Pre-Proceedings of 8th International Workshop on Engineering Multi-Agent Systems
(EMAS 2020 @ AAMAS), 8-9 May 2020, Auckland, New Zealand.
Eds. Cristina Baroglio, Jomi F. Hübner, and Michael Winikoff.

Simulating Vehicular IoT Applications by Combining a Multi-agent System and Big Data

Ryo Neyama (✉)[1], Sylvain Lefebvre[1], Masanori Itoh[1], Yuji Yazawa[1,2], Akira Yoshioka[1,2], Jun Koreishi[3], Akihisa Yokoyama[2,1], Masahiro Tanaka[2], and Hiroko Okuyama[2,1]

[1] Toyota Motor Corporation
{neyama,slvn-lefebvre,masanori.itoh,yazawa,yoshioka}@toyota-tokyo.tech
[2] Toyota Research Institute - Advanced Development, Inc.
{akihisa.yokoyama,masahiro.tanaka,hiroko.okuyama}@tri-ad.global
[3] Sole proprietorship

Abstract. Describing an accurate simulation model of the driving behavior of real-world vehicles is a laborious or even impossible task, because a driver reacts to a dynamically changing environment. As multiple external factors determine driving behavior, it is usually difficult to obtain an accurate model, owing to a lack of sensors or inability to collect data. In this paper, we propose a novel technique to combine driving behavior in vehicular Internet of Things (IoT) big data with a multi-agent system. This enables correct and scalable simulation without modeling the behavior of vehicular IoT devices or the environment. We develop an extensible simulation framework, called FlowSim, that demonstrates the application of our technique for a simulation of camera-image data collection from connected cars.

1 Introduction

With the advocacy of the *CASE (Connected, Autonomous, Shared, and Electric)* vision by Daimler [4], connectivity based services for cars have been considered one of the key innovation enablers for automotive companies.

A developer of the next-generation services for connected cars may suffer from an insuperable gap between the car development cycle and the agility desired for making innovative services. For most car models, the development cycle is typically two to four years, and frequent and very large-scale software updates have not yet been achieved.

If one has a new service idea for connected cars in their mind, they would first test it in order to determine an appropriate specification, or to verify its business feasibility before production. For instance, considering a data collection service from connected cars to the cloud, developers may want to verify which data uploading algorithm is the most efficient, how long it takes, or how much it costs to collect a particular amount of data. Therefore, the research question is as follows: How can we verify services for millions of connected cars in an agile manner, such as in A/B testing [15]?

Multiple traffic simulation tools [16, 14, 13] or models [10, 12] are available in the literature, however, it is difficult to provide parameters to make the simulation realistic. To this end, we need to have hypotheses that are difficult to obtain for countrywide or continent-wide simulations. Namely, in the case of the aforementioned data collection service, we need to have hypotheses on when, where, and how many vehicles drive, to collect the data on the road under the influence of external factors, such as drivers' intentions, traffic signals, and traffic congestions. One solution to this issue could be to extract the parameters from real-world data. However, obtaining data about external factors is challenging, due to the high cost of embedding multiple sensors and processors in cars.

To overcome the aforementioned limitations, we propose a simulation technique to combine collected data from IoT vehicular devices with a multi-agent system. Our study makes the following contributions:

Requirements. There are some characteristic requirements in vehicular IoT simulations for the development of new services in the real world. We touch upon all the issues and categorize them into functional and non-functional requirements (Section 2).

Technique and Framework. We propose a technique to process highly scalable simulations by combining vehicular IoT big data with a multi-agent system, as well as a reusable simulation framework FlowSim to allow users to apply the technique to various vehicular IoT simulations (Section 3).

Case study. We apply the proposed technique to data collection for connected cars, and demonstrate some preliminary performance results, thereby demonstrating the usefulness of the technique and framework (Section 4).

2 Motivation

2.1 Use Case: Map-Generation Data Collection

In this study, we use an example of a connected-car service — the automated high-definition (HD) map generation and distribution by utilizing vehicular camera images [7, 3]. As depicted in Figure 1, a) the system collects on-board camera images of roads *(map-gen data)* from cars and stores them in a cloud-based service, b) the collected data are integrated using the service, thus generating an HD map, c) and delivering the HD map with a road network to automated driving systems or driver assistance systems.

Balancing between the cost and quality is a vital design decision in this service. While the coverage, quantity, and quality of the collected data influence the quality of the generated HD maps and the service itself, uploading high-quality map-gen data is expensive in communication and data center operations.

We consider the use of a simulator to verify the relationship between cost and quality (coverage, more specifically) during map-gen data collection. As discussed in Section 1, it is challenging to provide suitable parameters to the simulator. Therefore, we use vehicle log data containing only position, speed, and time stamp information, which are easily obtainable from current car models.

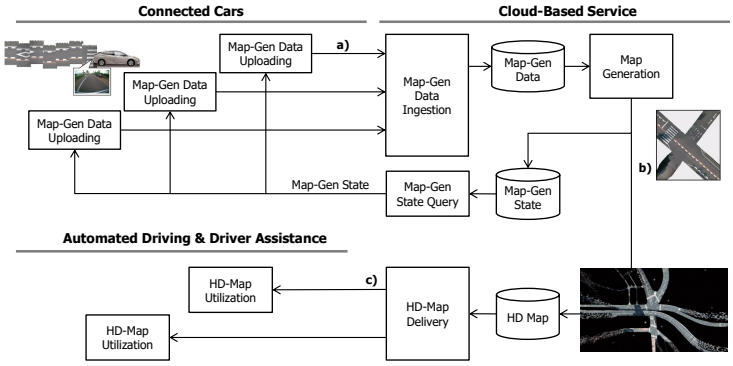

Fig. 1. Generation and Distribution of HD Maps

Here, we assume a road network denoted by an undirected graph $G = \{V, E, I, F\}$ where V is a set of vertices (intersections), $E \subseteq \{(x, y)|x, y \in V \wedge x \neq y)\}$ is a set of edges (road links), I is a pair of functions that map an edge to its unique identifier and vice versa, and F is a set of functions that map an edge to its features, such as a road link length.

We have a set of cars $V = \{v_i\}$. When a car v enters a road link with an ID l at a time stamp t, and requires d seconds to pass through the road link, we obtain a record $p_i = (v, l, t, d)$, thereby resulting in a set of records $P = \{p_i\}$, which forms the vehicle log.

Let $c_{t,v}$ be the communication cost needed to collect the data from a car v at a simulation time t, $len(\cdot), gen_t(\cdot) \in F$ the functions that, for a given road link (and time), provide the length, and determine whether an HD-map segment is generated, respectively, and $H_t = \{h_i|h_i \in E \wedge gen_t(h_i)\}$ represents the set of generated road links at t.

Our simulation goal is to evaluate two metrics, namely i) the total cost of communication in collecting the map-gen data and ii) the coverage of generated road links for the entire road network at a simulation time t_n. Formally:

$$C_n = \sum_{i=1}^{n} \sum_{j=1}^{|V|} c_{t_i, v_j} \qquad R_n = \frac{\sum_{i=1}^{|H_{t_n}|} len(h_i)}{\sum_{i=1}^{|E|} len(e_i)}$$

One can consider multiple data collection strategies, e.g. always uploading, randomly uploading, or filtering uploads by checking the status of each road link using the on-cloud service. C_n and R_n can vary depending on the data collection strategy $\mathsf{S_m}$.

2.2 Requirements

In the case of map-gen data collection, we have the following requirements:
Correctness. We create a business plan based on the simulation results. Therefore, the results obtained have to be sufficiently accurate. It is difficult to manually build an accurate model for simulation, as discussed in Section 1.

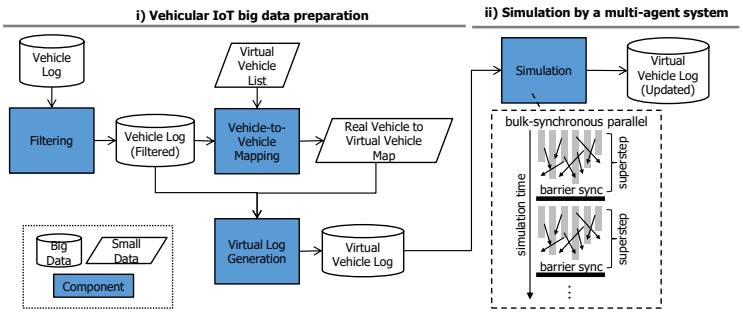

Fig. 2. Simulation Data Pipeline

Scalability. We create HD maps for a country or continent, and not solely for a block or city, which means that the system is required to scale up on the order of millions of cars.
Extensibility. We use various data collection strategies or simulation assumptions, e.g. communication method or in-vehicle storage size, as well as other applications, in addition to the map-gen data collection.

The *correctness* and *scalability* are in a trade-off relationship. The more we try to obtain accurate results, the greater the need to serialize task execution in the simulation, thereby rendering the system difficult to scale. In Section 3, we will discuss a method to balance the *correctness* and *scalability*.

3 Simulation by Combining a Multi-agent System with Vehicular IoT Big Data

3.1 Proposed Simulation Technique

To achieve both *correctness* and *scalability* in simulation, we propose a simulation technique that combines a multi-agent system and vehicle IoT big data. Figure 2 shows the simulation data pipeline, which comprises two parts:
i) Vehicular IoT big data preparation. We generate a virtual vehicle log by filtering the (real) vehicle log (P in Section 2.1), thus assigning real vehicles to the virtual ones. Virtual vehicle log records have one-to-one correspondence with the real vehicle log records: the system only offsets the records' time stamps and replaces the vehicle IDs. Therefore, *correctness* can be maintained without the need for modeling based on various hypotheses, as discussed in Section 1.
ii) Simulation by a multi-agent system. We carry out the simulation based on the bulk-synchronous parallel (BSP) model [19]. In each superstep of BSP, each agent thread runs concurrently for better *scalability*, while communicating with each other to make a right decision. This is eventually followed by a barrier synchronization at a point of sliced simulation time for better *correctness*. This balances the trade-offs mentioned in Section 2.2, according to the application's requirement.

As shown in Figure 3, the in-vehicle device and the cloud-based service work as agents, while IoT big data works as the environment in a multi-agent system.

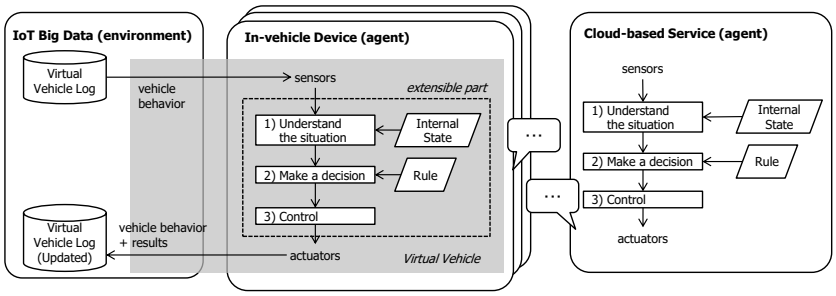

Fig. 3. Simulation by a Multi-agent System

The simulation regards the virtual vehicles' behaviors extracted from the data (p_i in Section 2.1) as the signals from the environment, and thereby simulation users can focus on modeling the in-vehicle device that provides the services (see the motivation in Section 2).

For example, in the map-gen data collection introduced in Section 2.1, the in-vehicle device 1) understands the characteristics of captured data (e.g. road type and map-gen data size) based on the behaviors, how long it can communicate with the cloud-based service for data uploading, and how much local storage is available. Then, 2) it ensures that there is enough free disk space to save the captured data, and exchanges request and response messages with the cloud-based service to determine whether it should upload the data on the local storage. Finally, 3) it stores the captured data to the local storage, and uploads the data on the local storage based on the available communication time.

3.2 The FlowSim Framework

We present a framework called FlowSim that is designed and implemented by generalizing the proposed technique, and we discuss the method for achieving *extensibility*. The components specified in Figure 2 are implemented as follows:
Vehicle-to-Vehicle Mapping. We map a set of virtual vehicles $U = \{u_j\}$ to real vehicles V to "copy" the behavior of a real vehicle v_i to one or more virtual vehicles $\{u_{j_1}, u_{j_2}, \ldots\}$, thus giving 1-to-$n$ correspondence.[4] The user can change the way virtual vehicles are generated along the simulation time and how they are mapped from a given list of real vehicles.
Virtual Log Generation. We generate a virtual vehicle log from the real vehicle log.[5] The users can modify or aggregate the vehicle log from the original one, according to the application's requirement.

[4] It is desirable that one real vehicle is assigned to only one virtual vehicle, from the viewpoint of *correctness*; otherwise, this can duplicate the behavior of the real vehicle in the simulation, which is unrealistic. To this end, a sufficient number of real vehicles are required, namely $|U| - |V| \ll |V|$ when $|U| > |V|$.

[5] In the system, we store the input data, intermediate data, and output data onto the Apache Hadoop [1] Distributed File System (HDFS), with two replicas. We use Apache Spark [2] to process data in a scalable way.

Simulation. The simulation repeatedly executes supersteps (see BSP in Section 3.1). We implement supersteps as sequential Spark [2] jobs. The virtual vehicle log is partitioned by (virtual) vehicle ID, and sorted by vehicle ID and time. Within a superstep, a set of Spark executors processes each partition in parallel. The partitioning scheme relies on the vehicle IDs. While sorting vehicles by ID and time ensures that events are processed in order for each vehicle, this guarantee does not hold across vehicles or partitions. This is mitigated by the choice of a suitable superstep length, and the barrier synchronization after each superstep execution.

As discussed in Section 2.1, we intend to evaluate various data collection strategies, i.e. $\mathsf{S}_1, \mathsf{S}_2, \ldots$. We can implement various algorithms to trigger events, such as capturing map-gen data based on vehicle log records. Also, we can provide a new strategy to decide whether the in-vehicle device uploads map-gen data, by requesting the cloud-based service for the map-gen state. In other words, it uploads map-gen data for that road link, if and only if no HD-map segment was generated for the road link. In our implementation, we use a key-value store, Hazelcast IMDG [6], for this purpose.

We can implement our own virtual data communication module, e.g. mobile network and Wi-Fi, as well as various network bandwidth models such as a uniform bandwidth or a bandwidth based on the Poisson distribution.

4 Case Study: Map-Gen Data Collection Simulation

4.1 Evaluation Environment

We used Scala (version 2.11.12) on top of Apache Spark 2.3.1 to develop the system. Log data and intermediate states were stored in HDFS (3.0.0) and Hazelcast IMDG (version 3.11.1), respectively. The Spark cluster spans 24 nodes (1 executor and 23 workers) over two racks and uses a single Hazelcast node. Machines are equipped with two Intel Xeon Gold 6126 24C CPUs and 384 GB of RAM along with 2.9 TiB of NVMe-SSD storage. The nodes are connected through a 100 Gb Ethernet network. Every machine runs Ubuntu 18.04 with kernel v4.15.0, and we use Docker (version 18.09.0) to deploy the software stack.

4.2 Evaluation with Map-gen Data Collection

We applied FlowSim to the simulation of map-gen data collection and confirmed that the proposed technique and FlowSim work as expected. Table 1 summarizes the information about the input data (the real vehicle log), settings, and results of the simulation. We used the input data described in Section 2.1. By counting the number of collected map-gen data for each road link, we could calculate the cost C_n and the coverage R_n under each data collection strategy $\mathsf{S}_1, \mathsf{S}_2, \ldots$ (see Section 2.1).

Figure 4 shows the execution time and the speedup (the right axis) of the simulation. Despite the large input data volume, the large number of real and

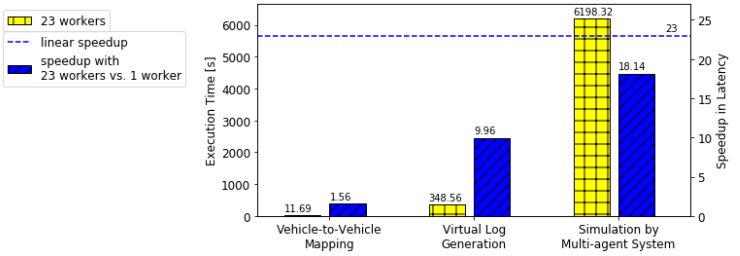

Fig. 4. System Performance for Each Simulation Step

Table 1. Map-Gen Data Collection Simulation

Item		**Description**
Input data	Data acquisition area	Japan
	Data acquisition period	365 days
	Data size (w/o replicas)	44.2 GB
	Number of data records	2,608,986,929
	Number of unique vehicles	414,329
Simulation settings	Number of vehicles (fixed)	1,000,000
	Simulation period	365 days
	Micro-batch interval	5 days
Simulation results	Number of captured map-gen data on the vehicles	2,535,104,457
	Number of collected map-gen data on the cloud-based service	29,520,776

virtual vehicles, and the long simulation period, the simulation can run in a reasonable time (1 h 49 m 19 s). The speedup indicates how much the system could improve the latency with 23 workers vs. 1 worker, namely 23 in linear speedup. Although the system showed poor speedup (1.56 and 9.96) for i) Vehicular IoT big data preparation (i.e. Vehicle-to-Vehicle Mapping and Virtual Log Generation) mentioned in Section 3.1, these steps account for only a small percentage (5.5 %) of the total execution time. In contrast, the system showed excellent speedup (18.14) for ii) Simulation by a multi-agent system, which accounts for most of the execution time (94.5 %).

5 Related Work

In [9] Ahmed et al. provide a survey of existing vehicular cloud computing simulation frameworks. They highlight the need for modularity and bidirectionality to integrate mobility simulators and network simulators. While this survey notes that the majority of studies rely on the SUMO [16] mobility generator, there also exist multiple approaches for extending multi-agent mobility simulators, such as MATSim, see [14, 17].

The common strategy for integrating traffic/mobility simulators and higher level agent simulation [17] or network simulation [18] is to build a software

bridge between the two simulators. This relies either on log files or sockets, or the simulation software API. For example, the Veins [18] framework uses bi-directional coupling between the network simulator Omnet++ [20] and SUMO. This coupling enables authors to observe the influence of vehicle-to-vehicle communication protocols on the simulated traffic.

The main limit of these approaches is that careful synchronization is needed between the two simulators to maintain correctness and realistic simulation. We believe this can hinder the scalability of these approaches. Therefore we use a trace-based approach to generate events, which trades off flexibility for scalability.

For example, Ahlbrecht et al. [8] implemented the Jason multi-agent framework on top of Apache Spark [2], demonstrating the efficiency of this approach by running a simulation of up to one million agents on a single node. While this approach yields important performance and scalability benefits by leveraging in-memory data processing frameworks to implement multi-agent simulation, the user has to provide an accurate simulation model.

In [11] Blythe et al. built a multi-agent simulation to predict usage on the GitHub platform [5]. They modeled the behavior of agents with a graph link prediction algorithm, trained using historical usage data. Although this study highlights the need to combine large agent simulation with big data mining and analysis tools, to provide better modeling of global services, extracting the nature of the original data is challenging when we apply this approach to vehicular IoT big data.

6 Conclusions and Future Work

Traffic simulation is one of the most dependable ways to verify new services for connected cars in an agile manner. However, creating a correct behavior model for vehicles is difficult, or even impossible, when we consider countrywide or continent-wide simulation.

We classify the system requirements into correctness, scalability, and extensibility. To this end, we propose a new simulation technique that combines a multi-agent system, IoT big data, and a framework, FlowSim that allows us to run simulations for various applications. We verified the usefulness of the proposed technique and FlowSim through a case study with map-gen data collection.

We leave the comparison with another agent based simulation or vehicular network simulator and the application of FlowSim to other applications, such as a dynamic map platform evaluation, for future work. Although our technique is scalable and can maintain correctness, we cannot simulate an anomaly or adversarial behavior that the original data set does not contain, because we use the original data set without modification. For the same reason, the behavior of agents cannot diverge from the information contained in the data, which limits the flexibility of our approach.

References

1. Apache Hadoop. https://hadoop.apache.org/
2. Apache Spark. https://spark.apache.org/
3. Automated Mapping Platform — HD Mapping that Empowers. https://www.tri-ad.global/areas-of-focus/automated-mapping-platform
4. CASE — Intuitive Mobility. https://www.daimler.com/innovation/case-2.html
5. GitHub (a software development platform). https://github.com/
6. Hazelcast IMDG (In-memory Data Grid). https://hazelcast.org/
7. Toyota to Display New Map Generation System at CES 2016. https://global.toyota/en/detail/10765074
8. Ahlbrecht, T., Dix, J., Fiekas, N.: Scalable multi-agent simulation based on mapreduce. In: Multi-Agent Systems and Agreement Technologies, pp. 364–371. Springer (2016)
9. Ahmed, B., Malik, A.W., Hafeez, T., Ahmed, N.: Services and simulation frameworks for vehicular cloud computing: a contemporary survey. EURASIP Journal on Wireless Communications and Networking **2019**(1), 4 (2019)
10. Bham, G.H., Benekohal, R.F.: A high fidelity traffic simulation model based on cellular automata and car-following concepts. Transportation Research Part C: Emerging Technologies **12**(1), 1–32 (2004)
11. Blythe, J., Bollenbacher, J., Huang, D., Hui, P.M., Krohn, R., Pacheco, D., Muric, G., Sapienza, A., Tregubov, A., Ahn, Y.Y., Flammini, A., Lerman, K., Menczer, F., Weninger, T., Ferrara, E.: Massive multi-agent data-driven simulations of the github ecosystem. In: Demazeau, Y., Matson, E., Corchado, J.M., De la Prieta, F. (eds.) Advances in Practical Applications of Survivable Agents and Multi-Agent Systems: The PAAMS Collection. pp. 3–15. Springer International Publishing, Cham (2019)
12. Burghout, W., Koutsopoulos, H.N., Andreasson, I.: A discrete-event mesoscopic traffic simulation model for hybrid traffic simulation. In: 2006 IEEE Intelligent Transportation Systems Conference. pp. 1102–1107. IEEE (2006)
13. Düntgen, C., Behr, T., Güting, R.H.: Berlinmod: a benchmark for moving object databases. The VLDB Journal **18**(6), 1335 (2009)
14. Horni, A., Nagel, K., Axhausen, K.W.: The multi-agent transport simulation MATSim. Ubiquity Press London (2016)
15. Kohavi, R., Longbotham, R.: Online controlled experiments and a/b testing. Encyclopedia of machine learning and data mining **7**(8), 922–929 (2017)
16. Lopez, P.A., Behrisch, M., Bieker-Walz, L., Erdmann, J., Flötteröd, Y.P., Hilbrich, R., Lücken, L., Rummel, J., Wagner, P., WieBner, E.: Microscopic traffic simulation using sumo. In: 2018 21st International Conference on Intelligent Transportation Systems (ITSC). pp. 2575–2582. IEEE (2018)
17. Padgham, L., Nagel, K., Singh, D., Chen, Q.: Integrating bdi agents into a matsim simulation. In: Proceedings of the Twenty-First European Conference on Artificial Intelligence. pp. 681–686. ECAI'14, IOS Press, NLD (2014)
18. Sommer, C., German, R., Dressler, F.: Bidirectionally coupled network and road traffic simulation for improved ivc analysis. IEEE Transactions on mobile computing **10**(1), 3–15 (2010)
19. Valiant, L.G.: A bridging model for parallel computation. Communications of the ACM **33**(8), 103–111 (1990)
20. Varga, A.: Discrete event simulation system. In: Proc. of the European Simulation Multiconference (ESM' 2001). pp. 1–7 (2001)