IsV2C: An Integrated Road Traffic-Network-Cloud Simulator for V2C Connected Car Services

Heejae Kim*, Jiyong Han*, Seong-Hwan Kim, Jisoo Choi, Dongsik Yoon†, Minsu Jeon, Eunjoo Yang, Nhat Pham, Sungpil Woo, Jeongkyu Park‡, Daeyoung Kim§ and Chan-Hyun Youn§

School of Electrical Engineering and School of Computing, Korea Advanced Institute of Science and Technology, Daejeon, Korea, e-mail: {kim881019, jyhanzz, s.h_kim, jisoochoi, msjeon, yejyang, nhatphd, woosungpil, kimd, chyoun}@kaist.ac.kr

†The 8th R&D Institute, Agency for Defense Development, Daejeon, Korea, e-mail: dsyoun@add.re.kr

‡Big Data Planning Team, KIA Motors Corporation, Seoul, Korea, e-mail: jk.park@kia.com

***Abstract*—The recent growth of connected car technology encourages IT and vehicle organizations to develop advanced vehicle-to-cloud (V2C) services such as driving assistance, infotainment, and vehicle maintenance. However, since their performances highly depend on the resource configuration and regional characteristics, it is almost impossible to examine the service feasibility in every candidate region by constructing real testbeds. To overcome this problem, we present an integrated road traffic-network-cloud simulator for V2C connected car services (IsV2C) with a user-friendly GUI. The IsV2C aims to evaluate V2C services with the user-specified V2C environment and service scenarios in a particular region. In the IsV2C, road traffic, network, and cloud simulation are intimately linked to reflect both the realistic movement of vehicles and service transactions in real-time. To accurately mimic task execution in both vehicles and the cloud, the IsV2C utilizes a rigorous emulation for evaluated services. Simulation results of the IsV2C show whether each simulated application satisfies the service level objectives regarding service time, cloud cost, and data transmission performance. As for the validation, we evaluated three sample V2C applications in an urban area, and the results proved that the IsV2C could offer useful information to both service providers and cloud providers for their service launching and profit estimation. To the best of our knowledge, the IsV2C is the first work that presents an integrated road traffic-network-cloud simulation framework for an end-to-end V2C service evaluation.**

Keywords-connected cars; end-to-end simulation; integrated simulator; service feasibility; vehicle-to-cloud services;

I. INTRODUCTION

The connected car which refers a vehicle providing network connectivity is the latest trend in the automotive industry. This new concept of vehicles has received much attention all over the world in recent years. According to Gartner,[1] connected cars are forecast to reach 25 billion by 2020, and the value will have a five-fold increase in comparison to 2015. The significant increase of this popularity encourages automotive and IT organizations to develop advanced connected car services such as driving assistance[2,3], infotainment[4,5], and vehicle maintenance[6,7]. Moreover, the assistance of a central cloud has been providing an opportunity to mount more functionalities to the connected car services [1,2,3]. It is because the central cloud enables to 1) overcome the limitation of computation performance and storage size in vehicles and 2) provide a service that is hard to implement in a single vehicle such as local dynamic maps (LDMs) [4]. As part of the recent trend, Fiat Chrysler Automobiles and Hyundai Motor Company are collaborating with Google for their connected cars, and Microsoft and Toyota announced their partnership for Azure-based Toyota Big Data Center.[8,9] Also, BMW built CARASSO which offers dynamically updated map information using sensor data from vehicles leveraging Amazon Web Services.[10]

However, it is still a big challenge to examine the feasibility of vehicle-to-cloud (V2C) services. In general, cloud-based services are complexly constructed, and their execution is often distributed in various ways. Therefore, their performances highly depend on the resource management approach and the processing capacity. Even worse, local conditions such as the number of vehicles, network infrastructure, and the data traffic size in service coverage areas significantly affect the performance of the services. For this reason, it is nearly impossible to evaluate V2C services in every candidate region by constructing real testbeds.

A simulation is one of the attractive alternatives to overcome the problem since it can provide a variety of evaluations with any resource configurations and regional settings. There have been several works to develop road traffic simulators [5], network simulators [6,7], and cloud simulators [8,9,10,11,12,13]. However, the independent use of these simulators has the limitation to model a V2C environment in a given region completely. Although the existing mobile cloud simulators [14,15,16,17] provide an effective way to simulate task execution with resource-constrained devices and the cloud, they are still not enough to evaluate the end-to-end performance between vehicles and the cloud because their modeling of network and task execution is rather simplified. As for a cellular network, the linkage of a road traffic simulation is required for realistic vehicle mobility scenarios which consider cellular performance factors such as handover trials of user equipment (UE) and the distance to base stations [18,19,20]. The integration of road traffic and network simulations have also been introduced in several works [21,22,23]; however, these are not suitable for V2C services when focused on the

*These authors contributed equally to this work.

§Corresponding authors

[1]http://www.gartner.com/newsroom/id/2970017

[2]https://waymo.com

[3]http://drivenet.pilotlab.co

[4]https://www.harman.com/connected-car

[5]https://www.nuance.com/mobile/automotive/dragon-drive.html

[6]https://www.kaaproject.org/automotive

[7]https://pivotal.io/industries/automotive

[8]https://www.mediapost.com/publications/article/292155

[9]https://news.microsoft.com/2017/03/22

[10]https://aws.amazon.com/ko/solutions/case-studies/bmw

TABLE I. THE CLASSIFICATION OF V2C CONNECTED CAR SERVICES

Service Classification			Example	IsV2C Support
Request Pattern	One-to-one service		Intelligent personal assistants, vehicle maintenance	Supported
	One-to-many service	Async.	LDMs, crowdsourced services	Supported
		Sync.		Not Supported
Repeatability	Non-repetitive service		Intelligent personal assistants, emergency handling	Partially Supported
	Repetitive service		LDMs, sensor data analytics , self-driving	Supported
Offloading Availability	Offloadable service		All except for non-offloadable services	Supported
	Offloading-mandatory service		LDMs, most crowdsourced services	Supported

vehicle-to-vehicle (V2V) communication.

In this paper, we present the IsV2C, an integrated road traffic-network-cloud simulator for V2C connected car services. The goal of the IsV2C is to evaluate the performance and the cost of a V2C service provision under the V2C environment in a particular region. The IsV2C allows both service providers and cloud providers to pre-test their service feasibilities in a given area and to estimate their profits, respectively. To the best of our knowledge, the IsV2C is the first work that presents an integrated road traffic-network-cloud simulation framework for an end-to-end V2C service evaluation.

The main contributions of the IsV2C are as follows.

- The IsV2C allows simulation users to evaluate V2C services with a user-friendly GUI and provides a highly visible summary of simulation results.
- The IsV2C executes tightly coupled road traffic, network, and cloud simulation. The interoperability contributes a better insight into the end-to-end performance evaluation of V2C services in a given region.
- The IsV2C provides a reliable way to mimic computational behaviors of tasks in both vehicles and the cloud via an emulation. With a real cloud testbed, the emulation enables to overcome the difficulty in the accurate computational modeling of V2C services.
- The IsV2C provides evaluation results for service feasibility with various service level objectives regarding performance and cost. The metrics involve service time, cloud cost, and data transmission performance.

For the validation of the IsV2C, we evaluated three sample V2C applications in an urban area with various simulation scenarios.

The remainder of this paper is organized as follows. Section II and Section III describe the fundamentals and the design of the IsV2C. In Section IV, we present the validation of the IsV2C. Finally, we conclude the paper in Section V.

II. ISV2C FUNDAMENTALS

A. V2C Connected Car Services

As summarized in Table I, we have classified V2C connected car services by three criteria: request pattern, repeatability, and offloading availability. According to the request pattern, V2C connected car services are divided into one-to-one and one-to-many services. One-to-one services are only based on the direct communication between a single vehicle and the cloud, while one-to-many services such as LDMs and a variety of crowdsourced services usually require input data from multiple vehicles. One-to-many services run either asynchronously or synchronously. Another criterion is repeatability. In the case of repetitive services, they usually require periodic data collection. On the other hand, non-repetitive services are executed when service users request or when in an emergency. Finally, the other criterion is the availability of the service offloading to the cloud. There exist jobs that should not be offloaded in the case of privacy or security related services. On the contrary, the offloading can be much advantageous to some services which require high computing resources or storage capacity. Also, for particular services which need dynamically-updated external data (e.g., LDM), the offloading is mandatory. The IsV2C supports all the service types except for synchronous one-to-many services and non-repetitive services affected by external factors such as traffic accidents and rockslides.

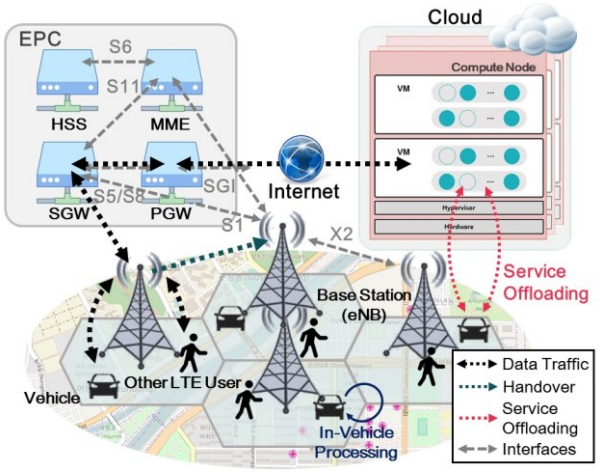


Figure 1. The simulated environment of the IsV2C.

B. Simulated Environment

Fig. 1 illustrates the simulated environment in the IsV2C. The V2C environment is composed of vehicles, base stations, and a cloud. In the environment, the LTE is mainly used as a cellular network. Vehicles are connected to the corresponding base stations (i.e., eNBs), and LTE handovers occur between base stations by the evolved packet core (EPC) if a signal strength is low. The EPC consists of the serving gateway (SGW), the packet data network gateway (PGW), the mobility management entity (MME), and the

home subscriber server (HSS). Data packets are transmitted to the Internet via the two gateways and vice versa.

In the simulation area, vehicles may provide several V2C services simultaneously. Basically, V2C services follow a client-server model. They are coordinated by the service provider leveraging virtual machines (VMs) from the cloud provider. If a service is not offloadable or decided not to offload, it is executed inside vehicles. Otherwise, the service is offloaded to the VMs. For more practical scenarios, the IsV2C considers not only vehicles but also other LTE users —non-vehicle LTE users (e.g., pedestrians)— as UEs.

C. Design Principle

User convenience: The IsV2C aims to be easy to use even if simulation users are not an expert in programming. Via a user-friendly GUI in simulation setup, the users can easily construct own V2C environment and service scenarios in a particular region. After simulation, the IsV2C summarizes and illustrates simulation results in a graphical representation to improve readability. Also, the users can utilize the IsV2C anywhere with the Internet because it is built as a RESTful web application.

Reliable simulation: For the accurate evaluation of V2C services, the IsV2C mimics the entire process of each simulated application with realistic service scenarios in a particular region. Once the area is given, the IsV2C models the movement of vehicles in the area and generates a vehicular trace. Based on the vehicular trace, the applications are simulated along a service path for each vehicle. To achieve an end-to-end performance evaluation, the IsV2C conducts organically linked road traffic, network, and cloud simulation. It enables to imitate every service transaction reliably. Also, the IsV2C takes an emulation-based approach to identify the performance of task execution in both vehicles and the cloud. The emulation is conducted by executing the applications in the same way with simulation scenarios to ensure a high accuracy.

Effective identification of the service feasibility: The IsV2C provides simulation results for each simulated application with a variety of service level objectives about service time, cloud cost, and data transmission performance. Firstly, the results show the average service completion time composed of data transmission and task execution time for each vehicle and each trial. Secondly, the IsV2C also provides an analysis of the cloud provider's profit derived as *cloud usage cost - compute node operating cost – service level agreement (SLA) penalty cost*. Finally, performance of a cellular network and end-to-end data transmission are illustrated in the results, for each vehicle and each data flow. Therefore, the simulation users can easily figure out the service feasibility from diverse perspectives.

III. IsV2C Design

A. Overview

The IsV2C operates its simulation in five steps as depicted in Fig. 2.

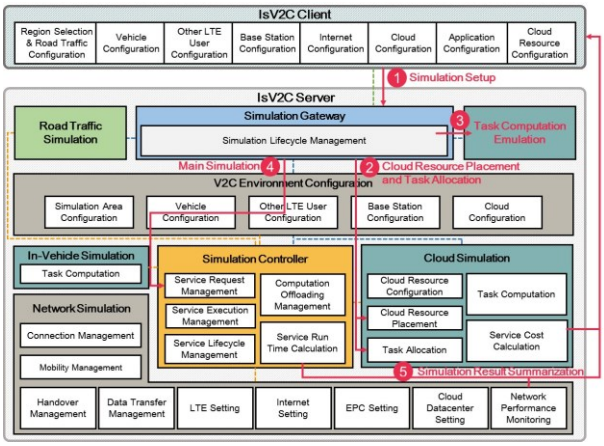


Figure 2. The architecture and simulation steps of the IsV2C.

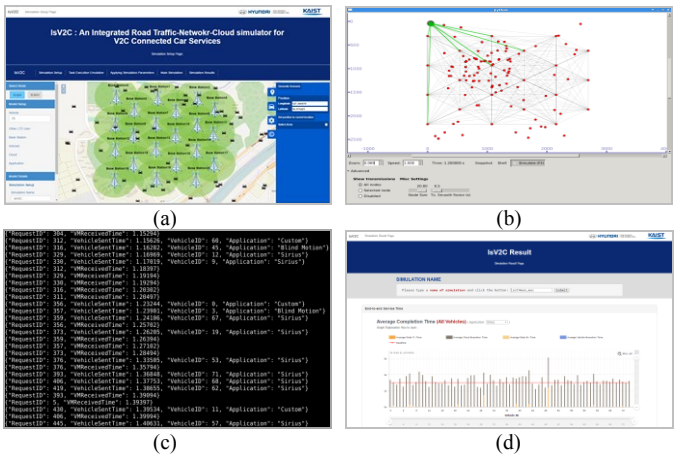


Figure 3. The IsV2C GUI. (a) The web interface for a simulation setup; (b) The visualization of simulation process; (c) Simulation logs; (d) The result webpage.

Simulation setup: The IsV2C provides a user-friendly web interface for a simulation setup. As shown in Fig. 3(a), the right sidebar implemented using SUMO [5] enables simulation users to select a region to be simulated and to generate a vehicle mobility scenario in the area. Designated longitude and latitude determine the region, and it is displayed in OpenStreetMap[11] at the center of the web page. The left sidebar is used for the parameter setting. Providing considerable autonomy to users, the IsV2C allows them to manipulate $10x$+30 parameters where x is the number of applications to be simulated.[12] The parameters are related to simulation lifecycle, vehicles, other LTE users (i.e., non-vehicle LTE users), base stations, the Internet, the cloud datacenter, applications, and cloud resource configurations of each application. Based on the parameter setting, all the simulation entities can be deployed in the simulated area as the user designates.

Cloud resource placement and task allocation: Based on the cloud resource placement and the task allocation strategy in the simulation setup, the IsV2C determines the number of co-located VMs in each compute node and co-allocated tasks in each VM, respectively.

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

[12]The detailed list of the simulation parameters is available on our website (http://ncl.kaist.ac.kr/wp-content/uploads/2017/02/supple.pdf).

An emulation of task execution: Based on the resource assignment results by the cloud resource placement and the task allocation, the task execution of each simulated application is emulated by its given behaviors using a real cloud testbed. The emulation results are utilized to determine the task execution time in the main simulation.

Main simulation: The simulation controller conducts the whole process of the main simulation. The main simulation starts with the creation of the simulated environment and the vehicular trace in the given region. An end-to-end simulation is then carried out for the applications. The simulation controller handles the whole service steps from service requests of vehicles, the task execution in either vehicles or the cloud, and finally to service responses. During the main simulation, the IsV2C visualizes the simulation process using PyViz[13] and records logs as shown in Fig. 3(b) and (c) respectively.

Summarization of simulation results: The IsV2C summarizes simulation results using various metrics to show service levels of each application. The final results are plotted into 19 graphs on the web page as shown in Fig. 3(d).[14]

B. Road Traffic Modeling

The IsV2C conducts a road traffic simulation using SUMO. Given the simulated region, the movement of vehicles in the area is modeled by two parameters: *count* and *through traffic factor*. The count, denoting the number of entities per hour per kilometer, determines when to add vehicles during the simulation. The interval between the vehicle creations is derived as $3600/(\frac{\delta}{1000}) / count$ where δ is the duration of a simulation. The through traffic factor affects a probability that vehicles start and end their movements at the boundary of the area. If the factor is high, then the probability increases. At present, the speed of each vehicle is set to the legal limit of lanes as default. If a simulation user denotes the maximum speed, the speed is restricted to lanes' legal-speeds. The region information and the configuration of road traffic parameters are delivered to the road traffic simulation module; then, it generates a vehicle mobility scenario. When the main simulation starts, a vehicular trace is created based on the mobility scenario.

C. Network Modeling

As mentioned earlier, we have considered the LTE as a cellular network in our work. Another cellular network, 5G was also considered in the initial steps of the research; however, its standard has not been fully established yet. Thus we have only covered the LTE in this work.

The whole network mechanisms of the IsV2C is based on ns-3 [7] and Lena [24]. As a preliminary stage of the main simulation, the creation of the simulated environment involves the following steps:

(1) Create a cloud datacenter and a evolved packet core.
(2) Install the Internet and establish connections between the datacenter and a packet data network gateway (PGW).
(3) Install IP stacks to VMs in the datacenter and configure their routing methods.
(4) Create base stations and place them in a hexagonal grid form in the entire simulated area.
(5) Install LTE devices to the base stations.
(6) Create UEs and configure their mobilities.
(7) Install LTE devices and network stacks to the UEs.
(8) Attach the UEs to the base stations and configure their routing methods.
(9) Install the client and the server part of each application to the UEs and the VMs respectively.

Now V2C applications have been deployed based on UDP. Their behaviors are determined by the simulation setup and also the task emulation. For other LTE users (i.e., non-vehicle LTE users), we have built a dummy application that only transmits and receives the user-given size of data. Also, the movements of the vehicles and the other LTE users are determined by the vehicular trace and a random direction model with a given speed.

Given the simulation scenario, an end-to-end simulation is then conducted by the simulation controller, and the network simulation module handles all the network operations between the UEs and the VMs. Besides, X2-based handovers among base stations are also conducted. If a signal strength between a UE and its connected base station is low, the handover occurs for the UE by switching the previous base station to closer and stronger one. After the handover is over, the UE establishes a new connection with the new base station in the neighbor cell.

D. Cloud Modeling

1) Cloud resource placement and task allocation: We assume that clusters are statically partitioned for each application in the cloud datacenter and consist of homogeneous compute nodes and VMs. Simulation users can configure their cloud environment by themselves. In the configuration, the IsV2C provides three strategies for cloud resource placement as follows.

- *Con:* A strategy to maximize VM consolidation by co-locating the possible maximum number of VMs in each compute node.
- *Man:* A strategy to co-locate the user-given number of VMs in each compute node.
- *Fair:* A strategy to maximize VM distribution by co-locating the possible minimum number of VMs in each compute node.

Also, for task allocation, the IsV2C fairly maps vehicles to VMs. Thus, most VMs can accommodate service requests from the same number of vehicles.

2) An emulation of task execution: It is significantly difficult to predict the computational performance of tasks in the cloud due to their hardware dependencies [25,26] and the complexity of performance interference [27,28]. Thus, contrary to most cloud simulators [8,9,10,11], which users should describe the details of computation workloads and their behaviors, the IsV2C takes an emulation-based approach similar to previous researches [12,13].

[13]https://www.nsnam.org/wiki/PyViz
[14]The details of the graphs are available on our website (http://ncl.kaist.ac.kr/wp-content/uploads/2017/02/supple.pdf).

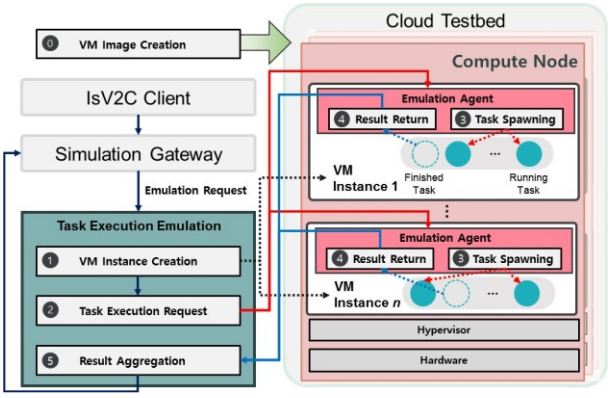


Figure 4. The emulation steps of the task execution of the IsV2C.

Fig. 4 shows emulation steps for each application. As a preliminary step, the VM image of the application should be created beforehand. Triggered by the simulation gateway, the emulation starts by creating VM instances with the created image. The number of the VM instances is given by the cloud resource placement strategy. After the creation, the emulation module requests the task execution to the emulation agent in each VM instance, and the task allocation strategy determines the number of the vehicles mapped to the instance. The emulation agent then spawns the tasks at the request interval given in the simulation setup. After the emulation is finished, each task execution time is delivered to the emulation module. For each vehicle, only one VM instance is utilized for the emulation. In the main simulation, the task computation time in the cloud and vehicles is determined to follow $N\left(e_{VM}\cdot\mu, e_{VM}{}^2\cdot\sigma^2\right)$ and $N\left(e_{VEH}\cdot\mu, e_{VEH}{}^2\cdot\sigma^2\right)$ respectively, where e_{VM} and e_{VEH} are the constants to represent the relative performance of VMs and vehicles, and μ and σ are the mean and the standard deviation of aggregated task execution time.

3) Cloud cost model: We denote $\pi(t)$ as the profit of the cloud provider during [0,*t*], and it is estimated as

$$\pi(t) = C_U(t) - C_O(t) - C_P(t) \tag{1}$$

where $C_U(t)$, $C_O(t)$, and $C_P(t)$ are cloud usage cost of the service provider, compute node operating cost and SLA penalty cost of the cloud provider in the period. Even though other costs, such as space cost, cooling cost, and maintenance cost, should also be reflected for better profit estimation [29], currently we only consider the portion related to the task computation in (1).

We now describe each element of (1) in detail. Firstly, for the cloud usage cost, we utilize the pricing policies of public cloud IaaS services such as Amazon EC2[15], Google Compute Engine[16], and Microsoft Azure[17]. It involves both VM instance charge and data transfer charge. Also, VM the instance charge is set differently depending on the rental types: *on-demand* and *reserved*.

Secondly, the operating cost of compute nodes is estimated based on Hamilton's datacenter cost model [30] as

$$C_O(t) = \sum_s \sum_j PUE \cdot \pi_{ELEC} \cdot p_{sj} \cdot \bar{u}_{sj}^{POW} \cdot t \tag{2}$$

where PUE is the value of power unit effectiveness, π_{ELEC} is an hourly electricity price, and p_{sj} and $\bar{u}_{sj}^{POW}$ are respectively the provisioned power and the average power usage in compute node j of cluster s. In this work, we approximate the $\bar{u}_{sj}^{POW}$ based on Blackburn's power consumption model [31] as

$$\bar{u}_{sj}^{POW} = 0.667 + 0.333 \cdot \frac{n_{sj}^{VM}}{n_{sj}^{VM,max}} \cdot \bar{u}_{sj}^{COMP} \tag{3}$$

where n_{sj}^{VM} and $n_{sj}^{VM,max}$ are the number of created VMs in the compute node and its maximum value respectively. $\bar{u}_{sj}^{COMP}$ is the average VM utilization in the compute node, and it is calculated as

$$\bar{u}_{sj}^{COMP} = \sum_k \sum_i \frac{T_{sjki}^{COMP}}{n_{sj}^{VM} \cdot t} \tag{4}$$

where T_{sjki}^{COMP} is the total task execution time for vehicle i on VM k in the compute node during [0,*t*].

Finally, for the SLA penalty cost imposed if the service levels do not meet, we consider four models that are proportional to performance degradation. One of the models is directly commensurate with performance degradation, and the others are based on the policies of Amazon EC2, Google Compute Engine, and Microsoft Azure each. Although public cloud providers offer their policies only for resource availability at present, we have modified them slightly also to consider performance degradation. The SLA penalty costs by the four models are obtained by

$$C_P(t) = \begin{cases} \sum_s \sum_j \sum_k c \cdot \pi_{sjk}^{VM} \cdot \bar{d}_{sjk} \cdot t \\ \textit{based on directly proportional model}, \\ \sum_s \sum_j \sum_k c \cdot \pi_{sjk}^{VM} \cdot \left(0.1 \cdot 1_{0.005<\bar{d}_{sjk}\le 0.01} + 0.3 \cdot 1_{\bar{d}_{sjk}>0.01}\right) \cdot t \\ \textit{based on Amazon EC2's model}, \\ \sum_s \sum_j \sum_k c \cdot \pi_{sjk}^{VM} \cdot \left(0.1 \cdot 1_{0.005<\bar{d}_{sjk}\le 0.01} + 0.25 \cdot 1_{0.01<\bar{d}_{sjk}\le 0.05} + 0.5 \cdot 1_{\bar{d}_{sjk}>0.05}\right) \cdot t \\ \textit{based on Google Compute Engine's model}, \\ \sum_s \sum_j \sum_k c \cdot \pi_{sjk}^{VM} \cdot \left(0.1 \cdot 1_{0.005<\bar{d}_{sjk}\le 0.01} + 0.25 \cdot 1_{\bar{d}_{sjk}>0.01}\right) \cdot t \\ \textit{based on Microsoft Azure's model}. \end{cases} \tag{5}$$

where c is the proportional constant, π_{sjk}^{VM} and $\bar{d}_{sjk}$ are each the hourly charge and the average performance degradation of VM k in compute node j of cluster s, respectively. Note that 1_A is the indicator function that returns 1 if A is satisfied and 0 otherwise.

E. Service Offloading

The IsV2C allows simulation users to determine whether each simulated application is to be offloaded or not. If they choose *NoOff*, the applications are only executed in vehicles.

[15]https://aws.amazon.com/ec2
[16]https://cloud.google.com/compute
[17]https://azure.microsoft.com

Otherwise, they are either offloaded in every trial (*AllOff*), or with the probability given by users (*ProbOff*).

IV. IsV2C Validation

In this section, we present the validation of the IsV2C via the evaluation of service feasibility for three sample V2C applications.

A. Simulation Setup

We implemented an IsV2C server in a machine running Ubuntu 14.04 LTS. A cloud testbed was built based on OpenStack[18], and we used a compute node that has eight cores of Intel® Xeon® CPU E5-2650 v2 @ 2.60 GHz model and 16 GB RAM for the emulation of task execution. The testbed provides three types of flavors, and their sizes are the same with *t2.small* (1 VCPU and 2 GB RAM), *t2.medium* (2 VCPU and 4 GB RAM), and *t2.large* (2 VCPU and 8 GB RAM) in Amazon EC2 respectively. At present, the IsV2C supports the following three applications to be emulated. *Sirius*[19] and *Blind Motion*[20] are open source projects for an end-to-end voice and vision-based intelligent personal assistant and for a deep learning-based application to detect maneuvers of vehicles using accelerometer and gyroscope, respectively. *Custom* is a sample application to impose computation workloads as much as users want.

As shown in Fig. 5, we chose the vicinity of Guseong-dong, Daejeon, Korea for the simulated region. The size of the area was set to 2.7 x 2.5 km. We conducted simulations for all the three applications, and the selected control parameters are shown in Table II.[21] Note that the parameters related to base stations were determined based on [32] and the speed of each vehicle was set to the legal limit of lanes as default.

B. Result[22] and Discussion

In this subsection, we show and discuss simulation results. We denote *baseline* as a reference case where the number of UEs is (75,40) for (vehicles, other LTE users), the number of co-located VMs in a compute node is (3,4,4) for (*Sirius, Blind Motion, Custom*), and the task offloading strategy is *AllOff*. For every case in the evaluation, all the simulation parameters were set as same as the baseline except for manipulated variables. Note that we employ the direct proportional model depicted in (5) and monthly reservation model for SLA penalty cost and cloud usage cost respectively in the following analysis.

1) Effects of the number of vehicles and other LTE users: Fig. 6 illustrates the simulation results on the number of vehicles and other LTE users. As shown in Fig. 6(a) and (b), we can observe that the average completion time, the deadline violation rate, and the average VM utilization increase for all the applications with the growth of the number of UEs. Note that *D* denotes deadline. The main rationale is the increase of the number of co-allocated tasks in each compute node, which causes more performance interference. As for the network performance, unlike our expectation, the growth of the number of the UEs did not affect significantly to the average transmission and reception time. We can see that the latency is low enough under the 150 milliseconds for every case. We guess that it is because the data size of each UDP application is relatively small; thus the vehicles do not generate burst network traffic. However, interestingly, Fig. 6(c) shows the increasing trend as the data traffic size grows for all the lines. $P\left[\text{Tx Time} > \bar{T}^{TX}_{baseline}\right]$ and $P\left[\text{Rx Time} > \bar{T}^{RX}_{baseline}\right]$ indicates when the data transmission and reception time are greater than the average value of the baseline. Thus, we can infer that the overall network performance is affected by the data traffic size.

Fig. 6(d) illustrates the analysis of the cloud provider's monthly profit assuming the service provision lasts 24 hours every day. We can see that the operating cost of compute nodes and the SLA penalty cost increase with the growth of the number of UEs. It is due to the increase of the computation load in compute nodes, which causes greater power consumption and also deadline violations. As the cloud usage cost does not vary, the profit decreases with the growth of the number of UEs.

2) Effects of the utilization of compute nodes: Fig. 7 shows the simulation results that manipulate the number of co-located VMs in each compute node. As we expected, the

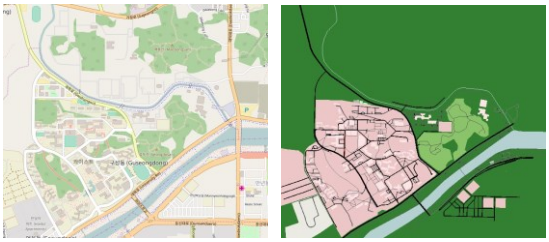


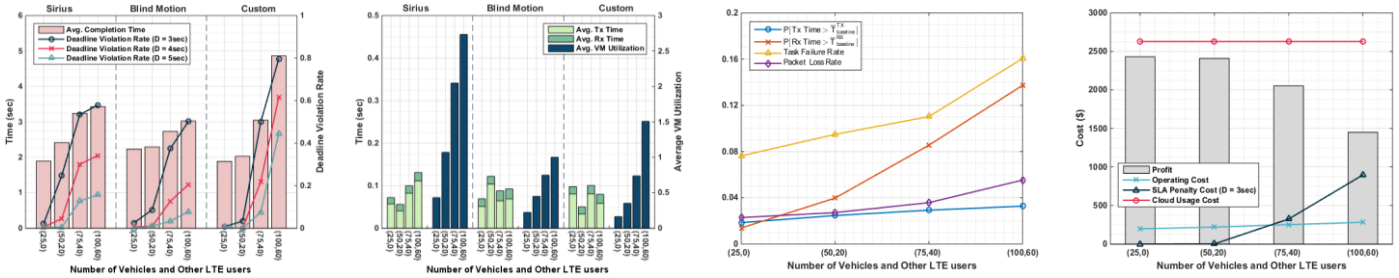
Figure 5. The simulation region (a) illustrated by the OSM and (b) illustrated by the SUMO.

TABLE II. The Selected Simulation Control Parameters

Simulation Duration (sec)		50
Vehicle	Max speed (m/s)	-
	e_{VEH}	3
Other LTE user	Speed (m/s)	1.1
	Input/Output data size (byte)	1,000,000
	Request interval (sec)	Exp(0.2)
Base station	Number	21
Internet	Data rate (Gb/s)	100
	Maximum transmission unit (MTU)	1500
	Delay (sec)	0.01
Cloud datacenter	Number of compute nodes	100
	provisioned power in compute nodes (W)	165
	PUE	1.45
	Hourly electricity price ($)	0.07
	e_{VM}	1
	Cost policy	Amazon EC2

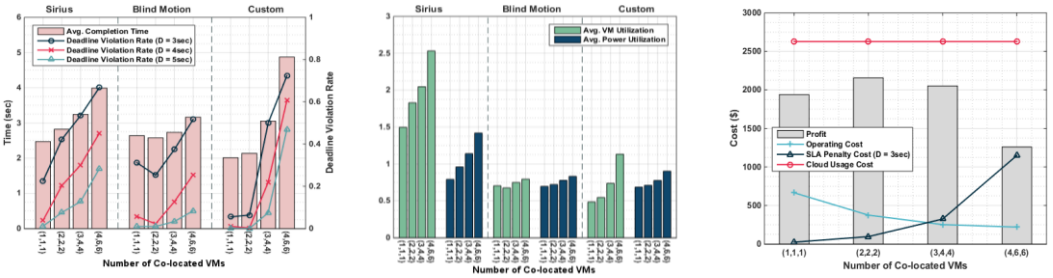
	Sirius	*Blind Motion*	*Custom*
Input/Output data size (byte)	50/25	8100/3700	2048/2048
Request Interval (sec)	Exp(0.25)	7.5	9
Number of VMs in the cluster	25	25	25
Flavor of VM instances	*t2.large*	*t2.medium*	*t2.medum*
Maximum number of co-located VMs in a compute node	4	8	8

[18]https://www.openstack.org
[19]http://sirius.clarity-lab.org
[20]https://blindmotion.github.io/
[21,22]The details of the simulation parameter settings and the graphical representation of the simulation results are available on our web site (http://ncl.kaist.ac.kr/wp-content/uploads/2017/02/supple.pdf).



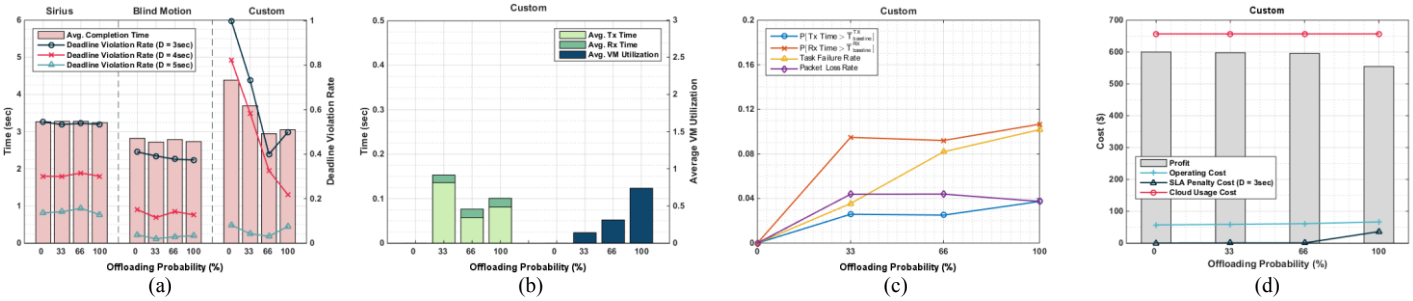
(a) (b) (c) (d)

Figure 6. Simulation results by manipulating the number of vehicles and other LTE users as (25,0), (50,20), (75,40), and (100,60).



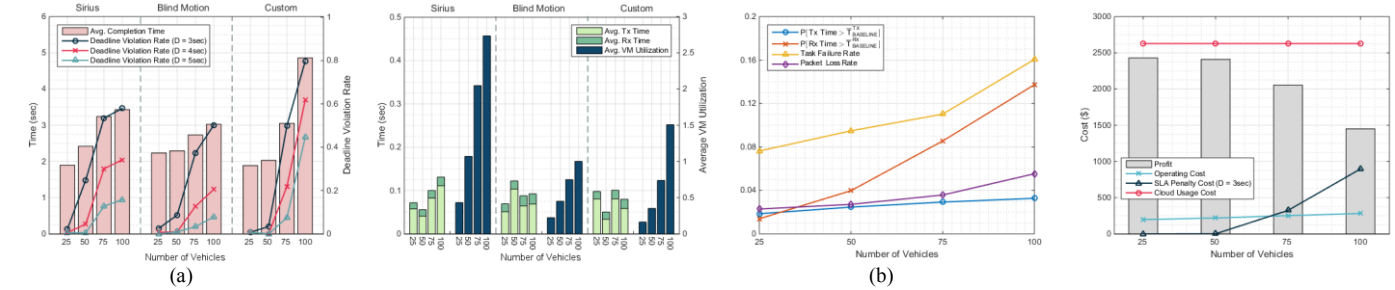
(a) (b) (c)

Figure 7. Simulation results by manipulating the number of co-located VMs in a compute node as (1,1,1), (2,2,2), (3,4,4), and (4,6,6).



(a) (b) (c) (d)

Figure 8. Simulation results by manipulating the offloading probability as 0 %, 33 %, 66 %, and 100 %.



(a) (b) (c) (d)

increased number of tasks allocated to compute nodes. In Fig. 7(c), we can see that the profit is the highest in the case of when the number of co-located VMs is two. Also, the compute node operating cost and the SLA penalty cost have a strong inverse relationship compared to Fig. 6(c) and 8(c).

3) Effects of service offloading policy: Fig. 8 indicates the relationship between the offloading degree and performance. As described in Fig. 8(a), for Custom, we can find that the average completion time is shorter when the offloading probability is 66 % compared to when it is 100 % even if we assumed that VMs outperform vehicles as much as three times. It is due to the reduction of performance applying AllOff.

C. *Usage*

The IsV2C enables simulation users such as service and cloud providers to find the optimal service setup by simulating a variety of configurations. For example, using the above results, service providers can easily identify that Sirius is the most vulnerable to the increase in data traffic size. Also, they might prefer to offload two-thirds of service requests for Custom if remaining conditions are the same. From the perspective of cloud providers, it seems that locating the maximum number of VMs in each compute

node is not a reasonable solution due to performance degradation. Also, simulation users can evaluate diverse scenarios by manipulating other parameters. For instance, if they want to identify the relationship between service provision and network performance in a particular region, the IsV2C allows them to configure a network environment in the area by manipulating parameters such as the number of base stations.

V. Conclusion

In this paper, we have presented the IsV2C, a new integrated road traffic-network-cloud simulator for V2C connected car services. The IsV2C provides (1) user convenience from the user-friendly GUI and the high visibility in result analysis, (2) a organic simulation to identify end-to-end service performance, (3) the careful emulation of task execution to mimic the computational behaviors in both vehicles and the cloud, and finally (4) a variety of metrics for the service evaluation. We have shown its functionality and effectiveness with the validation using three sample V2C applications in an urban area. We are now extending the IsV2C to be more elaborated and consider the ad-hoc vehicular cloud. We will also address a variety of challenging analytic problems such as the service quality optimization and the cost-effective resource assignment.

Acknowledgment

This work was supported by Hyundai Motor Company and KIA Motors Corporation.

References

[1] J. Wan, D. Zhang, Y. Sun, K. Lin, C. Zou, and H. Cai, “VCMIA: A Novel Architecture for Integrating Vehicular Cyber-Physical Systems and Mobile Cloud Compuating,” Mobile Networks and Applications, vol. 19, no. 2, 2014.

[2] H. Zhang, Q. Zhang, and X. Du, “Toward Vehicle-Assisted Cloud Computing for Smartphones,” IEEE Transactions on Vehicular Technology, vol. 64, no. 12, 2015.

[3] A. Ashok, P. Steenkiste, and F. Bai, “Enabling Vehicular Applications using Cloud Services through Adaptive Computation Offloading," Proc. MCS’15.

[4] "ITS; Vehicular Communications; Basic Set of Applications; Local Dynamic Map (LDM); Rationale for and Guidance on Standardization", ETSI TR 102 863, 2011.

[5] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, "SUMO-Simulation of Urban Mobility: An Overview," Proc. SIMUL'11.

[6] A. Varga and R. Hornig, "An Overview of the OMNeT++ Simulation Environment," Proc. SIMUTools'08.

[7] G. F. Riley and T. R. Henderson, "The ns-3 Network Simulator," Modeling and Tools for Network Simulation, 2010.

[8] D. Kliazovich, P. Bouvry, Y. Audzevich, and S. U. Khan, "GreenCloud: A Packet-level Simulator of Energy-aware Cloud Computing Data Centers," Proc. Globecom'10.

[9] B. Wickremasinghe, R. N. Calheiros, and R. Buyya, "CloudAnalyst: A CloudSim-Based Visual Modeller for Analysing Cloud Computing Environments and Applications," Proc. AINA'10.

[10] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms," Software-Practice and Experience, vol. 41, no. 1, 2011.

[11] A. Núñez, J. L. Vázquez-Poletti, A. C. Caminero, G. G. Castañé, J. Carretero, and I. M. Llorente, "iCanCloud: A Flexible and Scalable Cloud Infrastructure Simulator," Journal of Grid Computing, vol. 10, no. 1, 2012.

[12] R. N. Calheiros, M. A. S. Netto, C. A. F. De Rose, and R. Buyya, "EMUSIM: An Integrated Emulation and Simulation Environment for Modeling, Evaluation, and Validation of Performance of Cloud Computing Applications," Software-Practice and Experience, vol. 43, no. 5, 2013.

[13] I. K. Kim, W. Wang, and M. Humphrey, "PICS: A Public IaaS Cloud Simulator," Proc. CLOUD'15.

[14] Y. Jararweh, M. Jarrah, M. Kharbutli, Z. Alshara, M. N. Alsaleh, and M. Al-Ayyoub, "CloudExp: A Comprehensive Cloud Computing Experimental Framework," Simulation Modeling Practice and Theory, vol. 49, 2014.

[15] M. Amoretti, A. Grazioli, and F. Zanichelli, "A Modeling and Simulation Framework for Mobile Cloud Computing," Simulation Modeling Practice and Theory, vol. 58, no. 2, 2015.

[16] M. Quwaider, Y. Jararweh, M. Al-Ayyoub, and R. Duwairi, "Experimental Framework for Mobile Cloud Computing System," Proc. MCSMS'15.

[17] S. Guan, R. E. D. Grande, A. Boukerche, "An HLA-Based Cloud Simulator for Mobile Cloud Environments," Proc. DS-RT'16.

[18] S. Wang, C. Fan, C. -H. Hsu, Q. Sun, and F. Yang, "A Vertical Handoff Method via Self-Selection Decision Tree for Internet of Vehicles," IEEE Systems Journal, vol. 10, no. 3, 2016.

[19] M. Lauridsen, L. C. Giménez, I. Rodriguez, T. B. Sørensen, and P. Mogensen, "From LTE to 5G for Connected Mobility," IEEE Communications Magazine, vol. 55, no. 3, 2017.

[20] S. Lee, S. Hyeon, J. Kim, H. Roh, and W. Lee, "The Useful Impact of Carrier Aggregation: A Measurement Study in South Korea for Commercial LTE-Advanced Networks," IEEE Vehicular Technology Magazine, vol. 12, no. 1, 2017.

[21] M Piórkowski, M. Raya, A. L. Lugo, P. Papadimitratos, M. Grossglauser, J. -P. Hubaux, "TraNS: Realistic Joint Traffic and Network Simulator for VANETs," ACM SIGMOBILE Mobile Computing and Communications Review, vol. 12, no. 1, 2008.

[22] A. Wegener, M. Piórkowski, M. Raya, H. Hellbrück, S. Fischer, and J. -P. Hubaux, "TraCI: An Interface for Coupling Road Traffic and Network Simulators," Proc. CNS'08.

[23] C. Sommer, R. German, and F. Dressler, "Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis," IEEE Transactions on Mobile Computing, vol. 10, no. 1, 2011.

[24] N. Baldo, M. Miozzo, M. Requena-Esteso, and J. Nin-Guerrero, "An Open Source Product-Oriented LTE Network Simulator based on ns-3," Proc. MSWiM'11.

[25] B. Farley, A. Jeuls, V. Varadarajan, T. Ristenpart, K. D. Bowers, and M. M. Swift, "More for Your Money: Exploiting Performance Heterogeneity in Public Clouds," Proc. SoCC'12.

[26] F. Xu, F. Liu, and H. Jin, “Heterogeneity and Interference-Aware Virtual Machine Provisioning for Predictable Performance in the Cloud,” in IEEE Transactions on Computers, vol. 65, no. 8, 2016.

[27] D. M. Novaković, N. Vasić, S. Novaković, D. Kostić, and R. Bianchini, "DeepDive: Transparently Identifying and Managing Performance Interference in Virtualized Environments," Proc. ATC'13.

[28] C. Delimitrou and C. Kozyrakis, “HCloud: Resource-Efficient Provisioning in Shared Cloud Systems,” Proc. ASPLOS’16.

[29] C. D. Patel and A. J. Shah, "Cost Model for Planning, Development and Operation of a Data Center," HPL-2005-107(R.1), HP, 2005.

[30] J. Hamilton, “Cost of Power in Large-Scale Data Centers," in Keynote at SIGMETRICS'09 [Online]. Available: http://perspectives.mvdirona.com/2008/11/.

[31] M. Blackburn, “Five Ways to Reduce Data Center Server Power Consumption," White Paper, Green Grid, 2008.

[32] "LTE; E-UTRA; RF Requirement for LTE Pico Node B", ETSI TR 136 931, 2011.