ASM Sc. J., 13, Special Issue 1, 2020 for ICFICE, 104-113

Intelligent Transport Sensing Data Collection Strategy in Software Defined Edge Vehicular Network

Lionel Nkenyereye, Jaeung Lee and Jong-Wook Jang*

Department of Computer Engineering, Dong-Eui University 176 Eomgwangro,Busanjin-Gu,Busan,614-714,Korea

Software Defined Network (SDN) and edge computing are emerging technologies in information and communication technology to minimize data collection latency on the Internet of Vehicles (IoV). The car has the potential to contribute to the development of smart cities by accessing urban data that are collected and reported to a remote data centers through wireless communication. This paper presents the study on intelligent transport sensing data collection strategy in a Software-Defined Edge vehicular networking. The two sensing data collection schemes are cooperative vehicular and edge prediction mode based Software Defined Vehicular Network (SDVN). We prove that urban sensing data collection cost through a car sensing platform for urban data collection algorithm to find an optimal strategy. We set up a simulation scenario based on realistic traffic and communication features and demonstrate the scalability of the proposed sensing data collection. It can be seen that the edge cell predictive trajectory decision-based SDEVN mode scheme reduces greatly the cost when the road seems to have a high number of the vehicle

Keywords: vehicle as a sensor platform; software defined network; urban sensing; roadside units; vehicular network, edge computing

I. INTRODUCTION

A wide range of applications are proposed since the car is seeing as an urban sensing platform. The car has for long time used as single sensing probe where different sensors embedded in the vehicle collect urban sensing data. These sensors exchange data through a technology called Control Area Network (CAN) (Kiencke *et al.*, 1986). Since the wireless communication between vehicles and fixed infra-structures has been implemented, a vehicle can exchange information with others through Vehicle-to-Vehicle communication (V2V) or vehicle-to-infrastructures (V2I) like Road Side Units (RSUs) (Riva *et al.*, 2007).

The car has the potential to contribute to the development of smart cities by accessing urban data that are collected and reported to remote data centers through wireless communication. The advances of the automotive industry are considerably turn on the deployment of in-vehicle sensors which are nowadays considered as imperative components of any vehicle whatever how luxury it is. Sensors are used to monitor its operations and the status of its parts and enhance the driving experience. Some of these sensors are embedded into the vehicles and accredited under country regulation.

Software Defined Network (SDN) and edge computing (Boucetta *et al.*, 2017; Rong *et al.*, 2013) are emerging technologies in information and communication technology to minimize data collection latency on the Internet of Vehicles (IoV) (Bonomi *et al.*, 2012). These technologies lead the authors of this paper to propose a novel sensing data collection strategy schemes. The SDN is a growing computing and networking concept that has implicit ability to associate V2I and VANETS (Mohammad *et al.*, 2015). SDN separates the control plane and data plane entities. It executes the control plane software on general-purpose hardware. SDN permits self-supported deployment of control, traffic forwarding and computing devices. Fog computing relays on the use of network and

*Corresponding author's e-mail: jwjang@deu.ac.kr hardware virtualization concept to provide computation, storage, and networking services between subscribers' devices and existing cloud computing data centers. Nonetheless, fog network comports several devices that are considered to use a lower computation power in general. Therefore, one standalone device may demand an excessive amount of resources to effectively terminate a request. In fact, it is workable to execute computing tasks in fog network in distributed configuration with enough computation resources and network virtualized functions (NVFs) (Rajendra *et al.*, 2016).

This paper presents the study on urban sensing management strategy in a Software-Defined Edge vehicular networking. The two sensing data collection schemes are cooperative vehicular and edge prediction mode based Software Defined Edge Vehicular Network (SDEVN) (Nguyen *et al.*, 2015).

To the best of our knowledge, there is no work which studies the sensing data collection management in the context of SDVN. In this work we provide the following contribution: we propose two urban sensing data collection scheme. Firstly, cooperative vehicular mode. Therefore, with this scheme, the vehicle observes its neighboring vehicles and sets up a vehicular cluster for cooperative sensing data collection. The second scheme is edge prediction mode based SDEVN, The SDEVN Controller determines how much effort the sensing data collection request requires and calculates the number of RSUs required to support coverage of one RSU to the other. In case of probability neighboring cars to finish the urban sensing request is uncertain, edge prediction mode based SDEVN mode is selected. Thus, The Road Side Units (RSUs) extracts resources level OF neighboring vehicles and location information, then send that information to the SDEVN controller to compute the movement trajectory of the candidate's vehicles. The SDEVN forecasts and determines the position and then allows reconnection to the following RSU of each vehicle.

The remainder of the paper is organized as follows. Section 2 provides an overview of the SDEVN is provided. The idea of the urban sensing data collection and its management in SDEVN is elaborated in section3. In addition, the evaluation of the proposed sensing data collection is analysed and relevant results are provided in section 4. Section 5 concludes this paper with some hints to future research.

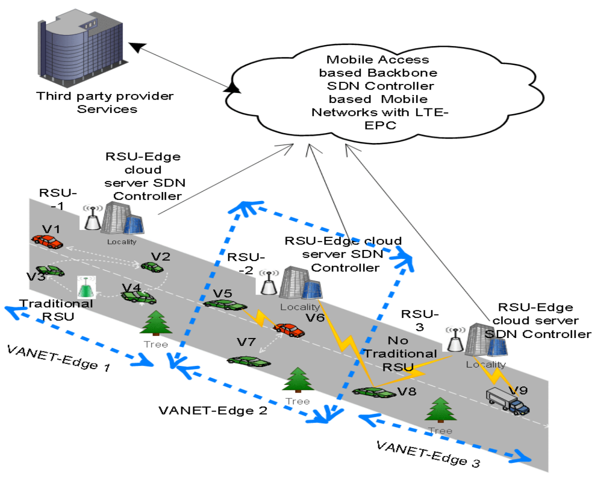
Third party provider Services
Mobile Access based Backbone SDN Controller based Mobile Networks with LTE-EPC
RSU-Edge cloud server SDN Controller
RSU--1
Locality
V1
V2
V3
V4
Traditional RSU
Tree
VANET-Edge 1
RSU--2
RSU-Edge cloud server SDN Controller
V5
V6
V7
Locality
Tree
VANET-Edge 2
RSU-3
No Traditional RSU
RSU-Edge cloud server SDN Controller
V8
V9
Locality
Tree
VANET-Edge 3

Figure 1. Topology structure of Software-Defined Edge Vehicular Network

II. SOFTWARE-DEFINED EDGE VEHICULAR NETWORK

Based on Figure 1, Software-Defined Mobile Edge vehicular network has several VANET-edge infrastructures. The VANET-edge 1 has big advantages in LTE for V2V before D2D LTE enables vehicles to communicate their geospatial coordinates to each other directly when in the low or no-connectivity range up to 100 meters. However, in VANET-edge 3, the restricted coverage issue is susceptible because of deficient RSUs. On the other hand, the wired network, that operates as backbone network would provide a higher volume of sensing data collected by vehicles and in turn, transferred to the cloud computing for further processing insights.

Unfortunately, in a pure network, the network traffic of data transmission is defined to last an unbearable processing time because of distance between a vehicle and cloud computing. To overcome the above issue, the edge computing technology has been presented to dive into an approach which decreases the overall delay due to the distance between the vehicle and the remote data center processing. Edge computing is defined as a network of small-scale data centers that store or aggregate data locally and push at demand received data to a cloud storage repository. In Edge computing, the cloud computing resources and storages are discovered at the edge of the vehicles networks and are close to the vehicles.

Since the applications are deployed on the edge computing machines, the edge computing server is installed along with the road Mobile Edge SDN Controller in this paper. The Mobile Edge SDN enabling SDN technology, Nguyen *et al.* (2015) show that SDN if applied to VANET, can provide adjustable, programmability and go in for new services. The SDN controller in Mobile Edge SDN performs the overall of the vehicles pass by the RSU controller. Therefore, Mobile Edge SDN based SDN is treated as data plane but stationary.

III. INTELLIGENT TRANSPORT SENSING DATA COLLECTION MODEL IN SDEVN

A. A System Model of the Car as an Urban Sensing Platform

In this section, we are presenting characteristics and information about all vehicles. We consider that the vehicle is capable of determining its position. The determination is its position is fulfilled by GPS receivers, which is one of the intrinsic assumptions for disseminating a message in the vehicular network by using well scalable position-based routing protocols Greedy Perimeter Stateless Routing (GPSR) (Broch, 1998). This location service information can be collected from the RSU. We consider (x_a, y_a) the location for a^{th} the vehicle. Let v_a be the a^{th} vehicle from all vehicles A^{area} in a given area. Then we describe such set as $A^{area} = \{v_0, v_1, v_2, \dots, v_a, \dots, v_{A-1}\}$, where $a \in \langle 0, A-1 \rangle, a \in \mathbb{N}, A \in \mathbb{N}$.

The location to be carried out in the abstraction of beacon information. In this case, the content of the beacon information differs from head sensing vehicle to simple vehicle candidate to collect sensing data request through the head sensing vehicle. We assume that each vehicle v_a has a certain number of characteristics such as acceleration, speed, General Resources (Input/output operations, storage, cache space, network) GN and vehicles ‘sensors. The speed $speed_a$ is a speed of a^{th} the vehicle at a particular time. The acceleration $accel_a$ is an acceleration of a a^{th} vehicle, determines how fast the velocity of an object divided by the time. CPU_a is the information that determines the performance resources of a^{th} the vehicle. In an urban environment, there are some obstacles compared to a highway road environment. Thus, most of the obstacles may difficult to forward the message to the neighboring vehicle. Therefore, we consider that junctions are seeing as an accurate location to forecast the movement of a vehicle. Let us consider another important characteristic of a^{th} a vehicle ‘sensors matrix A_a^{1xK}. This means that the matrix is controlling all information about the vehicle’s sensors. The following section will explain the model of head sensing vehicle.

B. Head Sensing Vehicle

First of all, we need to define the head sensing vehicle. This vehicle is the one the road traffic authorities sent to the request to collect environmental data through vehicle ‘sensor. When it receives the sensing request by the nearest RSU. The RSU sends beacon information that includes the following information: type of sensing data request, duration of sensing task, data sensing strategy. The head vehicle will generate packets to be executed by a neighbor vehicle in case of data sensing strategy is cooperative vehicle strategy. On the contrary, the edge cell trajectory prediction decision-based SDEVN is selected. The selection of the data sensing strategy depends on the duration the sensing task will last which consequently force the head sensing vehicle to handover several Road Side Units (RSU). This head sensing vehicle is seeing as a special vehicle and should have an index 0 . The large range of crowdsensing applications of the vehicular technology generates sensing tasks$x$. Let x_i be the i^{th} requested sensing task in a list of sensing request R^{tasks}, such list we would define $R^{tasks} = \{x_1, x_2, \dots, x_i, \dots, x_I\}$, where $i \in \langle 1, I \rangle, i \in \mathbb{N}, I \in \mathbb{N}$.

Let r_i be the i^{th} current performance resource level in a list of performance resource level P^{level}, then such list we have $P^{level} = \{r_1, r_2, \dots, r_i, \dots r_I\}$ where $i \in \langle 1, I \rangle, i \in \mathbb{N}, I \in \mathbb{N}$.

C. Sensing Request Size and Duration

SR_i is a duration of i^{th} sensing request. This parameter helps to calculate the duration of executing the sensing task and compute the total size of the sensing data collected. Therefore, SR_i^{result} is a resulting in the size of the sensing data for all candidates vehicles that perform the sensing request. Therefore, this parameter allows calculating the duration resulting in collecting environmental sensing data for i^{th} the sensing request. Then, SR_i – duration of sensing request i^{th} in seconds [s] SR_i^{result}- the size of i^{th} sensing request in bytes [B].

D. Instruction Block for Performing Sensing Request

The IB provides instructions to tell the central processing to tell what it is required to do. The IB purpose is to direct the head sensing vehicle to apply either cooperative vehicle strategy or edge cell trajectory prediction decision-based SDEVN. In cooperative vehicle strategy, the sensing task is performed in a traditional way where candidate's neighbor vehicles performed each their own collection of data and in turn sent them to the head sensing vehicle. For simplicity, we assume that all candidate's neighbor vehicles are in the same coverage area and have constant vehicle speed. If the sensing task requires the head sensing vehicle cross more than one RSU, the edge cell trajectory prediction decision-based SDEVN is recommended. In this situation, a trajectory decision is configured in the route flow table with the entries of the network topology of RSU at the edge network. As soon the head sensing tables receive SDN (OpenFlow) forwarding routing decisions, it will be possible to solve the issue of network disconnection due to the mobility of the vehicle as it leaves one RSU to another. In case the size is large and cannot be stored on the head sensing vehicle, this head sensing will transmit the sensing data to the RSU unit. As it travels from the coverage area of the RSU, the head sensing vehicle will not need the position of the next RSU and its connection address (MAC address) to report the current size of sensing data collected. As long as the flow entries about the forwarding decisions generated by the SDEVN controller, the head sensing vehicle will send the collected data to the RSU by reading the route flow tables, then not only the transfer time of sensing data will be decreased also the communication coverage issue will be solved.

E. Candidates and Neighbor Vehicles

We have already described a head sensing vehicle and all vehicles in a given area. The neighboring vehicle is a kind of vehicle that is within a communication range of the head sensing vehicle and V2V communication capability. A candidate neighboring vehicle means different neighboring vehicles participate in sensing request due to their availability and sufficient performance resource level. They hold this status until the data collection of the sensing task reach the final destination which may be either the head sensing task or the nearest RSU located at the position the sensing request is completed depends on the duration and the sensing strategy defined earlier in this paper. So in order to reach some destination vehicle, not only vehicle relay concept can be used but also the SDEVN controller can send a packet of sensing task to each RSU, and RSU transfers it to another vehicle that has not received the packet of sensing request yet through single-hop routing or V2V DSRC communication.

We consider that all vehicles that come out after the head sensing vehicle start scanning are seeing as neighboring vehicles N^{neigh} . A neighboring vehicle selected as candidates to perform sensing task in cooperation with other vehicles will be candidate neighboring vehicles $C^{candneighb}$. Let v_j is a j^{th} neighbor vehicle from all neighboring vehicles in a given area A^{area} after the vehicle v_0. Let v_k is a k^{th} candidate neighboring vehicle from all candidate neighboring vehicle in a list of neighboring vehicles V^{neigh} in a given area.

$C^{candneigh} = \{v_1, v_2, \dots, v_k, \dots, v_C\}$ (1) Where $k \in \langle 1, N \rangle; k \in \mathbb{N}, C \in \mathbb{N};$ $C^{candneigh} \subseteq N^{neigh};$ $N^{neigh} \subseteq A^{area}$ In this subsection we overviewed the functional algorithm without math functions, just to understand the high-level concept of the proposed method. We already know all parameters required to find optimal candidate neighboring vehicle execute some tasks with fewer resources.

Before starting the description of a flowchart, it is very important to reader clearly understand the blocks which are used in the flowchart. The oval or rounded rectangle is signaling the start of a process. Set of operations that change value, form, or location of data represented as a rectangle.

So, let us imagine, in some area and head sensing vehicle starts to distribute sensing tasks. It can be any sensing tasks of updating or maybe alerted, that something happened on a road. Adding to this, the crowdsensing task is to collect information about road condition, traffic conditions, weather information or taking picture or videos in a certain area ahead to the driving destination.

If we have no sensing request at the head sensing vehicle (or at the RSU infrastructure), the algorithm will check the beacon information every τ_1 millisecond until it receives the sensing request. We agree that at the time the RSU receives the request, it will broadcast to the near passing head sensing vehicles. The message is abstracted in beacon information. We assume that the head sensing vehicle can accept to execute only one sensing request a time.

It is recommended that each sensing task has a duration time and it decrements by τ_2 . Depending on a type of sensing strategy, the head sensing vehicle proceeds by a proactive scanning of neighboring vehicles after getting the sensing request. The head sensing vehicle will select among neighboring vehicles candidates vehicles those are most optimal to perform the sensing request. Therefore, the determination of candidate neighboring vehicles depends on the probability to execute the sensing request applied as we will explain in this section. The head sensing v_0 vehicle is then adding to a set of candidate neighboring vehicles $N^{neigh} = \{v_1, v_2, \dots, v_j, \dots, v_N\}$ where $k \in \langle 0, N \rangle$. Subsequently, for each, v_j determines vehicle candidate probability $P_{cand} = (v_j, x_i)$ for participating in executing sensing request x_i with the highest performance resourcePR_i . The duration time for executing the sensing request is known from the head sensing vehicle and it is constant time. Also, the total of resources consumption to complete the sensing task in the given duration. The total resource consists of three phases: transmit performance resource level, execution of sensing request, send back the collected sensor data.

$$Reso_j^{TR}(v_j, x_i) = reso_j^{trx}(v_j, x_i) + reso_j^{exe}(v_j, x_i) + reso_j^{send}(v_j, x_i)$$

Note that for v_0 the vehicle to be among candidate with candidate probability P_{cand} is 1, or 100% because the probability the head sensing vehicle do not complete the sensing request tends to zero. Therefore, $P_{cand}(v_0, x_i) = 0$, $Ress_j^{TR}(v_0, x_i) = 0$.

Subsequently, all vehicle candidate probability for $j + 1$ vehicles (neighboring candidates vehicles and head sensing vehicle $N^{neigh} = \{v_1, v_2, \dots, v_j, \dots, v_N\} + v_0$.

We determine the minimum vehicle candidate probability to pass the candidate request. The value will be constant and will be updated for each new proactive scanning triggered by the head sensing vehicle. After compare, we will get a new cluster with neighboring vehicles, which values more or equal to minimal vehicle candidate probability. After that operation at the best case scenario, we all neighboring candidates vehicles will stay with the head sensing vehicle, but usually with neighboring vehicles with a lower vehicle candidate probability will not be included in the cluster of neighboring candidate's vehicles.

$$C^{candneigh} \leftarrow \left(P_{cand}(v_j, x_i) \geq P_{min-vehcand}\right)$$

It may be that the neighboring candidate vehicles do not have any optimal performance resource due that all neighboring are busy, or the vehicle candidate probability is so low, or the head sensing vehicle rides alone on a road.

If there are no suitable neighboring candidatures vehicles, the SDEVN controller sends a message to each RSU, and each RSU transfers the message to vehicles within the RSU coverage. Then vehicle transfer message to another vehicle that has not received the message. We need to check if we get candidate neighboring vehicles. In that case, the RSU in the coverage area with those vehicle received message about sensing request will process the vehicle candidate probability. In that case, the heading sensing vehicle provides the sensing task and give the right to the SDEVN controller to process in turn the current sensing request. The SDN trajectory topology decision will be applied to help the vehicle to report the collected sensor data through its RSU in their coverage area. If there are suitable neighboring candidates' vehicles, the sensing task will be sent to them.

G. Vehicle Candidate Probability

In this subsection, we talk about the vehicle candidate probability $P_{cand(v_j,x_i)}$ function, that a sensing request will be attributed to the neighboring vehicle after the head sensing vehicle scans after receiving the sensing request. Therefore, the vehicle candidate probability consists of one significant parameter, which collectively complements each other and is seeing as a filter that helps to determine the neighboring candidate vehicles to perform a sensing request. There are performance resource parameter $\emptyset_j^{PR}$, $P_{cand}(v_j, x_i) = \emptyset_j^{PR}\ (v_j, x_i)$

H. Performance Request Filter Parameter

$\emptyset_j^{PR}$ is working with a Performance Resources (PR) of sensing request. It shows if some vehicle can execute a sensing request within its PR, otherwise sensing request will not be assigned to this neighboring vehicle. Initially, we define $Reso_j^{PR}(v_j, x_i)$ with $\emptyset_j^{PR}$ which is impervious.

Total resource consumption

$Reso_j^{PR}(v_j, x_i)$ is a total resource consumption that j^{th} vehicle needs to execute sensing request i^{th} . It is a sum of three resource consumption: reporting (transmit) current resource performance $reso_{0j}^{trx}(v_j, x_i)$, execution of sensing request $reso_j^{exe}(v_j, x_i)$ and sent back collected sensor data $reso_{jo}^{send}(v_j, x_i)$.

Resource consumption for reporting current resource Performance

Let IB is number of instructions of special request task i^{th} need to query the j^{th} vehicle and GN_j general resources of processing IB on j^{th} the vehicle, the resource consumption for a special incoming request i^{th} for reporting resource performance for j^{th} a vehicle is $reso_{0j}^{trx}(v_j, x_i) = \frac{IB_i}{GN_j}$.

Resource consumption for executing a sensing request

Let vehicle's sensors (A_a^{1xK}) are triggered to read the information as described in the Instruction block (IB) of a sensing request i^{th} until the duration time given in IB and GN_j general resources of j^{th} the vehicle. The resource consumption for executing a sensing request can be given by the following equation$reso_j^{exe}(v_j, x_i) = \frac{IB_i*(v_j(x_0,y_0),A_a^{1xK})}{GN_j}$.

The resource consumption for sending back the data size of the collected sensor data for a sensing request

Let SR_i^{result} the total size of collected data of a sensing request i^{th} . We set $Ch_{oj}(d_{oj})$ as the Shannon-hartley theorem (E. Price, 2012) to calculate the maximum rate at which information can be transferred over a channel for a given bandwidth and according to the distance between head sensing vehicle v_0 and the neighboring candidate vehicle v_j, so the resource consumption for transmitting back the collected data is

$reso_j^{send}(v_j, x_i) = \frac{SR_i^{result}}{Ch(d)*GN_j}$ where $Ch(d)$ is the channel capacity. Based on the 3GPP TR 36.785 in their Release 14 (Scapy), we extract parameters of a bandwidth B = 10 Mhz and received the power of -22dBm.

Knowing the resource performance of reporting resource performance execution of sensing request and sent back collected sensor data, we get the total resource consumption

$$Reso_j^{TR}(v_j, x_i) = reso_j^{trx}(v_j, x_i) + reso_j^{exe}(v_j, x_i) + reso_j^{send}(v_j, x_i)$$

Let $\emptyset_j^{PR}$ is the resource performance filter parameter of neighboring vehicles for sensing request i^{th}, if the total resource consumption for sensing request i^{th} will be less than the performance resource of sensing request i^{th} PR_i, the parameter $\emptyset_j^{PR}$ return 1, otherwise zero. This function will assist us to not select neighboring vehicles that need more resources before fora sensing requested is completed in the given duration of the sensing request.

$$\emptyset_j^{PR}(v_j, x_i) = \begin{cases} 1, if\ Reso_j^{TR}(v_j, x_i) < PR_i \\ 0, otherwise \end{cases}$$

I. Car Sensing Platform for Urban Data Collection Algorithm

The car sensing platform algorithm is presented shown on Figure 2.

```
Algorithm 1: Vehicle Sensing Platform for urban data collection
  Input: $R^{tasks} = \{x_1, x_2, ...., x_i\}$
  Output: RSU adding sensing request $x_i$ to the vehicle $v_j$ for collecting different type of
          data
  initialization;
  while $R^{tasks} \in \Theta$ do
    sensing request $x_i$ received by Head sensing vehicle $v_0$;
    if cooperative-vehicle mode then
      $v_0$ extract instructions of $x_i$;
      scanning vehicles close to $v_0$ getting a list of $N^{neigh}$;
      $N^{neigh} \leftarrow v_0$;
      while $v_j \in N^{neigh}$ do
        calculate $Resoj^{tr}(v_j, x_i)$;
        calculate $P_{cand}(v_0, x_i)$ ;
      end
      while $C^{candneigh}$ do
        $v_k$ start collecting sensor data ;
        if $v_k$ finish collecting data then
          sent back collected data to $v_0$ or RSU;
        else
          if $P_{cand(v_i),x_i} \geq P_{min-vehcand}$ then
            sent back collected data to $v_0$ or RSU;
          else
            Apply SDVN based eNB-type RSU approach;
            start collecting sensor data;
          end
          if $v_k$ finish collecting data then
            sent back collected data to $v_0$ or RSU;
          else
          end
        end
      end
    else
    end
  end
```

Figure 2. Car Sensing platform for data collection urban algorithm

The driving prediction parameter of vehicle during executing sensing task essential as long as it help for determining for a selected neighboring candidate vehicle how a candidate neighboring vehicle will ride within the signal coverage range or what will happen when if any of neighboring candidates vehicles needs more resources to complete the sensing request or when there is no any optimal neighboring candidates due to some case where they are busy or the resource are low to execute the sensing request. These assumptions leads us to the use of the SDN concept in VANETs by using the SDEVN controller. The strategy is based on the SDN Edge trajectory (topology) prediction decision mode. The topology establishment is based on trajectory prediction with double purposes. The first purpose concerns the prediction the position of the vehicle and force the RSU to judge the accurate RSU where the next reconnection with the vehicle should occur. For simplicity, we select the SDVN based architecture proposed by Cao *et al.* (2016) in order to solve the above scenario case.

IV. EVALUATION

Vehicles use only V2V communications type in case vehicles are located in the same coverage area. Otherwise, the SDEVN controller forces RSU to transmits sensing request messages to the other vehicles those cannot be reached by one hop communication type. All assumptions can be briefly written in the following way:

- Grid road map equipped with one-way roads
- V2V or V2I communication are considered
- Only the head sensing vehicles generates sensing requests, but each vehicle including head sensing vehicle can participate to the collection of data

depending on the type of strategy associated to the sensing requests.

- Maximal speed is 50km/h
- Performance resource decreases when a candidate neighboring vehicle is performing a sensing task. It gains it when is driving head or resting (no sensing receives from the head sensing vehicle)
- Candidate neighboring vehicle on which during execution of sensing task drop to the minimum level of performance resource must stop the current sensing request when this case happen. If this happen it will continue the sensing tasks assign until the sensing request is finished or when its level of performance resource is below to the critical threshold.
- We assume that each sensing request requires a certain amount of efforts to be executed. This amount of effort includes duration and maximum number of vehicle to be selected as candidates particularly when the cooperative vehicle mode is applied. This information is included in the beacon sent directly to the head sensing task by the request from urban management & monitoring through the nearest RSU.
- The effort is not increase over time and only decrease by the conditions of candidates neighboring vehicles
- The V2V communication follow a probabilistic model of delivery, that is, messages delivery is not guaranteed
- The messaging model used in broadcasting with a station hop-count. Each receive neighboring vehicles with a decrement (hop-count) in case (when) its hop-count is greater than zero
- We assume that all sensing task (request from urban management & monitoring authorities) requires cooperative collection, means more than 2 vehicles. This sensing request should be performed in cooperative manner.
- We assume all vehicles have the same type of vehicle's sensors

A. Simulation Environment

The part of our experiment is describing a type of controller. There are several well-known controllers available. The one that is used in our solution is POX. POX is broadly used for experiments, it is fast, lightweight and proposed as a platform, then a custom controller can be built on top of it. POX presents a framework for communicating with SDN switches through either the OpenFlow or OVSDB protocols. In addition, POX generally supports Windows, MAC OS, and Linux. In summary, POX can be immediately used as a basic SDN controller by using special components called stock components that come packed with it. For those with intention of using complex network topologies, SDN controller can be developed by creating new POX components.

POX is not the only SDN controller used for experimentation. Thus, three other controllers must be enumerated. Floodlight is another widely used controller. Floodlight is an open source and is currently written in Java. Some advantages of the floodlight are that is proposed as a powerful tool to help third parties to fatly modify the software and develop applications using a set of constraints to build web services by Applications Programming Interface (API) like Representational State Transfer (REST) API are concerned to simplify the applications interfaces to the product. OpenDaylight controller is the newly addition to OpenFlow controllers.

B. Network Emulator

Mininet-WiFi is the network emulator that we used in this paper for executing experiments.it is a routine network emulation tool that can be used for SDN. Therefore, in Mininet, network namespace gives subjective process with their own network interfaces, address ARP tables, routing tables. Mininet has advantages inherited of this feature of the kernel. It generally uses process-based virtualization to run networks devices such as nodes (hosts), switches on a single operating systems (OS) kernel. Example large networks up to 4096 hosts can experiment on a single operating system with different topologies can be experimented and emulated.

Implementing a network scenario in Mininet-Wifi is as simple by entering the command mn to have started with a network topology which includes one switch connected to two hosts and a controller located at a remote address. NOX (NOX) is the default controller for Mininet.

C. Packet Generator

Packet generation is executed by Scapy. It is an important packet manipulation tool for packet generation, scanning, trace routing, packet forging written in Python. Thus, scapy is used in this paper to generate UDP packet and send up the source IP address of the packets.

In POX controller, network developer uses a python programming language. Python language is in turn used to generate random source IP address. The core generator Mersenne Twister, used by Python generate a float of 53-bit precision (IEEE Std., 1997). By this function, we used a random function written in python inherits by "randrange". Based on this function, we can generate random numbers with precise distribution in order to compute a long period of random generation. To model send up a packet of source IP address these random numbers are combined together. The range of packet generation and type of packet are other parameters we determined in the scapy script.

In this paper, both urban sensing request and reporting sensing data we used UDP packets. To simply perform the experimentation, the interval of packets generation was adjusted. For an interval of sending urban sensing request with 40% rate of duration of sensing request, normal sending sensing interval is 0.15 and report back sensing data is 0.04. this will give us a window of 40% packets allowed to a single host.

D. Analysis of Results

We analyze the simulation results to demonstrate the performance of our proposed car sensing platform for urban data collection schemes. We consider ten RSUs located along a grid roadmap equipped with one-way roads and use Simulation of Urban Mobility (SUMO) to simulate the road traffic The density of the vehicles on the road is set as l = 0.4. , Maximal speed is 50km/h. the performance resource decreases when a candidate neighboring vehicles are performing an urban sensing request. It gains it when is driving head or resting (no sensing received from the head sensing vehicle).Candidate neighboring vehicle on which during execution of urban sensing task drop to the minimum level of performance resource must stop the current sensing request when this case happen. If this happens it will continue the sensing tasks assign until the urban sensing request is finished or when its level of performance resource is below to the critical threshold. The success of an urban sensing data collection tasks of these vehicles is grouped into five modes with the probabilities {0.04, 0.15, 0.3, 0.4, 0.1}. As resource performance is the most important factor affecting the urban sensing data collection request.

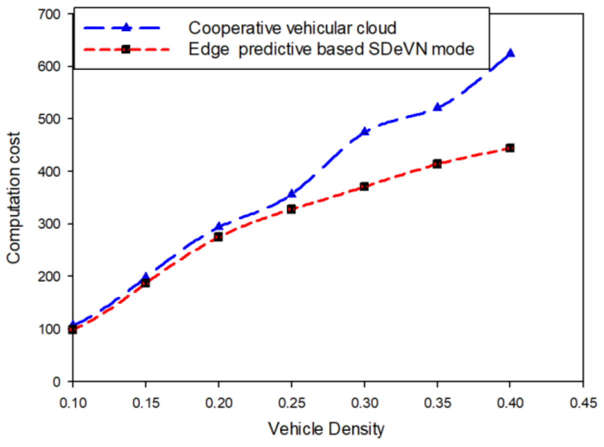

Figure 3. Computation request of sensing data collection strategy costs

Figure 3 evaluates the total computation urban sensing data collection costs in terms of the consistency of the vehicles on the road. We compare the performance of the proposed algorithm, edge cell predictive trajectory decision mode and cooperative vehicular mode. It can be seen that the edge cell predictive trajectory decision mode scheme reduces greatly the cost when the road seems to have a high number of the vehicle.

Furthermore, a large portion of the off-loaded sensing data collection tasks on the Mobile edge computing SDN controllers can be accomplished within the defined period when the vehicles accessing the RSU controller have not been altered. Therefore, there is no need to adopt a direct vehicular cloud strategy.

V. CONCLUSIONS

In this article, we proposed a sensing data collection schemes in Software-defined edge vehicular networks. Based on the schemes, we discussed the strategy based on the number of RSU controller needs to accomplish the sensing data task completion and the request task cost of direct vehicular cloud or edge cell trajectory prediction modes. The results demonstrated that our scheme greatly reduces the sensing data collection cost.

VI. ACKNOWLEDGEMENT

This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the ICT Consilience Creative program(IITP-2019-2016-0-00318) supervised by the IITP (Institute for Information & communications Technology Planning & Evaluation).

VII. REFERENCES

A.S. Mohammad, A. Fuqaha. and M. Guizani 2015, 'Software-Defined Networking for RSU Clouds in Support of the Internet of Vehicles', *IEEE Internet of Things Journal*, vol. 2, no. 2, pp. 133-144. DOI:10.1109/JIOT.2014.2368356.https://dx.doi.org/10.1109/INM.2015.7140467

B.T. Nguyen, M.L. Gyu, and G.D Yacine 2015, 'Software Defined networking-based Vehicular Adhoc Network with Fog Computing', in *Proceedings of 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pp. 1202-1207.

C. Rajendra, F. Syed., S. Paresh 2016, *Network Functions Virtualization (NFV) with a Touch of SDN*, ISBN-10:0-10-446305-6, Addison-Wesley.

E. Price, D.P. Woodruff 2012, 'Application of the Shannon-Hartley Theorem to Data stream and sparse Recovery', http://www.cs.cmu.edu/afs/cs/user/dwoodruf/www/pw12.pdf

F. Bonomi, R. Milito, J. Zhu, S. Addepalli 2012, 'Fog Computing and its role in the internet of things', in *Proceedings of MCC workshop on Mobile Cloud computing*.

Floodlight, http://www.projectfloodlight.org/floodlight/

I.S. Boucetta, Z.C. Johanyak and L.K. Pokoradi 2017, 'Survey on Software Defined VANETs', Gadus, vol. 4, no. 1, pp. 272-283.

J. Broch, D. Maltz, D. Johnson, Y. Hu, J. Jetcheva 1998, 'A performance comparison of multi-hop wireless ad hoc network routing protocols', Mobicom Mininet-WiFi [online] https://github.com/intrig-unicamp/mininet-wifi

NOX controller, https://github.com/noxrepo/nox

O. Riva, T. Nadeem, C. Borcea and L. Iftode 2007, 'Context-aware migratory services in as hoc networks', *IEEE Trans. Mobile Computation*, vol. 6, no. 12, pp. 1313-1328.

OpenDaylight Prject [online] at https://github.com/opendaylight

POX network platform [online] https://github.com/noxrepo/pox

Scapy Project, https://scapy.net/

Simulation of Urban Mobility, http://sumo.dlr.de/index.html.

U. Kiencke, S. Dais, and M. Litschel 1986, 'Automotive serial controller area network', SAE Tech, Paper 860391.

Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification, IEEE Std. 802.11, 1997.

Y. Rong, Y. Zhang, S. Gjessing, W. Xia and K. Yang 2013, 'Toward Cloud-based Vehicular Networks with Efficient Resource Management', *IEEE Network*, vol. 5, no. 27, pp. 48-55. DOI:http:10.1109/MNET.2013.6616115.IEEE

Y. Cao, L. Kaiming and W. Chao 2016, 'A novel min-cost Qos routing algorithms for SDN-based Wireless mesh network', 2nd IEEE International Conference On Computer and Communications (ICCC).