack of data or the data is found not suitable to answer that particular query; the required number of answers is meant to gather a sufficient amount of data from the vehicles to be meaningful for the analysis task at hand. In this context, we need a big enough data sample (e.g. for statistical significance or to decrease possible privacy leaks) but also small enough (because of the time to collect all the data [11], and network/CPU time). Once the query's condition has been checked locally on a vehicle, a yes/no answer is transmitted to C.

A query q in our model is represented by a pair (P, n) where P is a general condition ranging over the local sensed data and n is the minimum number of positive[1] answers to collect in order to complete q; we will use the notation $q.P$ for the condition and $q.n$ for the minimum number of answers required to complete query q. The condition P encodes which sensors are affected by the query and an overall condition for all concerned portions of streams to satisfy to be able to participate in answering the query. In the following: $S_q \subseteq S$ is a subset of sensors on which the query q is focusing (by default $S_q = S$); (t_{start}, t_{end}) are time bounds spanning q's time interval of interest such that only the portion of stream $\{(t, x) \in s_i \mid t_{start} \leq t \leq t_{end}\}$ for sensors $s_i \in S_q$ is examined (if non-specified, the full recorded data is considered); (τ, δ) defines a minimum sensor sampling frequency requiring at least τ units of time of data with a maximum δ units of time between consecutive records (imposing at least τ/δ records within the imposed timespan).

Example Query: To estimate traffic flow in a city center during the morning rush hour, one can ask to select $n = 100$ vehicles in a $k = 10,000$ vehicles fleet, with $P =$ "driven within area A with an average speed more than 50km/h between 8 and 9 o'clock with GPS measurements spaced by at most 5s from each other". In this query, $S_q = \{\text{GPS}\}$, $(t_{start}, t_{end}) = (8{:}00, 9{:}00)$, $(\tau, \delta) = (1\text{h}, 5\text{s})$, and A delimits a bounding box approximating the city center.

B. Problem Definition

The collection of a large enough number of answers provides an estimation of the fraction of vehicles that satisfy $q.P$ in the fleet. Such a query mechanism can be used as is in many scenarios, e.g. to estimate the proportion of electric cars driving close to a potential location for a new electrical charging station. Once C is aware of a positive answer, it can subsequently ask that vehicle to perform a query-dependent task. For instance, the expected outcome might be the result of an aggregate function computed locally over the portion of data that has been checked (e.g. compute the average speed) or the data itself. In this work, we do not consider the query post-treatment but rather concentrate on the dissemination part, i.e., securing a set of vehicles that will participate in the query analysis. In practice, we can also carry out simple aggregation tasks (min/max, mean, etc.) during $q.P$'s processing on the vehicle. This will entail minor changes in the time needed to process $q.P$ and the aggregated value can be transmitted along the vehicle's yes/no-answer to C.

We suppose C has no access to the vehicles' data other than through communication with them, so that the amount of work needed to test $q.P$ cannot be estimated before processing the query locally on the appropriate vehicle. Furthermore, C knows exactly how to contact all vehicles in the fleet. In the likely event that data is missing for the requested time period, it is equivalent to considering that the involved node does not satisfy $q.P$ and cannot answer the query. In a similar fashion, vehicles unwilling to participate in a query's task (for privacy or other reasons) are modelled by negative answers. For simplicity, the set of participants is considered constant here, hence we do not consider that new vehicles may join the network or that vehicles may leave within the time interval spent resolving a particular query (in our case study, the query processing time is low enough to entail a negligible churn of the vehicles during the execution of the

[1] "positive" or yes-answer implies that P holds locally.

algorithms). Finally, we ignore here the transmission time of the answer as we assume it to be constant and very small (one bit for a yes/no answer and other $\mathcal{O}(1)$ additional information as the vehicle id, the time it took for the processing, etc).

C. Performance metrics

We associate with each query two performance measures: (i) **Query (response) Time:** the elapsed time between receiving a query at C and having collected the appropriate number of positive answers at C; (ii) **Analysis Cost (or total workload):** the overall computing load on the vehicles defined as the sum of individual processing times of all nodes involved in processing the query.

Notice that satisfying the 2 measures at the same time is not trivial as they go in opposite directions. More concretely, query time is minimized by simply asking all vehicles in the fleet and ignoring answers after n positive answers are retrieved (requiring maximum resources) while the workload is balanced by asking 1 car at the time in a round-robin fashion (requiring maximum time). Furthermore, the amount of uncertainty in the model is challenging: each query requires different amounts of time per node that cannot be predicted[2]. For instance, on the one hand, a query of the form "has the vehicle driven within x meters from a parking lot between 12:00 and 18:00?" induces that positive answers may be sent as soon as a matching GPS record is found whereas negative answers need to scan first all records within the time interval. On the other hand, a large number of negative answers for q may be received at C much quicker than any positive answers due to missing on-board data.

III. QUERY SPREADING ALGORITHMS

The challenges described in the preceding section lead us to design different strategies to disseminate a set of queries over the fleet. We present here algorithms that select a subset of nodes among the k_q nodes satisfying P for a *single* query $q = (P, n)$ assuming $k_q \geq n$. For a set of queries $q_1, \ldots, q_r$, each query can be resolved by executing the procedures of this section at C, either in parallel or sequentially.

Without further assumptions on the distribution of nodes satisfying P, it is natural to randomly select nodes to question in a uniform way within the pool of nodes that have not been interrogated yet. However, other factors (as the number of queries currently running on the vehicle, local computation time so far, etc.) can be used to bias the selection process. We present in this section four algorithms focusing on different measures:

- BASELINE1 optimizes the time needed to answer q,
- BASELINE2 optimizes communication by interrogating new vehicles only when needed (hence no more than n positive answers are received),
- BALANCED-ALGO balances the two baseline algorithms in order to collect quickly n answers without inducing a too high load on the vehicular nodes, and
- FAIR-ALGO extends BALANCED-ALGO by prioritizing the least-used vehicles in the selection.

The *baseline* options introduced in this work are meant to benchmark the balanced algorithms against edge-cases (i.e., optimizing only one aspect) of the spectrum of possible trade-offs.

A. Simple Model Description

For the ease of presentation, we first briefly describe the first three algorithms assuming a synchronous model of query-checking so as to provide the intuition which is behind each presented algorithm. Let us assume in this section that no time is spent sending query messages from C and that nodes need a constant amount of time to check the property P, so that after a "round" of time has elapsed, C has received answers (yes/no) from all nodes that were interrogated during that round. In this simplified situation, only two aspects have to be considered in order to measure the performance of query-spreading procedures: (1) the number of rounds needed at C to receive n answers; (2) the number of nodes m that have checked their capability to answer q (which is equivalent to the total workload on the vehicles as each interrogated vehicle has spent 1 round checking the query).

1) BASELINE1 *(optimal time):* In order to optimize the processing time, all nodes have to be contacted upon q's arrival; this process then solves the query in a single round. Indeed, with any other algorithm that leaves at least one node v un-contacted, if $k_q = n$ and if P holds on v, then only $k_q - 1$ answers are received after one round and a second round of communication is required to get all answers. Following this simple procedure to obtain all needed answers leads nonetheless to a potentially huge effort on the nodes. In particular, the number of interrogated nodes is always k independently of k_q and n. Hence all nodes always participate in order to resolve q even though the number of required answers n might be relatively small.

2) BASELINE2 *(optimal load):* In the opposite direction of potential solutions to our problem, there is an algorithm focusing on the computational overhead induced on the nodes. To ensure that only the minimum number of nodes are being asked to process q, we should only question at most as many new nodes as the number of missing answers. Any algorithm satisfying such an assertion is associated with a minimal analysis cost as interrogating more nodes might lead to the reception of more than n answers, hence involving a higher than necessary analysis cost. The best algorithm in this category selects randomly as many nodes as possible by interrogating $m \leq n$ "new nodes" at each round when there are m missing answers. When n positive nodes are found, the procedure stops. The worst processing time is $k-n+1$ rounds where $(n-1)$ positive answers are collected in the first round and the last answer is obtained after questioning every other vehicle in the subsequent rounds. However, the average is

[2]One may consider here the time interval on which the query is focused as a good indicator of the amount of work needed to answer, but this would undoubtedly ignore that some properties can quickly be resolved (in case of missing data for instance) whereas some other properties might need more than linear time (e.g. when sorting is needed).

much lower (in the order of $\log_{k/(k-k_q)}(n)$ rounds) and the average number of interrogated nodes in $\mathcal{O}(n\log(n))$.

3) BALANCED-ALGO*:* BASELINE1 needs only one round but all nodes end up being questioned to process q, whereas BASELINE2 communicates with a minimum number of nodes but requires a longer time to collect all answers. We present here an efficient scheme to achieve a reduced computing load on the network within a few processing rounds. The main idea behind BALANCED-ALGO is to evaluate the proportion of vehicles that are able to answer positively to a query by measuring it through the execution of the algorithm. A first sample probability of $p = a/n$ is obtained by randomly questioning n vehicles and getting a positive answers; subsequently, $\ell = \alpha \cdot (n-a)/p$ vehicles are randomly interrogated so that if p was a good estimation, very few vehicles will remain to interrogate during next round (when $\alpha = 1$) and we repeat this process until all n answers have been collected. The parameter α allows us to depart from the estimated expected number $(n-a)/p$ of vehicles to interrogate to get the remaining answers, by sampling more or fewer vehicles. This parameter allows us to either shorten (when $\alpha > 1$) the average number of rounds needed to resolve q while potentially increasing the analysis cost, or on the contrary (when $\alpha < 1$) to slow down q's processing by being more prudent and avoiding questioning more vehicles than necessary (and by this way getting closer to receiving exactly n answers at the end). In the case that no positive answers have yet been received, we evaluate the probability of positive answers to $p = 1/(m+1)$ where m is the number of nodes interrogated so far.

Algorithm 1 BALANCED-ALGO

Input: parameters $\alpha, \beta > 0$ and query q with $n = q.n$
Output: n answers to query q

```
p ← 1                                   ▷ estimation of probability to answer yes
S ← ∅                                   ▷ set of interrogated vehicles
A ← ∅                                   ▷ set of answers collected
R ← ∅                                   ▷ set of collected positive answers
while |R| < n do
    ℓ ← ⌈α · (n − |R| − (|S| − |A|) · p) · p^{-1}⌉
    i ← 0
    while i < ℓ and |S| < k do
        v ← random(S)                   ▷ random vehicle excluding S
        send(q, v)                      ▷ send query q to vehicle v
        S ← S ∪ {v}
        i ← i + 1
    end while
    if |S| = k then
        return R                        ▷ set of possible vehicles exhausted
    end if
    repeat
        r ← receive()                   ▷ Block till receiving next answer
        A ← A ∪ {r}
        if positive(r) then
            R ← R ∪ {r}
        end if
    until |A|/|S| ≥ β
    p ← max{|R|/|A|, 1/(|A|+1)}
end while
return R
```

B. General Model

We now generalize the algorithms of the previous section to our general query-answering model, i.e., when query processing time is vehicle- and context-dependent. In our typical vehicular setting, we cannot in general assume bounds on the time a vehicle needs to process a query and on the communication network.

1) BASELINE1*:* optimizes the time needed to answer a single query q by interrogating all vehicles upon receiving it. In our general model, the query time then needed for q is the best possible and corresponds to the n-th fastest positive answer received at C. In contrast to that, the analysis cost is the highest possible, as all k vehicles have processed q.

2) BASELINE2*:* optimizes the number of interrogated vehicles (hence minimizing needed communication to spread queries) by questioning a new vehicle only if strictly needed. This is achieved by interrogating n vehicles and asking new vehicles to check P only upon receiving a negative answer. Since in our general model it is not guaranteed that vehicles will have similar answer times (and it is even expected the opposite), this algorithm does not necessary imply a minimum load on the network (i.e., it might be that questioning more vehicles that need a short processing time to answer will require less resources overall).

3) BALANCED-ALGO*:* Algorithm 1 is similar in essence to its round-based version described in § III-A but needs to be tuned to take into account that all nodes do not reply at the same time (after 1 round in the aforementioned version). To do so, we wait to receive a certain proportion β of answers over all interrogated vehicles before proceeding to the next batch of selection (and re-evaluating the probability to answer positively). When $\beta = 1$, the algorithm waits for the reception of all answers before continuing; setting a lower value for β allows us to take a decision without having to wait for the slowest vehicles. Another change is about taking into account vehicles that have not yet answered when a new iteration has started. Based on previous answers, we evaluate that a fraction of $(|S|-|A|)\cdot p$ will answer positively, where $|S|-|A|$ counts the number of queried vehicles that have not answered yet, so we remove those *expected* answers when calculating the number of remaining answers to collect.

4) FAIR-ALGO*:* We suggest a variation of the previous algorithm, differing on how we select vehicles once we know the number ℓ of vehicles to interrogate in the next batch. Instead of randomly selecting new nodes to interrogate, vehicles having low local workload are picked first in the selection phase. The main difference with Algorithm 1 is the function `random(S)` that selects a vehicle among the not yet interrogated ones. Instead of selecting randomly, vehicles are selected in the order of their lowest *local workload* measured as (1) number of simultaneous queries running on the vehicle and (2) reported local processing time since the start. Vehicles are for that purpose stored in an updatable priority queue where vehicle v's priority is updated upon sending a new query to v and receiving v's answer (containing v's local processing time).

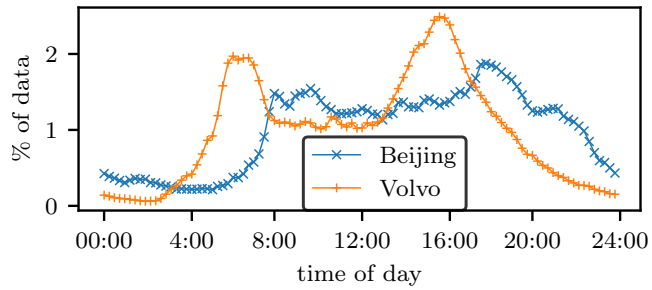


Fig. 1: Distribution of records in both datasets during 1 day.

IV. EVALUATION

To attest to the performance of the proposed algorithms, we evaluate them on two large real-world sets of vehicular data. In this section, we first describe the datasets and how the data has been pre-processed, then present a set of common queries that will serve to benchmark the different algorithms, and finally show the experiments' results.

A. Experimental Setup

1) Beijing Dataset: The first dataset consists of trajectories collected within the scope of the (Microsoft Research Asia) *Geolife* (version 1.3) project by 182 users over approximately four years [33]. The trajectories were collected from diverse users using different mobile devices and feature predominantly vehicular usage (by car, taxi, or bus). The original dataset consists of 18,670 GPS traces containing between 50 and 92,645 records of the form timestamp (s), latitude (deg), longitude (deg). After pre-processing the data, we used 9,772 files each for one daily usage of one user.

2) Volvo Dataset: The second dataset is made of CAN data and GPS traces from 20 hybrid cars collected in Sweden in the years 2014 and 2015 by *Volvo Cars Corporation* [11]. After pre-processing, we use 3246 traces corresponding each to a daily usage of one vehicle. Among the large quantity of CAN data, we have concentrated on two signals, the combustion engine rotation and electric engine rotation. These can be combined, leading to three possible driving modes: electric (*elec.*), combustion (*comb.*), hybrid (*hybr.*).

3) Simulation Setup: In our experiments, we will simulate a fleet consisting of 9772 (for Beijing) and 3246 (for Volvo) vehicles; each simulated vehicle will have one day of data stored on it. The distribution of data over a day for both datasets is depicted in Figure 1. To evaluate our algorithms, we define 15 queries to be run locally on the vehicles (presented in the following section). The queries are programs written in Python that are transferred to the vehicle via mobile broadband communication, then executed *on-board* the vehicle over their already stored data (1 day each). We note size(q) the amount of code and data[3] that needs to be transferred from C to each vehicle in order for the latter to be able to process q on-board. The elapsed time T (in milliseconds) needed between the coordinator sending a query message to a vehicle v and the reception of the corresponding answer is approximated as $T = T_l + \text{size}(q)/R + T_p(v, q)$ where T_l is a fixed round-trip latency for wireless communication, R is a constant standing for the wireless link data rate, and finally $T_p(v, q)$ is the time needed by the vehicle to decide if it can answer q or not; recall here, the transfer of the response message (yes/no answer and the small pieces of information about the vehicle) is neglected (cf. § II-B). In our set of experiments, we have set $T_l = 50$ms and $T_d = 10$Mb/s which are within current 4G/LTE latency and download rates (similar results are obtained using 5G parameters). To have a fair estimation $T_p(v, q)$, we have computed all queries on a vehicular processing unit representative [11]: an ODROID-XU3 single-board computer to approximate the low-power processor of a vehicle, equipped with a Samsung Exynos 5422 (Cortex-A15 2.1GHz Quad-Core and 1.4GHz Quad-Core CPUs) and 2 GB of LPDDR3 RAM at 933 MHz. We then use the computed time measured on the vehicular stand-in to simulate $T_p(v, q)$ for every possible vehicle v and query q. Based on the measured transfer time (through an Ethernet link with software-capped bandwidth to $T_d = 10$Mb/s), $\text{size}(q)/T_d$ expressed in ms is very well approximated by the size of data to transfer expressed in Kb.

The coordinator receives a certain number of queries in a random uniform order, and starts the batch of interrogations in the same order as the queries arrival's one. The queries are then answered in parallel by the vehicles and the coordinator reacts to each message arrival by either just updating its internal statistics for the corresponding query or by sending a new batch of interrogations to a new set of vehicles. If a vehicle v receives a new query task q' while already processing a previously received query q, then q' is added to v's local *task queue*; once v has terminated its processing with q, v's task queue is then processed in FIFO order. This approach simplifies the vehicles' internal computing architecture and is well suited in situations for which the remaining computing resources on-board the vehicles (if any) can be used to process security-sensitive applications.

B. Selected Queries

We introduce here a set of 15 queries, representative of possible vehicular analysis tasks. Table I presents (cf. § II for notations) the query q key (Q_1 to Q_{15}), the time interval $t_{start} - t_{end}$ given in hours, size(q) given in Kb, the description of the condition $q.P$, and the average answer rate (rounded to closest percentage) for Beijing (B%) and Volvo (V%) datasets. We use three possible values for (τ, δ): $\Delta_0 = (60, 5)$, $\Delta_1 = (45, 10)$ and $\Delta_2 = (80, 10)$, all values being in seconds.

Of the queries, 10 are run over both datasets whereas 5 additional queries focus on signals only contained within the *Volvo* dataset. The queries were chosen to represent different requirements (on time interval, queried sensors, geographic constraints, sampling constraints, etc), have distinct positive answer rates ranging from about 50% to about 1%, and finally give meaningful insights into the fleet's behavior. The parameters of the first 10 queries have been slightly

[3] e.g. GPS positions of stationary elements as parking lots or fuel stations.

Key	Time	Size (Kb)	Condition for *Beijing*	*Volvo*	B%	V%
Q_1	8-12	0.3	at least 1 record		55	47
Q_2	0-24	7.5 19.9	one parking within 50m	25m	43	
Q_3	17-18	1.3	Δ_0	Δ_1	28	
Q_4	0-24	0.5	Passed by City	Δ_2	19	
Q_5	17-18	1.1	Max speed $\geq$ 80km/h	68km/h	14	
Q_6	0-24	1.6	Speed $\geq$42km/h for 15min	22min	6	
Q_7	0-24	1.4	Driven in Downtown, Δ_0	Δ_2	5	
Q_8	17-18	0.8	Passed by City		3	2
Q_9	12-13	0.7	Stayed in City		1	
Q_{10}	0-24	3.8 7.7	*Stopped* at some Gas stations[4]		1	
Q_{11}	0-24	4.8	*comb.* (alone) used 50% of the time		15	
Q_{12}	0-24	5	Driven on *elec.* only outside City		13	
Q_{13}	0-24	4.4	Have used *hybr.* mode for 10min		10	
Q_{14}	0-24	6.6	*elec.* speed reached over 100km/h		7	
Q_{15}	0-24	4	3 diff. charging stations on *elec.*		4	

TABLE I: Selected query conditions and their parameters.

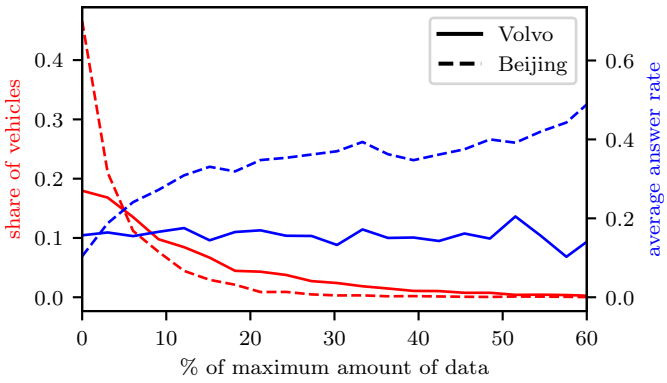


Fig. 2: Distribution of data volumes (left vertical axis) and average query answer rates (right vertical axis) for *Volvo* (solid) and *Beijing* (dashed line) dataset.

tuned between the two datasets (in Table I the *Volvo* column indicates differing parameter in the query's condition) so that each query in both datasets has a similar fraction of positive answers. Two geographical zones are defined for both datasets: *City* is the area of a large city chosen within the dataset and *Downtown* is a sub-area within *City* thought of as its heart. In our experiments, all queries require $n = 50$ answers to get resolved.

C. Experiments

1) Query answers rates: The average answer rates over all queries as well as the distribution of the data volume is presented in Figure 2 for the *Volvo* (solid) and *Beijing* dataset (dashed lines); 100% on the x-axis corresponds to the vehicle with the largest data volume among all vehicles in the respective dataset. For *Beijing*, the average query answer rate (blue, dashed) appears to be positively linked to the data volume; thus, vehicles with larger amount of data will have a higher chance to answer queries. For *Volvo* (blue, solid), the average query answer rate is almost flat, which indicates that vehicles with a large amount of data are roughly as

[4]In *Beijing* dataset, we require Δ_0 records within 50m of a fuel station, and in the *Volvo* dataset we require that the vehicle stopped (speed=0km/h) for 10min within 20m of a fuel station.

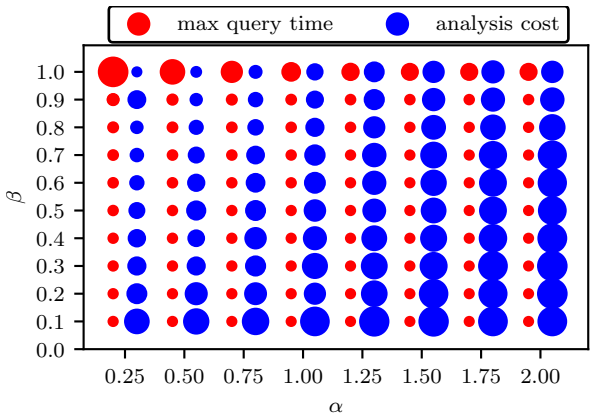


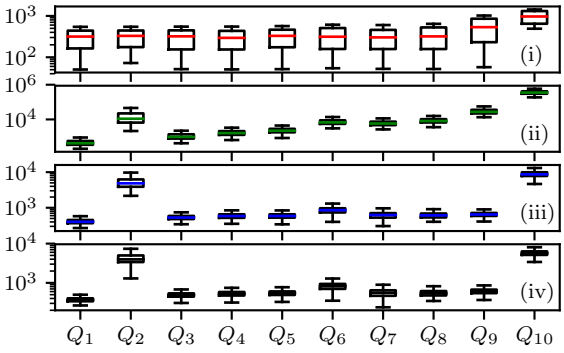
Fig. 3: Maximum query response time and analysis cost needed to resolve $Q_1 - Q_{10}$ over the *Beijing* dataset for BALANCED-ALGO for different α, β. Circle size scales with maximum time (red) and total workload (blue), respectively.

likely to answer a query as vehicles with only little data. Concerning the distribution of data volume among the fleet, *Beijing* dataset shows an exponential distribution whereas *Volvo* presents a logistic distribution with a long tail.

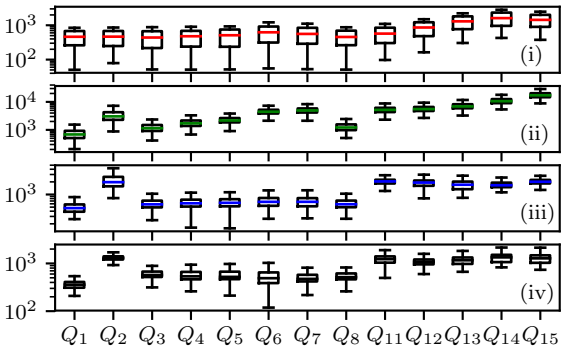
2) Parameter exploration α, β: To choose well-fitted parameters for our evaluation, we explore the parameter space for BALANCED-ALGO in the *Beijing* dataset. We run 1000 times the query set in our query simulator with different values for the parameter α (proportion of vehicles to ask, higher value translates to interrogating more vehicles) and β (fraction of vehicles to wait before sending next batch, higher fraction translates to longer waiting time between two query batches). For each run, we measure the time needed to answer all queries (i.e., the maximum query resolution time among the query set) and the analysis cost (total workload on the system measured as the sum of the processing time for each participating vehicle) and present them on a 2D plot in Figure 3 (note absolute values are given in Figure 4).

This exploration shows that for most values of β up to 0.9, the maximum time is similar (independently of α) but increases significantly for $\beta = 1$ due to waiting for the slowest vehicles to answer before asking a new set of vehicles (here α decreases the maximum query time when set to higher values). The total workload is, as expected, minimized when the algorithm asks few vehicles and waits for all answers before the next batch ($\alpha = 0.25$ and $\beta = 1$), whereas it increases if either β decreases (waiting fewer vehicles might lead to overestimate the proportion of negative answers) or α increases (questioning more vehicles will lead to receiving more than the required amount of answers and some vehicles might have thus processed unnecessarily the query). We set in the following $\alpha = 0.25$, $\beta = 0.9$ for BALANCED-ALGO and FAIR-ALGO, resulting in optimal trade-offs.

3) Comparison of the different algorithms: To give a general idea of the response time for the different queries, we present in Figure 4 the query response time measured over 1000 experiments consisting in resolving all 10/15 queries arriving in a random order; for the *Volvo* dataset, Q_9



(a) *Beijing* dataset



(b) *Volvo* dataset

Fig. 4: Query response time (in ms) for all valid queries executed over the (a) *Beijing* and (b) *Volvo* dataset for (i) BASELINE1, (ii) BASELINE2, (iii) BALANCED-ALGO, (iv) FAIR-ALGO.

and Q_{10} have been removed as all vehicles end up being asked (there are fewer than 50 positive answers in this case). The interesting facts to note are: BASELINE1's response time is short and varies the least among queries, BASELINE2 has the largest response times, clearly depending on the queries' answer rate (lower rate is associated with larger response times), BALANCED-ALGO and FAIR-ALGO present similar query times that do not vary significantly with the queries' answer rate. We also note that computationally heavier queries (Q_2, Q_{10}, and $Q_{11} - Q_{15}$) get resolved much slower than lightweight queries. As a summary, Figure 5 presents the average time (left side) and total work (right side) over all queries for the four algorithms relative to the average response time of BASELINE1 (over all queries) and the average workload of BASELINE2, respectively. The *average time* to answer queries is up to 2 orders of magnitude higher in BASELINE2 (in the *Beijing* dataset) compared to BASELINE1, whereas the proposed algorithms are about 1-4 times slower than BASELINE1; FAIR-ALGO performs really well in the *Volvo* dataset being close to the optimal solution. The *average analysis* cost is in the order of 3-5 times larger in BASELINE1 compared to BASELINE2. BALANCED-ALGO performs well on both datasets, having an average cost very close to the baseline. Finally, FAIR-ALGO's analysis cost is dependent on the distribution of data in the fleet: with skewed data (as in the *Beijing* dataset),

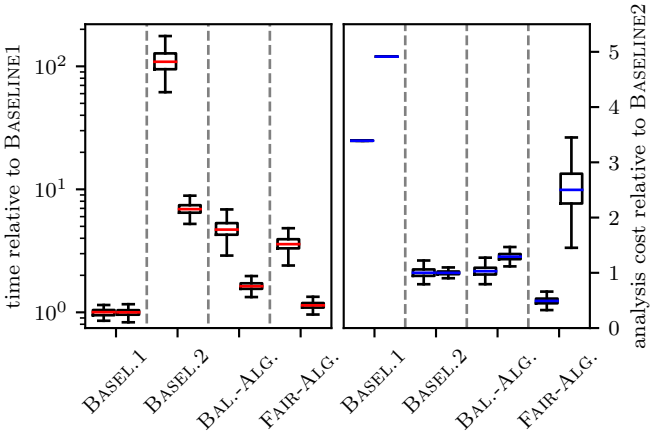


Fig. 5: The average relative time (left) and total work (right) between the four algorithms for the *Beijing* dataset (left boxplots in each column) and *Volvo* dataset (right boxplots).

it completely outperforms BALANCED-ALGO, whereas in a uniformly spread dataset (e.g. *Volvo* one) FAIR-ALGO presents a relatively poor choice (with performance varying between the two baseline algorithms).

4) Summary of the results: The presented solutions (well-tuned BALANCED-ALGO and FAIR-ALGO) provide large improvements in the trade-off of query response time versus on-board workload compared to baseline solutions. Furthermore, a query's response time in the proposed algorithms is not negatively impacted by a low positive answer rate among the fleet. The new solutions perform also better on different data distributions among the fleet: BALANCED-ALGO is more suited to a uniform distribution of positive answers (*Volvo* dataset) whereas FAIR-ALGO for a skewed distribution of positive answers (*Beijing* dataset).

V. RELATED WORK

The traditional approach to query a set of vehicles has been through SQL-inspired languages [30], [13], [31] to process continuous queries on live vehicular sensors' data. Two main differences with the current work are that in previous works (i) "queries" were usually initiated by vehicles themselves (see e.g. [19], [32], [6], [1]) and (ii) the full fleet was queried upon receiving new queries (as in [18]), contrary to our work where a known and fixed set of *general* queries is deployed from the coordinator to the fleet.

Query-answering mechanisms for vehicular networks in the literature also predominantly concentrate on using the architecture of the network (for instance using pre-existing P2P approaches, as in [21], [27], [17] or 2-tier architectures [25], [2]) to resolve the query. In this work, we do not presume any connections between vehicles; that positions our work in readily deployable technologies on modern vehicles.

In vehicle data analysis, privacy aspects are important when dealing with e.g. location-based services (see e.g. [29], [12], [3]). We suggest that our work, by allowing to check whether a certain number (chosen by the analyst) of vehicles

meets a given condition, can complement applications where privacy is supported by aggregating data from many sources.

VI. CONCLUSIONS

We presented algorithms able to spread a batch of queries over a set of vehicles, in a balanced fashion in terms of the time needed to resolve the queries and the on-board workload on the vehicles. We further presented a simulation environment, that we used to evaluate our algorithms, using real traces of accumulated data and realistic query processing times for vehicles. One future direction is to consider computationally heavier queries that necessitate longer processing times on the vehicles; this will require taking into consideration the churn rate of the vehicle set (since some vehicles might be turned off during the query resolution time). It will also be interesting to investigate the benefits of embedding V2V-related methodology, such as [10]. Finally, this work lays a first stone on the path to produce a complete on-board query simulation (producing simulated query answers and simulated query response times) that could be integrated with traffic simulations to be able to produce a fully simulated environment for vehicular data analysis benchmarking.

ACKNOWLEDGMENT

Work supported by VINNOVA, the Swedish Government Agency for Innovation Systems, proj. "Onboard/Offboard Distributed Data Analytics (OODIDA)" in the program FFI: Strategic Vehicle Research and Innovation (DNR 2016-04260); the Swedish Foundation for Strategic Research, proj. "Future factories in the cloud (FiC)" (grant GMT14-0032), and the Swedish Research Council (Vetenskapsrådet), proj. "HARE: Self-deploying and Adaptive Data Streaming Analytics in Fog Architectures" (grant 2016-03800).

REFERENCES

[1] G. M. N. Ali, E. Chan, and W. Li. Supporting real-time multiple data items query in multi-rsu vehicular ad hoc networks (vanets). *Journal of Systems and Software*, 86(8):2127–2142, 2013.

[2] C.-M. Cheng and S.-L. Tsao. Adaptive lookup protocol for two-tier vanet/p2p information retrieval services. *IEEE Trans. on Vehicular Technology*, 64(3):1051–1064, 2015.

[3] D. Christin, A. Reinhardt, S. S. Kanhere, and M. Hollick. A survey on privacy in mobile participatory sensing applications. *Journal of systems and software*, 84(11):1928–1946, 2011.

[4] R. Coppola and M. Morisio. Connected car: technologies, issues, future trends. *ACM Computing Surveys (CSUR)*, 49(3):46, 2016.

[5] S. Costache, V. Gulisano, and M. Papatriantafilou. Understanding the data-processing challenges in intelligent vehicular systems. In *2016 IEEE Intelligent Vehicles Symp.*, pages 611–618. IEEE, 2016.

[6] T. Delot, N. Mitton, S. Ilarri, and T. Hien. Decentralized pull-based information gathering in vehicular networks using geovanet. In *12th IEEE Int'l Conf. on Mobile Data Management*, volume 1, pages 174–183. IEEE, 2011.

[7] R. Duvignau, V. Gulisano, M. Papatriantafilou, and V. Savic. Streaming piecewise linear approximation for efficient data management in edge computing. In *34th ACM/SIGAPP Symposium On Applied Computing SAC'19*, pages 593–596, 2019.

[8] A. Fernandez and M. Fallgren. 5gcar scenarios, use cases, requirements and kpis. *Fifth Generation Communication Automotive Research and innovation, Tech. Rep. D*, 2, 2017.

[9] M. Gerla and L. Kleinrock. Vehicular networks and the future of the mobile internet. *Computer Networks*, 55(2):457–469, 2011.

[10] A. Gidenstam, B. Koldehofe, M. Papatriantafilou, and P. Tsigas. Scalable group communication supporting configurable levels of consistency. *Concurrency and Computation: Practice and Experience*, 25(5):649–671, 2013.

[11] B. Havers, R. Duvignau, H. Najdataei, V. Gulisano, A. C. Koppisetty, and M. Papatriantafilou. Driven: a framework for efficient data retrieval and clustering in vehicular networks. In *35th IEEE Int'l Conf. on Data Engineering (ICDE 2019)*, pages 1850–1861, 2019.

[12] X. Huang, R. Yu, J. Kang, and Y. Zhang. Distributed reputation management for secure and efficient vehicular edge computing and networks. *IEEE Access*, 5:25408–25420, 2017.

[13] B. Hull, V. Bychkovsky, Y. Zhang, K. Chen, M. Goraczko, A. Miu, E. Shih, H. Balakrishnan, and S. Madden. Cartel: a distributed mobile sensor computing system. In *4th Int'l Conf. on Embedded networked sensor systems*, pages 125–138. ACM, 2006.

[14] M. Kakkasageri and S. Manvi. Information management in vehicular ad hoc networks: A review. *Journal of network and computer Applications*, 39:334–350, 2014.

[15] F. Kong and J. Tan. A collaboration-based hybrid vehicular sensor network architecture. In *2008 Int'l Conf. on Information and Automation*, pages 584–589. IEEE, 2008.

[16] I. Ku, Y. Lu, M. Gerla, R. L. Gomes, F. Ongaro, E. Cerqueira, et al. Towards software-defined vanet: Architecture and services. In *Med-Hoc-Net*, pages 103–110, 2014.

[17] N. Kumar and J.-H. Lee. Peer-to-peer cooperative caching for data dissemination in urban vehicular communications. *IEEE Systems Journal*, 8(4):1136–1144, 2014.

[18] Y. Lai, L. Zheng, T. Wang, F. Yang, and Q. Zhou. Cloud-assisted data storage and query processing at vehicular ad-hoc sensor networks. In *Int'l Conf. on Security, Privacy and Anonymity in Computation, Communication and Storage*, pages 692–702. Springer, 2017.

[19] U. Lee, J. Lee, J.-S. Park, and M. Gerla. Fleanet: A virtual market place on vehicular networks. *IEEE Trans. on Vehicular Technology*, 59(1):344–355, 2010.

[20] U. Lee, E. Magistretti, M. Gerla, P. Bellavista, and A. Corradi. Dissemination and harvesting of urban data using vehicular sensing platforms. *IEEE Trans. on vehicular technology*, 58(2):882–901, 2009.

[21] C.-L. Liu, C.-Y. Wang, and H.-Y. Wei. Cross-layer mobile chord p2p protocol design for vanet. *Int'l Journal of Ad Hoc and Ubiquitous Computing*, 6(3):150–163, 2010.

[22] D. Palyvos-Giannas, V. Gulisano, and M. Papatriantafilou. Genealog: Fine-grained data streaming provenance at the edge. In *19th Int'l Middleware Conf.*, Middleware '18, pages 227–238, New York, NY, USA, 2018. ACM.

[23] D. Palyvos-Giannas, V. Gulisano, and M. Papatriantafilou. Haren: A framework for ad-hoc thread scheduling policies for data streaming applications. In *13th ACM Int'l Conf. on Distributed Event-Based Systems (DEBS)*, pages 19–30. ACM, 06 2019.

[24] E. Schoch, F. Kargl, M. Weber, and T. Leinmuller. Communication patterns in vanets. *IEEE Communications*, 46(11):119–125, 2008.

[25] S.-L. Tsao and C.-M. Cheng. Design and evaluation of a two-tier peer-to-peer traffic information system. *IEEE Communications*, 49(5):165–172, 2011.

[26] G. Ulm, E. Gustavsson, and M. Jirstrand. Oodida: On-board/off-board distributed data analytics for connected vehicles. *arXiv preprint arXiv:1902.00319*, 2019.

[27] T. Wang, L. Song, and Z. Han. Collaborative data dissemination in cognitive vanets with sensing-throughput tradeoff. In *1st IEEE Int'l Conf. on Communications in China (ICCC)*, pages 41–45. IEEE, 2012.

[28] K. Wevers and M. Lu. V2x communication for its-from ieee 802.11 p towards 5g. *IEEE 5G Tech Focus*, 1(2), 2017.

[29] J. Xu, Z. Jin, M. Xu, and N. Zheng. Mobile-aware anonymous peer selecting algorithm for enhancing privacy and connectivity in location-based service. In *7th Int'l Conf. on E-Business Engineering*, pages 172–177. IEEE, 2010.

[30] Y. Yao and J. Gehrke. The cougar approach to in-network query processing in sensor networks. *ACM Sigmod record*, 31(3):9–18, 2002.

[31] Y. Zhang, B. Hull, H. Balakrishnan, and S. Madden. Icedb: Intermittently-connected continuous query processing. In *23rd IEEE Int'l Conf. on Data Engineering (ICDE 2007)*, pages 166–175. IEEE, 2007.

[32] Y. Zhang, J. Zhao, and G. Cao. Roadcast: a popularity aware content sharing scheme in vanets. *ACM SIGMOBILE Mobile Computing and Communications Review*, 13(4):1–14, 2010.

[33] Y. Zheng, X. Xie, and W.-Y. Ma. Geolife: A collaborative social networking service among user, location and trajectory. *IEEE Data Eng. Bull.*, 33(2):32–39, 2010.