Dynamic Hierarchical Aggregation for Vehicular Sensing

Jagruti Sahoo, *Member, IEEE*, Soumaya Cherkaoui, *Senior Member, IEEE*, Abdelhakim Hafid, and Pratap Kumar Sahu, *Member, IEEE*

***Abstract*— Vehicular sensing has gained prominence in recent years with its use in entities, including traffic management centers, forensic authorities, and air pollution control units. It also provides end users with real-time street images, parking summaries, and road congestion status. To reduce bandwidth usage and improve the content value, the sensed data must be aggregated. Data aggregation is said to be efficient when the destination (i.e., a node that serves as a data collection point in the network) is capable of receiving sensed data from a significant proportion of vehicles. However, when a large number of vehicles attempt to send sensed data, the network becomes congested eventually causing packet losses and collisions. Thus, if aggregation is performed without considering key factors, such as number of vehicles and network dynamics, it is difficult to ensure the efficient collection of sensed data at the destination. In this paper, we propose a dynamic hierarchical aggregation scheme in which sensed data is aggregated using a hierarchy. Moreover, the hierarchy is dynamically updated based on theoretically estimated delivery efficiency. In particular, we perform partition and merge operations within the hierarchy to achieve an improved value of delivery efficiency. The simulation results show that the proposed scheme ensures efficient data collection even with stringent delay requirements and achieves scalability with respect to a number of vehicles in the network.**

***Index Terms*— Aggregation, road-side unit, vehicular sensing.**

I. Introduction

IN THE last decade, vehicular networks have drawn considerable attention from academia and industry because of their immense potential in areas ranging from vehicular safety, to traffic management, driver assistance, navigation, and infotainment. Besides these applications, vehicular sensing has gained significant interest recently because of the interactions between sensor-equipped vehicles and the physical world. Indeed, vehicles can accommodate a wide variety of sensors such as GPS, still/video cameras, accelerometers and pollution detectors; thus, vehicular sensing offers tremendous opportunity to visualize the dynamics of the environment at any instant, or over a period of time. Vehicular sensing in urban environments has many dimensions namely capturing surveillance videos of streets through image/video sensing [1]–[3], discovering available parking spaces [4], measuring the concentration of carbon dioxide (CO2) [5], [6], sensing traffic related events [6], discovery of location and number of available Road Side Units (RSUs) such as WiFi Access Points (APs) [7], [8]. The last application uses compressive sensing techniques and is useful in provisioning various services such as Internet Access and information sharing in vehicular networks.

Some of the sensing applications involve generation of huge amount of sensed data. One such application is multimedia sensing application [3]. Multimedia data is of paramount importance in intelligent transportation systems as they can be used in advanced driver assistance systems [9], [10] to enhance vehicle safety by increasing visibility of drivers. Self-driving vehicles also rely on stereo images [11], [12] and complex algorithms to compute 3D perception of the environment around them. Real-time images can be used by police to track criminal activities on roads. Images can also be used by traffic management centers to monitor events such as accidents and congestion.

To reduce bandwidth and achieve a scalable transmission of sensed data, various in-network data aggregation schemes have been proposed for vehicular networks [13]–[20]. These schemes achieve bandwidth savings by compressing the collected data using some transformation operation, known as aggregation [21]. Aggregation serves an integral part of sensing applications as users are generally interested in an aggregated view of an event rather than its finer details. There are mainly two types of aggregation: syntactic and semantic. In syntactic aggregation proposed for vehicular networks [19], [20], sensed data are concatenated in order to fit in a single communication packet, resulting in lower header overhead. On the other hand in semantic aggregation [19], [20], meaningful data is extracted by using certain aggregation functions (e.g. Average, Count, and Maximum). For example, a traffic information system uses the average velocity of a road segment (The portion of street between adjacent intersections is called a road segment) to determine its congestion status. Syntactic aggregation allows a full reconstruction of the original data from the aggregated data; whereas the semantic aggregation results in complete or partial

Manuscript received January 1, 2016; revised August 14, 2016 and December 7, 2016; accepted December 17, 2016. Date of publication May 2, 2017; date of current version August 28, 2017. The Associate Editor for this paper was D. Wu.

J. Sahoo is with South Carolina State University, Orangeburg, SC 29115 USA (e-mail: jagrutisahoo@ieee.org).

S. Cherkaoui is with the Department of Electrical and Computer Engineering, University of Sherbrooke, Sherbrooke, QC JIK 2RI, Canada (e-mail: Soumaya.Cherkaoui@USherbrooke.ca).

A. Hafid is with the Department of Computer Science and Operation Research, University of Montreal, Montreal, QC H3C 3J7, Canada (e-mail: ahafid@iro.umontreal.ca).

P. K. Sahu is with the CONNECT Centre, Trinity College Dublin, Dublin 2, Ireland (e-mail: pratap.k.sahu@ieee.org).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TITS.2017.2650991

loss of original information. Media compression algorithms such as JPEG [22] and MP3 [23] can be considered as a form of semantic aggregation in the context of vehicular sensing. Both semantic (lossy) as well as syntactic (loss-less) aggregation can be applied to an image sensing application where images captured by vehicles moving through the same or even different road segments are aggregated in the network.

Although, data aggregation in sensor networks has been addressed thoroughly in the literature, vehicular sensing is significantly different from sensing in traditional sensor networks in terms of the nature of sensing, node characteristics and network dynamics. To be specific, vehicles are not constrained by battery power and storage; they also exhibit patterned mobility behavior. Furthermore, the topology in vehicular networks changes very rapidly due to large velocity differences among vehicles. As far as sensing is concerned, vehicles can generate data in huge scale; (e.g., image/video sensing generates considerable volumes of data). Thus, vehicular sensing requires specific and innovative data aggregation solutions quite different from those developed for traditional sensor networks.

Vehicle-to-Vehicle (V2V) communication has been leveraged to design proactive and reactive methods for sensing applications in vehicular networks. On the one hand, in V2V based proactive data collection methods [1], [24], the sensed or aggregated data is usually broadcasted periodically to inform other vehicles of the event being sensed, causing significant packet losses [25]. On the other hand, V2V based reactive methods [26] are not efficient because users only have access to the instantaneous content and cannot obtain information of events that occurred earlier. The limitations of both methods can be overcome by using an infrastructure-based approach where vehicles upload sensed data to a base station directly using cellular networks. The sensed data is then aggregated and stored in servers and later downloaded by vehicles when needed. The major drawback of the above approach is that a large number of vehicles trying to upload/download data overload the cellular networks.

In this paper, we propose an in-network aggregation scheme for a vehicular sensing application in which the sensed data is collected, aggregated and then sent to an RSU. The collection of the sensed data in real-time is critical to the performance of vehicular sensing applications. Therefore, the objective of the proposed aggregation scheme is to ensure data collection within a specified delay. In our work, we rely on DSRC RSUs [27] which are connected to a server through a backhaul network. Although, few works on aggregation use DSRC RSUs, they are designed for highway scenarios. In contrast, we focus on data collection in an urban scenario. Using V2V communications, sensed data can be aggregated in the network and using Vehicle-to-Infrastructure (V2I) communications [28], the aggregated data can be uploaded to RSU. However, the greater the number of vehicles involved in sensing, the higher the contention among V2V communication paths (usually the shortest path). Contention increases the delay in delivering data to RSU. As a consequence, a destination RSU may receive a small fraction of the data being sensed within the specified delay.

The proposed in-network aggregation scheme is a novel approach in the sense that it establishes a hierarchy in a dynamic manner for data collection and aggregation in a given region. The dynamic nature of the hierarchy suits the dynamic behavior of vehicular network. We introduce a metric, called *delivery efficiency* (to be defined later) to measure the efficiency of data collection. Moreover, the hierarchy is updated by performing partition and merges in order to achieve an improvement in delivery efficiency. The important contributions of this paper are as follows:

- A state transition method that controls the dynamic update of the hierarchy. Moreover, it also ensures a stable hierarchy when the network condition remains same.
- A novel graph partition algorithm that partitions a given region represented as a connected graph into two smaller regions, also represented as connected graphs.
- Analytical Estimation of delivery efficiency metric.

The rest of the paper is organized as follows. Section II presents related work. Section III describes the proposed scheme. Section IV presents the analytical estimation method. Section V evaluates the proposed scheme via simulations. Finally, Section VI concludes the paper and presents future works.

II. Related Works

In infrastructure-based approaches, vehicles perform sensing as they move along roads and send sensed data to a central server if internet connectivity is available. The server either proactively disseminates the sensed data or disseminates it in response to queries by users. SOCRATES (System of Cellular Radio for Traffic Efficiency and Safety) [29] is one of the earliest contributions that propose the use of 1/2G cellular technology to predict traffic flow based on traffic information received from vehicles. Every vehicle periodically sends its position, speed and travel times to a traffic information center (TIC) that processes the sensed data and sends the current traffic conditions back to the vehicles. Cocar [30] is a client-server system that evaluates the suitability of UMTS for traffic sensing. In order to limit the number of reports sent to the TIC, a vehicle reports an incident (e.g. road accident) to the TIC only if it has not received a similar report within a certain period. ParkNet [3] is another centralized architecture that collects on-street parking information sensed by vehicles moving in streets. One major common drawback of these systems [3], [29], [30] is that the capacity of cellular links is not taken into consideration. When a large number of vehicles attempt to upload their sensed data, voice/video services might suffer in terms of quality of service.

CarTel [31] is a sensing platform that does not require vehicles to use cellular links, rather uses opportunistic relaying through WiFi or Bluetooth to connect to the Internet. It offers a versatile solution by not being constrained by the type of sensed data and hence makes it easy to integrate new sensors. In addition to traffic monitoring, it is also used for environmental monitoring, automotive diagnostics and geo-imaging. A CarTel node can use storage devices such as USB keys and flash memories as “data mules”; it relies on those mules to deliver sensed data in best-effort manner. Because of

its delay-tolerant nature, CarTel cannot be used for real-time acquisition of sensed data.

PeerTIS [32] is an overlay infrastructure-based approach adopted for traffic sensing applications. Vehicles create a peer-to-peer (P2P) overlay over the Internet. They generate travel time reports (timestamp, road segment ID and measured travel time) which are stored in their local storage. To search travel time of a road segment, a lookup scheme similar to the one used in a structured P2P architecture is used. The requested traffic data is published in a particular node according to the lookup mechanism. In an improved version of PeerTIS [33], the street network is represented by a graph, which is partitioned into sub-graphs on detecting a new vehicle in the system. When vehicles perform route planning, they generate queries to obtain travel reports. These queries result in implicit subscription of vehicles to road segments whose travel reports are requested. Vehicles are informed about change in travel time of road segments they are subscribed to. Although, P2P based solutions provide better performance than the approaches using cellular links, maintaining the overlay structure is cumbersome in the highly dynamic vehicular network.

In [34]–[36], infrastructures or RSUs are used to collect traffic data (e.g. traffic density, mean vehicle speed and travel times) sensed by vehicles. RSUs send the collected data to a central server. In [35], few RSUs as task organizers (TO) are deployed along the road. To collect traffic data, a TO broadcasts a message containing its position. Vehicles on receiving the message from TO trigger the sensing task and send the sensed data back to the TO. In order to collect city wide traffic data, this scheme requires RSUs to be deployed along all roads which is not feasible.

To reduce the amount of data transmitted to the RSUs, Salhi *et al.* [34] and Miloslavov and Veeraraghavan [36] propose an in-network aggregation scheme to create fewer content-rich reports out of many sensor readings. In [34], a cluster-based architecture is proposed for the collection of aggregated traffic data along a straight road. A given straight road is divided into a number of small segments; a cluster head (i.e. a vehicle) is elected for each segment. The cluster head aggregates the data sensed within the segment and sends it to the cluster head of the adjacent segment closer to RSU. In the last segment (i.e. the segment closest to RSU), the cluster head sends the aggregated data to this RSU. Using the above schemes [34], [36], it is not possible to collect sensed data in a complex environment (e.g. entire city), as it would involve a huge cost to deploy RSUs on all road segments. In [6], a cyber-physical sensing framework is proposed for urban sensing using drone swarms. The urban sensing scheme proposed in [6] allows independent depots to deploy drones and assign sensing tasks (sensing traffic information, air pollution and noise). A drone can store the sensed data locally which is collected by the depot when the entire swarm returns to the depot. The sensed data is also disseminated to drones in other swarms to relay the data to the depot. In [7], a WiFi lookup scheme is proposed. Vehicles such as bus, patrol cars, and private cars use compressive sensing technique to localize nearby WiFi APs along their routes and upload the WiFi AP lookup data to a server. At the server, the data collected from different vehicles is refined and aggregated to estimate WiFi AP distribution. Vehicles that require Internet access can download the WiFi AP distribution from the server and connect to a nearby WiFi AP in an opportunistic manner.

In infrastructure-less approaches [1], [24], [26], vehicles cooperate to collect, aggregate and disseminate the sensed data in the network. In such approaches, the sensed data is broadcasted up to a certain distance. Each vehicle can aggregate the received data and broadcasts the aggregated data in the network. In case of proactive dissemination [37], [38], the broadcast storm problem results in packet losses; thus, sensed data cannot be delivered to all vehicles especially the ones that are far from the place of the sensed event. Broadcast storm problem is a critical problem for Vehicular Ad Hoc networks (VANET). It results in serious redundancy, contention, and collision when a large number of vehicles try to broadcast the packet at the same time. The broadcast storm problem has been studied extensively in VANETS. Various suppression schemes by combining probabilistic and time-delay-based methods have been discussed in [39]. In most of the works, the main focus relies on improving reliability of periodic beacons (one hop broadcast) [40] and emergency message transmissions (multi-hop broadcast) [40]–[42]. Unlike beacons and emergency messages, the aggregated information is generally broadcasted periodically over multiple hops. Thus, the broadcast storm mitigation schemes need to be revisited from an aggregation perspective. In [43], probabilistic data aggregation is used to eliminate the impact of broadcast storm problem on data transmission and reduce bandwidth consumption. Recently, a broadcast storm mitigation scheme is proposed for safety information broadcast in VANETs, where the mathematical models of single lane and multiple lane roads are used to design a probabilistic flooding scheme [44].

On the other hand, in reactive approaches [26], [45], requested data is obtained with longer delays because of the time spent in collecting sensed data from distant vehicles and performing aggregation. Vehicles only exchange data when data is requested. As a result, it is difficult to obtain data of a past sensed event and the temporal scope of sensed data is limited to a few seconds rather than hours, days or weeks. Furthermore, none of these approaches presents aggregation schemes that can adapt to vehicular density variations.

In [46], a hierarchical aggregation scheme is presented. Although the hierarchy is updated, the need for an update is identified using a simple metric: total number of transmissions. Moreover, the metric does not indicate the efficiency of data collection within a certain delay. Our work is closely related to the Delay Bounded Vehicular Data Gathering (DB-VDG) [47]. The protocol focusses on data collection in urban vehicular sensor networks within a certain delay, also specified as the query life time. Basically, the data collection area is a circular region centered at the base Station. Vehicles moving in the area transmit their sensed data towards the base station by selectively using one of the two methods: data muling and next hop forwarding. A vehicle on receiving data from other vehicles performs aggregation and carries the aggregated data or forwards it to a suitable forwarder. Data muling refers

to a carry-forward paradigm where the data is carried by a vehicle as the vehicle moves along a road. A vehicle chooses data muling method if the time taken by the vehicle to reach the base station is smaller than the remaining query life time. Otherwise, next-hop forwarding is selected in which the vehicle forwards it data to a neighbor that has lowest aggregation level (amount of data carried). The limitation of DB-VDG is that in-network aggregation is performed by vehicles independently and without considering the current network load. As a result, the potential of aggregation in reducing bandwidth is not leveraged properly.

For data transmission, the aggregation schemes rely one of the following protocols: unicast routing protocols [26], broadcast protocols [26], [37] and geocast protocols [47]. In some reactive schemes, the protocol to be used is specified in the query packet. Unicast routing protocols are basically used in case of reactive schemes where the data needs to be dissemination from the vehicle carrying the requested data to the requesting vehicle. On the other hand, broadcast protocols are leveraged to disseminate sensed information both in one hop and over several hops. Geocast protocols provide an efficient way to disseminate information to a large number of vehicles in a specific destination region [20].

III. Proposed Scheme

A. Basic Idea

In this paper, we propose hierarchical in-network data aggregation scheme called Dynamic Hierarchical Aggregation for Vehicular Sensing (DHAVS). Aggregation is performed by a number of vehicles termed as aggregators. An aggregator is elected based on the following two criteria. First, it must be located in close proximity of a higher number of vehicles to prevent a large proportion of data packets from travelling over longer number of hops, thereby avoiding congestion in the network. Second, it must stay on a road segment for the longest amount of time among other candidate vehicles to avoid frequent selection of aggregators, thereby ensuring stability. The detail of the aggregator selection scheme is provided in Section-III. G. The final aggregation takes place at an aggregator that lies within the transmission range of RSUs through which the aggregated data is uploaded to an RSU. The concept of DSRC RSUs has been formalized in [27] according to which an RSU can communicate with the on-board unit (OBU) of vehicles located in its range. RSU is connected to the backhaul network and provides internet connectivity to vehicles. For multi-hop data (i.e. sensed/aggregated information) dissemination, vehicles use intersection based unicast routing. A list of intersections is provided in the header. The list is essentially a shortest hop-count path that will be traversed by the data packet. The packet is thus forwarded following the list until it reaches the destination. More details on the routing protocol are provided in Section-III.H.

We consider that sensing is performed in a given region, called sensing region. A sensing region is defined as a set of geographically connected road segments. In a city, a number of sensing regions can be determined in a way that each sensing region must contain one RSU. Determination of sensing regions is a non-trivial problem and can be solved using an approach similar to the one described in [48]. In our work, we consider that the sensing regions of a city are predetermined. We represent each sensing region as a connected graph $G = (V, E)$, where each vertex $v \in V$ denotes an intersection and each edge $e \in E$ denotes a road segment. If RSU of the sensing region is located on an edge (u, v), the edge is divided into two edges (u, k) and (k, v), where k denotes the position of the RSU. If the RSU is located at an intersection, its position is referred to using the vertex that denotes that intersection.

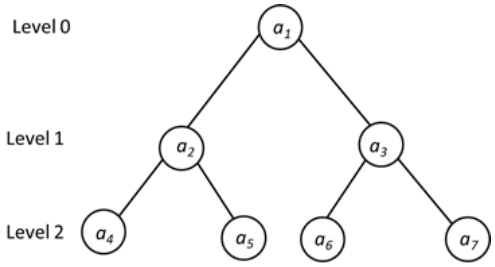

Fig. 1. Aggregation Hierarchy (height = 2).

A naïve and simple way to deal with aggregation for a sensing region is to send the sensed data directly to the aggregator located within the transmission range of the RSU. However, the eventual many-to-one communication would make packets from multiple vehicles share either the same path or a portion of it along their way to the aggregator. This would result in heavy contention among vehicles attempting to transmit or forward data. Moreover, in a highly dynamic vehicular scenario, vehicle density is likely to fluctuate rapidly. As a result, the number of vehicles in a sensing region is subject to change, making the sensing application produce the required data to the aggregator with large delays in some regions. This problem can be solved by using a hierarchy for collection and aggregation of sensed data within a sensing region. In the remainder of this paper, we restrict our discussion to one sensing region as the proposed schemes can be applied to each sensing region independent of the other.

The hierarchy is a binary tree which is established by partitioning the identified sensing region into smaller regions. The identified sensing region is denoted as the root of the hierarchy. When the sensing region is partitioned, two smaller regions are produced which are represented as the child nodes of the root. Each resultant region can again be partitioned into two smaller regions. An aggregator is selected for each node in the hierarchy. Aggregators at the lowest level of hierarchy i.e. leaf aggregators, collect sensed data from vehicles. Leaf aggregators perform aggregation on the collected sensed data and send the aggregated data to their parent aggregators.

At each level, aggregated data received from the lower levels are combined to form new aggregated data. The aggregated data ascend the hierarchy until they reach the root aggregator. The final aggregation takes place at the root aggregator which then sends the aggregated data to the corresponding RSU. Fig.1 shows aggregation hierarchy of height 2. Fig. 2 depicts aggregation at all levels of this hierarchy. The hierarchy needs to be updated with time because changes in network conditions, number of vehicles, topology, etc. affect the delivery

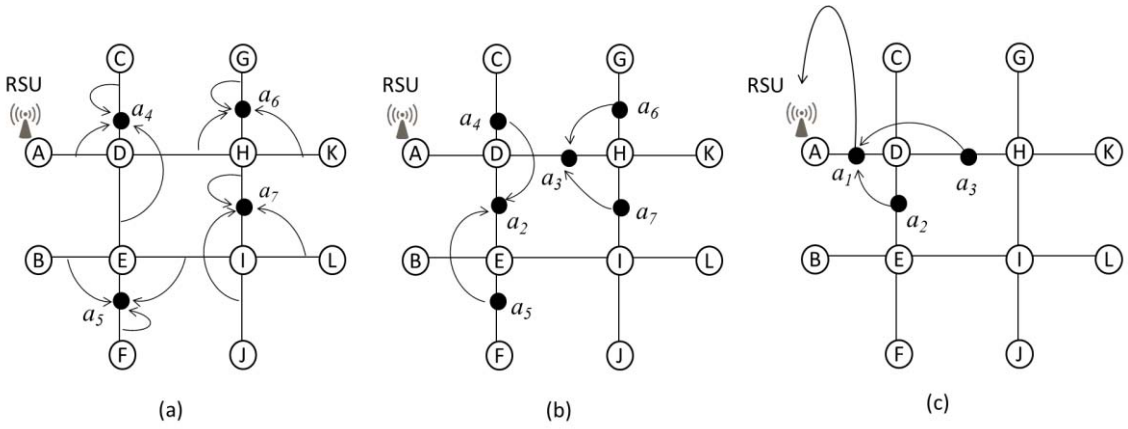

Fig. 2. Aggregation at (a) level 2, (b) level 1, and (c) level 0 of the hierarchy shown in Fig. 1. (Black circles denote aggregators).

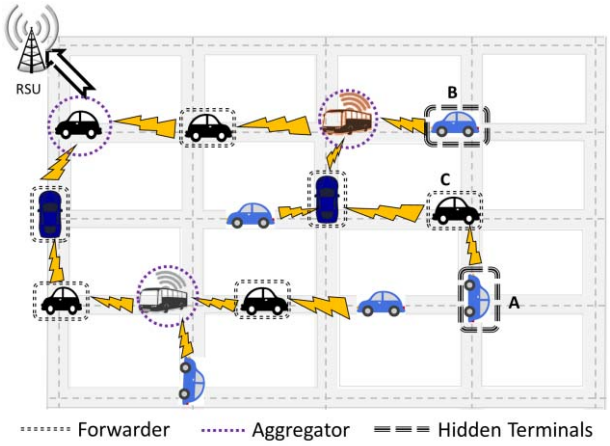

Fig. 3. Aggregation Scenario

TABLE I
DEFINITION OF ENTITIES

Entity	Definition
RSU	A DSRC road-side unit that serves as final destination of sensed data.
Forwarder	A vehicle that forwards messages during multi-hop communications between a vehicle and aggregator or between two aggregators.
Aggregator	A vehicle that serves as an aggregator. It receives sensed data from different vehicles, aggregates and sends the aggregated data to its parent aggregator, if any (i.e. if the height of the hierarchy is greater than 0)
Hidden Terminal	Vehicle A is said to be a hidden terminal with respect to vehicle B when vehicle A is outside the transmission range of vehicle B. Transmissions from A and B result in collision at vehicle C who is in transmission range of both A and B.

of data at the root aggregator. The dynamic update of the hierarchy is achieved by applying one of two operations: a partition operation or a merge operation. In the partition operation, the leaf nodes are partitioned into two child regions; whereas in the merge operation, two leaf nodes are merged back with the parent node making it a leaf node in the hierarchy. The partition and merge operations are described in detail in Section III. E.

In order to decide whether a hierarchy update is necessary, we use the notion of delivery efficiency the formal definition of which is given in section III.C. In section IV, we introduced an analytical model to estimate delivery efficiency. The necessity of estimating delivery efficiency arises mainly because we aim to adapt the hierarchy as quickly as possible. Thus, we estimate the status of the data collection in the next time interval and provide an updated hierarchy at the beginning of the next interval. Detecting performance degradation beforehand helps to update the hierarchy in a timely manner. The dynamic update of hierarchy is achieved using a state transition method described in Section-III. D. It provides triggering rules for partition and merge operations based on delivery efficiency values. Fig. 3 shows some entities used in the proposed scheme and in the analytical estimation. TABLE-I defines all entities.

B. Assumptions

1) Vehicles are equipped with various sensors, GPS and digital road map.
2) An urban area is considered for the vehicular sensing application.
3) In our work, we consider one RSU that hosts a vehicular sensing application. RSUs are deployed at the city intersections. We select one of the RSUs for our application. Each vehicle senses data periodically with an interval, called data collection interval (denoted as T_{img}). We also assume that the application needs to collect sensed data from all vehicles in the designated urban area within a specified delay equal to the duration of data collection interval. Any data received after the delay is discarded.
4) All vehicles are synchronized to the boundaries of data collection interval. The required synchronization is achieved through GPS [49].
5) Each vehicle generates one sensed data packet of fixed size in each data collection interval. Vehicles moving on a same segment may sense same or different events. We assume that each vehicle sends its kinematic profile

along with its sensed data. However, at leaf aggregator these profiles are discarded.

6) Certain vehicles are designated as aggregators which perform aggregation operation on the data packets from different vehicles or from different aggregators. The aggregation operation is basically a compression operation that reduces the amount of data to be transmitted.

C. Hierarchical Aggregation

In the proposed hierarchical aggregation scheme, vehicles start sending their sensed data to the leaf aggregators at the beginning of a data collection interval. The leaf aggregators perform aggregation and send the aggregated data to their parent aggregator, if any. Further aggregation is performed at each parent aggregator on reception of aggregated data from its two child aggregator. Aggregation is thus performed at each level of the hierarchy until the aggregated data reaches the root aggregator which performs the final aggregation operation and sends the aggregated data to the RSU.

Let the root of hierarchy be considered at level 0. The aggregators at level l of the hierarchy will be denoted as a_k and the amount of data they transmit to their parent aggregator is denoted as S_k, where $k=2^l, 2^l+1,\ldots 2^{l+1}-1$. Due to aggregation at each level of the hierarchy, the amount of data transmitted by an aggregator a_k to its parent aggregator is smaller than the amount of data the aggregator receives from its two child aggregators. In our aggregation scheme, the aggregation operation is a compression operation that reduces the amount of data. Note that the aggregation can be semantic or syntactic based on loss-less or lossy nature of the compression operation. We assume a loss-less compression; however the specific compression algorithm depends on the type of event sensed and the requirements of the sensing applications; hence is beyond the scope of this paper. For example, when the sensed data includes images, a suitable loss-less image compression algorithm can be used. We consider that the aggregation operation reduces data using a ratio, called as aggregation factor, denoted as ρ. The aggregation factor is the ratio of the amount of data generated after aggregation to the amount of data before aggregation. For example, in case of images, ρ represents the compression ratio used by an image compression algorithm. For sake of simplicity, we assume a constant value of ρ (e.g. 0.1 and 0.2 are used in our experiments).

The amount of data after the aggregation operation at a leaf aggregator a_k is calculated as

$$S_k = \rho * s * N \tag{1}$$

where N is the vehicle count (i.e. number of vehicles) of the region of leaf aggregator a_k. s is the fixed size of data packet. If a_k is a non-leaf aggregator, S_k is given by:

$$S_k = \rho * (S_i + S_{i+1}) \tag{2}$$

where S_i and S_{i+1} denote the amount of data transmitted by a_k's child aggregators a_i and a_{i+1} respectively.

In our scheme, we measure the efficiency of data collection at the root of the hierarchy using a metric, called delivery efficiency. The formal definition of delivery efficiency is given below:

Definition 1: The delivery efficiency for a time interval is defined as ratio of number of vehicles whose sensed data is received at the root aggregator within a delay $T_s$to the number of vehicles that transmitted sensed data during the interval. It is expressed as follows:

$$\beta(S, T_s) = \frac{N_{recv}}{N_{tot}} \tag{3}$$

where S denotes the region of the root aggregator, N_{recv} denotes number of vehicles whose sensed data is received at the root aggregator within a delay T_s and N_{tot} denotes the number of vehicles that transmitted sensed data.

Let T_u denote the delay incurred in sending aggregated data from root aggregator to the RSU. The delay T_s is given by:

$$T_s = T_{img} - T_u \tag{4}$$

Many factors such as changes in network conditions, number of vehicles and topology may result in lower delivery efficiency. It is necessary to update the hierarchy in order to achieve higher value of delivery efficiency. When the number of vehicles increases in a region, contention increases which in turn increases the number of collisions. Consequently, higher retransmission delays occur during transmission of sensed data which may not be delivered to RSU within the required delay. The higher retransmission delay can be avoided by partitioning region into two smaller regions. One more example of the topology change that lowers delivery efficiency is the scenario where the numbers of vehicles within leaf regions remain same, but are more concentrated near the parent of the leaf aggregator. Because of the aggregation hierarchy, these vehicles must still send their sensed data to the leaf aggregators rather than sending them directly to the parent aggregator which is located in close proximity of them. As a result, sensed data will reach RSU with a significantly higher delay than the required delay. This condition can be avoided by merging the two region into one region and removing the corresponding leaf nodes from the aggregation hierarchy.

However, it is not wise to engage in any kind of update (partition or merge) of the hierarchy if a very small improvement in delivery efficiency is achieved. In this regard, we devise specific conditions that trigger the update. The state transition method in the next section describes these conditions in detail.

D. State Transition Method

We propose a state transition method that controls the dynamic update of the hierarchy. Moreover, it also ensures a stable hierarchy when the network condition remains same. According to this method, at any data collection interval, the aggregation hierarchy remains in one of the four states: initial, partitioned, merged and steady. At the beginning, the hierarchy contains only root and is in the initial state. Afterwards, the state of the hierarchy changes from initial state to either partitioned or merged state. Transition from one state to

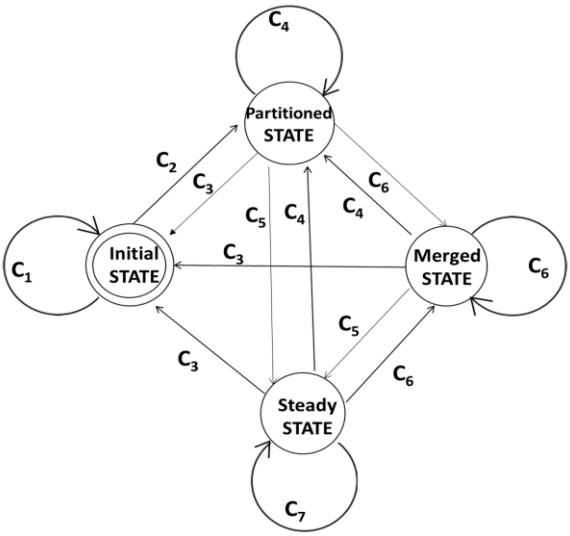

Fig. 4. State Transition Diagram

another is triggered by certain conditions. A transition to the partitioned state is preceded by a partition operation in which a given region is partitioned into two smaller regions. Similarly, a transition to the merge state is preceded by merge operation in which two regions are combined to form a single region. A steady state indicates a stable hierarchy where no update (partition or merge) is performed. The state may change from steady to any of the other three states if the triggering conditions are satisfied. Note that initial state indicates a hierarchy with only root. Thus, a merge operation on a hierarchy of height 1 with partitioned/merged/steady as its current state will lead to initial state. The state transition diagram is shown in Fig. 4. The triggering conditions for change of state are denoted as C_i, i=1,2..7. The triggering conditions are derived based on observed and estimated value of delivery efficiency.

The hierarchy is updated by one or more entities, referred to as updating entity. When the hierarchy consists of a root node only, the root aggregator itself serves as the updating entity. Otherwise, the parents of leaf aggregators serve as the updating entity.

The delivery efficiency is observed at the root aggregator at the end of a data collection interval t and is shared with all updating entities. At the beginning of the data collection interval t+1, the updating entities estimate the delivery efficiency expected for the data collection interval t+1. Then, using the observed value for interval t and estimated value for t+1, the updating entities decide the update operation as well as state of the hierarchy for interval t+1. Specifically, the updating entities determine the estimated values of delivery efficiency expected for interval t+1 for three update operarations:1) No operation, 2) partition, 3) merge, that can be performed on the hierarchy. Then, the update operation as well as the next state of the hierarchy is determined by checking the triggering conditions. TABLE-II shows the delivery efficiency values and some thresholds used in the triggering conditions. TABLE-III shows the triggering conditions (please see Fig. 4) and their meaning.

TABLE II

MEANING OF VARIABLES USED IN TRIGGERING CONDITIONS FOR STATE TRANSITION

Variable	Meaning
h	Height of aggregation hierarchy.
β_{init}	Delivery efficiency observed at the end of the very first data collection interval or at the end of the interval elapsed after the hierarchy enters the initial state from another state.
β_t	Delivery efficiency observed at the end of data collection interval t.
β_s	Delivery efficiency observed at the end of the data collection interval elapsed after the hierarchy enters the steady state from another state.
β_{t+1}^{Nop}	Estimated value of delivery efficiency for data collection interval t+1 if no update is performed on the hierarchy during interval t+1.
β_{t+1}^{Pt}	Estimated value of delivery efficiency for data collection interval t+1 if partition operation is performed during the interval t+1.
β_{t+1}^{Mg}	Estimated value of delivery efficiency for data collection interval t+1 if merge operation is performed during interval t+1.
f_{init}	Fluctuation in delivery efficiency if no operation is performed on the hierarchy during data collection interval t+1. It is given by the absolute difference between β_{init} and β_{t+1}^{Nop}
f_{th}	Threshold for fluctuation in delivery efficiency.
$G_{init} = \beta_{t+1}^{Nop} - \beta_{init}$	Gain in delivery efficiency from the time the hierarchy was in initial state to the data collection interval t+1 if no operation is performed on the hierarchy. It is given by the difference between β_{init} and β_{t+1}^{Nop}
$G = \beta_{t+1}^{Nop} - \beta_t$	Gain in delivery efficiency from data collection interval t to t+1 if the hierarchy remains same (i.e. no operation is performed) during interval t+1. It is given by the difference between β_{t+1}^{Nop} and β_{init}.
$G_s = \beta_{t+1}^{Nop} - \beta_s$	Gain in delivery efficiency from the time hierarchy was in steady state to data collection interval t+1 if no operation is performed on the hierarchy for the interval t+1. It is given by the difference between β_{t+1}^{Nop} and β_s.
$G^{Pt} = \beta_{t+1}^{Pt} - \beta_t$	Gain in delivery efficiency from interval t to interval t+1 if a partition operation is performed on the hierarchy during interval t+1. It is given by the difference between β_{t+1}^{Pt} and β_t
$G^{Mg} = \beta_{t+1}^{Mg} - \beta_t$	Gain in delivery efficiency from data collection interval t to interval t+1 if a merge operation is performed on the hierarchy during interval t+1. It is given by the difference between β_{t+1}^{Mg} and β_t

TABLE III

MEANING OF SYMBOLS THAT DENOTE TRIGGERING CONDITIONS

Symbol	Meaning
C_1	$(f_{init} \leq f_{th}) or\ (f_{init} > f_{th}\ and\ G_{init} \leq G^{Pt})$
C_2	$f_{init} > f_{th}\ and\ G^{Pt} > G$
C_3	$f_{init} > f_{th}\ and\ G^{Mg} > G\ and\ G^{Mg} > G^{Pt} and\ h = 1$
C_4	$f_{init} > f_{th}\ and\ G^{Pt} > G^{Mg} and\ G^{Pt} > G$
C_5	$f_{init} > f_{th}\ and\ g > G^{Pt} and\ g > G^{Mg}$
C_6	$f_{init} > f_{th}\ and\ G^{Mg} > G\ and\ G^{Mg} > G^{Pt}$
C_7	$f_s > f_{th}\ and G_s > G^{Mg}\ and\ G_s > G^{Pt}$

E. Update Operations

1) *Merge Operation:* The merge operation involves simple set operations on two input leaf nodes (i.e. regions) of the hierarchy. It produces a connected graph whose

vertex set and edge set is given by the union of the vertex sets and edge sets of two leaf nodes.

2) *Partition Operation:* We propose Partition Algorithm (PA) to perform the partition operation. The objective of PA is to divide a region into two smaller regions. The region to be partitioned is represented by a connected graph. PA is a bi-directional search algorithm. Bi-directional search was investigated thoroughly by Pohl in [50]. It consists of a search in the forward direction from an initial node and a search in the backward direction from a goal node. It has several applications including finding a shortest path between two points. Running Breadth-First-Search (BFS) in forward and backward directions can lead to an intersection point. Once the searches meet, the shortest path from the initial node to the goal node is obtained by tracing through the vertices that have been visited. In PA, the initial node and goal node are selected as two vertices that are farthest apart in the connected graph. There are several design choices to design the bi-directional BFS. Two choices are described as follows: 1) alternating through the searches, visiting one vertex at a time; 2) running a search till a certain depth and then running the other search till it intersects the first search. The objective of PA is to ensure the size of two regions to be as close as possible (assuming a uniform grid topology for the region to be partitioned).

Thus, we adopt first design choice in the proposed PA. Algorithm 1 shows the pseudo-code of PA. The objective is to traverse edges of the input region in order to produce two connected sub-graphs as the output regions. Note that the input region is represented as a connected graph $G = (V, E)$, where V and E denotes the set of vertices and set of edges in graph G respectively. For each output region, we select a vertex in V at which the traversal begins. A pair of vertices is selected such that the distance between the vertices is the largest among any pair of vertices. For each of the two vertices, an edge is randomly selected among all edges that are incident on it. The traversal starts at the selected edges. E_1 and E_2 denote the set of edges for two output regions. Two queues Q_1 and Q_2 are used to store the edges for the two searches. During the execution of the while loop in Algorithm 1, edges are enqueued and dequeued to and from the queues. After an edge is dequeued from a queue, it is added to the edge set of the corresponding output region. Then, its neighboring edges that are not yet visited are enqueued to the queue. The set of neighboring edges of an edge e is denoted as $N(e)$.

In Algorithm 1, each edge is enqueued and dequeued exactly once. The Enqueue and Dequeue operation take place in O(1) time. Thus, Algorithm 1 requires O($|E|$) time to enqueue and dequeue all edges of the input graph G. We assume that an edge-adjacency list is used in which the neighbors of an edge are found in O(1) time. As a result, the total time needed for all edges is O($|E|$). The overall time complexity of Algorithm 1 is O($|E|$). The space complexity of Algorithm 1is O($|E|$). This is because the queues as well as the edge-adjacency list have a space requirement of (O($|E|$) each. Fig. 5(a) shows a connected graph in which vertex A and vertex L have the maximum separation distance among all pairs of vertices. For A and L, the start edges *e1* and *e2* are selected. When we apply breadth-first traversal to edges *e1* and *e2*, we obtain the output regions *r1* and *r2* respectively. Regions *r1* and *r2* are shown in Fig. 5(b). In Fig. 5 (c), execution of PA partitions *r1* into regions *r1,1* and *r1,2*. Similarly, *r2* is partitioned into regions *r2,1* and *r2,2*.

Algorithm 1 Partition Algorithm (PA)

```
    Initialization:
1.  Select a vertex u1 and vertex u2 s.t. Dist (u1, u2) is
    maximum
2.  Randomly select edge e1 incident on u1
3.  Randomly select edge e2 incident on u2
4.  E1=E2=φ
5.  Q1 = Q2 = φ
6.  for ∀w ∈ E − {e1, e2}
7.      visited[w] = false
8.  end for
9.  Enqueue(Q1,e1)
10. Enqueue(Q2,e2)
11. visited[e1] = true
12. visited[e2] = true
13. while either Q1 or Q2 is not empty
14.   e = Dqueue(Q1)
15.   E1 = E1 ∪ {e}
16.     for ∀w ∈ N(e)
17.         if visited[w]! = true then
18.             Enqueue(Q1,w)
19.             visited[w] = true
20.         end if
21.     end for
22.   e = Dqueue(Q2)
23.   E2 = E2 ∪ {e}
24.     for ∀w ∈ N(e)
25.         if visited[w]! = true then
26.             Enqueue(Q2,w)
27.             visited[w] = true
28.         end if
29.     end for
30. end while
```

1) *Example:* An example of merge and partition operations is illustrated in Fig. 6. In Fig. 6 (a), once *a3* decides a partition operation; it communicates its decision to the leaf aggregator *a6* and *a7* which then initiate aggregator selection for their newly formed regions by broadcasting a message. Once aggregators are chosen, vehicles will send their sensed data to the new aggregator. Fig. 6 (b) shows a merge operation. Once *a3* decides merge operation, it broadcasts this decision as well as its own position in regions of the leaf aggregator *a6* and *a7*. Vehicles in these on receiving the broadcast message are informed of the new aggregator (i.e. *a3*) to which they will send their sensed data.

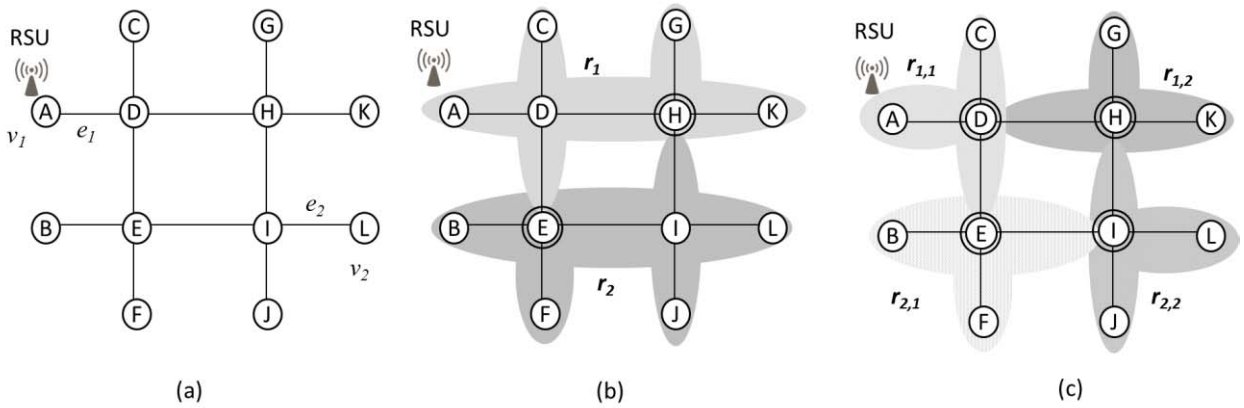

Fig. 5. Partition Algorithm (PA) (a) Connected Graph, (b) Output regions r_1 and r_2 by partition operation on connected graph (c) Output regions $r_{1,1}$ and $r_{1,2}$ by partition of r_1 and output regions $r_{2,1}$ and $r_{2,2}$ by partition of r_2.

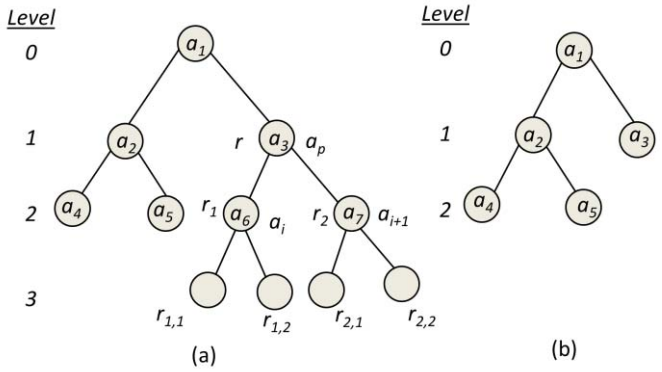

Fig. 6. Execution of DUA at aggregator a_3 of hierarchy shown in Fig. 1. Sub-tree rooted at a_3 (a) after partition operation, (b) after merge Operation.

Algorithm 2 Delivery Efficiency and Delay Estimation (DEE)

```
1.  for each vertex u ∈ V_L
2.      N[u] = 1
3.  end for
4.  n_min = 1
5.  while(true)
6.      for i = 1,2....n_slot
7.          for each vertex u ∈ V_L
8.              if N[u] ≥ 1 and SL[u] = i then
9.                  N[u] = N[u]-n_min
10.                 N[Next[u]] = N[Next[u]] + n_min
11.                 if Next[u] = u_dst then
12.                     N_recv = N[Next[u]]
13.                 end if
14.                 D_ret[u] = FindDelayRet (u)
15.             end if
16.         end for
17.         T_d[i] = T_slot + FindMaxDelayRet (i)
18.         T_delay = T_delay + T_d[i]
19.         if T_delay ≥ T_S then
20.         return N_recv
21.         end if
22.     end for
23.     n_slot = Update_Contention( )
24.     n_min = FindMin( )
25. end while
```

F. Delivery Efficiency Calculation

As described in Section-III.D, the triggering rules of state transition are based on two types of delivery efficiency: actual value observed at the root aggregator at the end of data collection interval and an estimated value that is expected for the next data collection interval. Once the actual value is obtained, the root aggregator shares it with all updating entities; whereas the estimated values are computed by the updating entities independently.

The root aggregator computes the actual value of delivery efficiency using the values of N_{recv} (i.e. number of vehicles whose data have been received at the root aggregator) and N_{tot} (i.e. number of vehicles that transmitted their sensed data). In order to allow root aggregator know N_{recv} each aggregator computes the total number of vehicles whose messages are aggregated and send this information its parent aggregator. The parent aggregator on receiving such information from its two child aggregators computes the sum and sends it to its parent. Similarly, to allow root know N_{tot}, each leaf aggregator sends the total number of vehicles in its region to its parent aggregator. Root aggregator computes N_{tot} as the sum of the information about number of vehicles received from its two child aggregators. In order to compute the estimated delivery efficiency, the updating entities requires the predicted vehicle count of all road segments within its region for the current data collection interval. Note that the leaf aggregators piggyback this information when they send their aggregated data to their parent.

The method to predict vehicle count is described later in this section. Using the received predicted vehicle counts, the updating entities use algorithm DEE (i.e. Algorithm 2) to compute the estimated value of delivery efficiency. In particular, they will compute the fraction of vehicles whose sensed data can reach the root aggregator after being aggregated in the hierarchy within a certain delay, T_s.

Vehicle Count Prediction: Estimation of delivery efficiency requires vehicle count information of all road segments in the next data collection interval. We propose a simple method that uses vehicle's kinematics profile in the current data collection interval and moving average of vehicle arrival rate to predict vehicle count in the next data collection interval. The kinematic profile of a vehicle includes its position, speed and acceleration. If $n(t)$ denotes the vehicle count of a road segment at current data collection interval t, vehicle count at time t+1 is given as:

$$n(t+1) = n(t) + \lambda_a(t+1) + \lambda_d(t+1) \tag{5}$$

where, $\lambda_a(t+1)$ represents the number of vehicles that arrive at the road segment in the time interval t+1 and $\lambda_d(t+1)$ represents the number of vehicles that depart the road segment in the time interval t+1. $\lambda_a(t+1)$ is obtained as the simple moving average [51] of most recent $k$observations on vehicle arrival rates. In order to calculate the initial moving average, we allow first few data collection interval to pass before vehicles start to sense data. $n(t)$ is thus determined using the kinematic profiles. Also, using speed and acceleration values, vehicles that will no longer stay in the same road segment in the next data collection interval can be identified and hence $\lambda_d(t+1)$ can be obtained.

G. Aggregator Selection

The road segment, where aggregation is performed, termed as aggregation road segment, is selected using a method that considers vehicle count of all road segments. According to the multi-hop dissemination strategy, data packets carrying aggregated/sensed data are broadcasted along the shortest hop-count path; thus, multiple data packets are disseminated along the same road segment. If the aggregator is located in a region of high vehicle density, then delivery efficiency will increase; indeed, in this case, a large proportion of data packets will travel few hops resulting in packets being delivered in shorter delays. Based on this observation, we adopt a COG (Centre-of-Gravity) method to compute the position of aggregator. Let n_{ist} denotes the number of intersections. For intersection i, the coordinates are denoted by (X_i, Y_i). The number of data packets, denoted by W_i, that travels through intersection i is measured by looking at the path specified in the packet header. The weighted COG is denoted by (X_c, Y_c) and is calculated as follows:

$$X_c = \frac{\sum_{i=1}^{n_{ist}} W_i X_i}{\sum_{i=1}^{n_{ist}} W_i}, Y_c = \frac{\sum_{i=1}^{n_{ist}} W_i Y_i}{\sum_{i=1}^{n_{ist}} W_i} \tag{6}$$

If point (X_c,Y_c)lies on a road segment, then the latter is chosen as the aggregation road segment. If (X_c,Y_c) does not lie on a road segment, then the road segment closest to $(X_c,Y_c)$is selected as the aggregation road segment. Let (X_s, Y_s) denotes a point on a road segment where the perpendicular drawn from (X_c, Y_c) intersects the road segment. Then, the closest segment is determined as the segment whose perpendicular has the smallest distance to (X_c,Y_c)among the candidate road segments. Once the aggregation road segment is selected, we select an aggregator by considering a small cell around the center of the aggregation road segment. Vehicles located in the cell exchange messages (e.g. one hop beacons) to elect one of them as aggregator. In particular, a vehicle which is expected to stay in the cell for the longest time period is elected as aggregator. Vehicle having lowest velocity can stay longer compared to other vehicles. The size of the cell is determined based on maximum velocity of vehicles, and the duration of data collection interval. We consider that the length of the cell is given by the distance (e.g., 100 m) travelled by a vehicle at the maximum speed (e.g., 20m/sec) for a time period of few (e.g. 1) data collection intervals (e.g., 5 sec). Vehicles located in the cell exchange messages with each other to elect one of them as the aggregator. The aggregator informs existing vehicles in the cell to elect a new aggregator before it leaves the cell. The procedure to elect a new aggregator is based on *case 2* of the coordinator selection procedure proposed in [49]. Once a new aggregator is elected, the current aggregator transfers all data collected to the new aggregator if the data have not yet been sent to the parent aggregator and other information such as vehicle arrival observations.

H. Multi-Hop Communication

In our scheme, multi-hop communication is needed, when 1) when a vehicle sends its sensed data to an aggregator, 2) when one aggregator sends aggregated data to another aggregator. Unicast routing protocol is best suited for our multi-hop scenario as the communication takes place between two distinct entities. In this regard, we use Backbone-Assisted-Hop Greedy (BAHG) routing protocol [52] which is proposed in one of our previous works. BAHG is an intersection-based unicast routing protocol in which a routing path is computed as a sequence of intersections. BAHG selects a routing path that has shortest hop-count as well as highest connectivity. Thus, it offers higher performance than other intersection-based unicast routing protocols that rely on road-metric distance, connectivity [53] or both [54]. This is because a routing path selected using only connectivity metric may involve more hops leading to longer delays. Similarly, routing path having shortest road-metric distance may not necessarily provide lowest hop-count because of numerous intermediate intersections in urban scenario.

IV. Estimation of Delivery Efficiency

The parameter N_{tot} in the expression of delivery efficiency given by Eq. (3) is computed as:

$$N_{tot} = s * n(t+1) \tag{7}$$

where $n(t+1)$ is obtained using Eq. (5) and s is the fixed size of data packet. $N_{recv}$is computed using DEE algorithm described in this section. Since we use CSMA/CA (in 802.11p/DSRC standard [55]) as the underlying channel access scheme, we need to take into account contention that may result in packet collisions causing longer delays. In a sensing region with a high number of vehicles, the delivery efficiency will be

TABLE IV
SUMMARY OF NOTATIONS

Notation	Meaning
$N[u]$	Current load (size of packet multiplied by number of packets to be transmitted) of vertex u
$SL[u]$.	Time slot assigned to vertex u
$Next[u]$	Next-hop forwarder of vertex u
$N[Next[u]]$	Current load of next-hop forwarder of vertex u
n_{min}	Minimum load
n_{ret}	Average number of times a packet is retransmitted following a collision
t_x	Transmission time of a packet
N_{col}	Expected number of packets that undergo collision
T_{slot}	Duration of transmission of n_{min}/s packets
n_{slot}	Total number of time slots in a schedule.
S	Packet Size

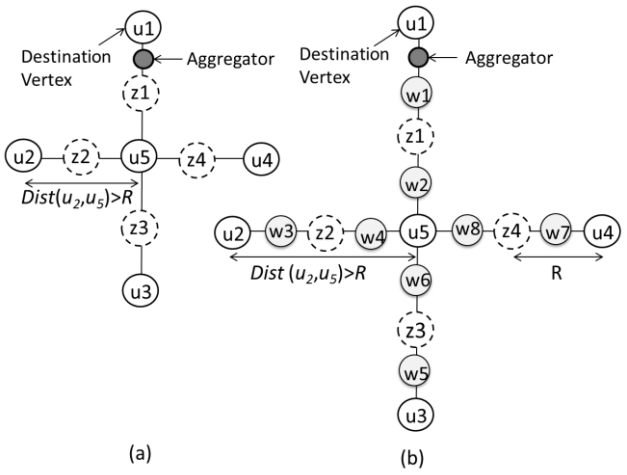

Fig. 7. Communication Link graph CLG (a) $Dist(u_j, u_{j+1}) > R$ (Without W_k vertices), (b) $Dist\ (u_j, u_{j+1}) > R$ (with W_k vertices).

low as packets from all vehicles cannot reach the aggregator within the required delay.

The network dynamics that occur in CSMA/CA can be captured by a combination of perfect scheduling MAC and expected number of collisions in the network. In a perfect scheduling MAC, no two transmissions are allowed to interfere with each other. First, we estimate the delay considering a perfect scheduling. Then, by adding the retransmission delay due to packet collisions to the above delay, the delay incurred using CSMA/CA is obtained. TABLE-IV shows the meaning of important parameters used in this section.

A. Transmission Using Perfect MAC Scheduling

A region S is a connected sub-graph $G_S = (V_S, E_S)$, $V_S \subset V$, $E_S \subset E$ of graph $G = (V, E)$. Graph G denotes the entire sensing region under consideration, where each vertex $v \in V$ and each edge $e \in E$ denote an intersection and a street (i.e. road segment) respectively. Let us refer to the vehicles that generate data as the source vehicles/source vertices. Each source vehicle computes a path to the destination vehicle/destination vertex (i.e. the aggregator). The path is a sequence of vertices $u_i \in V_S$. If the source vehicle and/or destination vehicle are not located at the intersections, but located along a road segment, then they are represented using the vertex that denotes their closest intersection.

Definition 2: A Communication Link Graph (CLG) is defined as the graph G_L=(V_L, E_L), where each vertex is either a group of vehicles or a forwarder.

CLG is constructed from G_S as follows. We denote the Euclidean distance between two vertices $u \in V_S$ and $v \in V_S$ as $Dist(u, v)$. For each source vertex, the path P_i is the shortest hop-count path from the source vertex to the destination vertex. P_i is represented as a sequence of vertices. For any two consecutive vertices u_i and u_j along path P_i, if $Dist(u_i, u_j) > R$ we consider that forwarders are placed R distance apart where R is the transmission range of a vehicle and hence we introduce a new vertex denoted by z_n, where $n = 1, 2, \ldots \lfloor Dist(u_i, u_j)/R \rfloor$. Then, the path segment between vertices u_i and u_j including them is the sequence $< u_i, z_1, z_2 \ldots z_{\lfloor Dist(u_i,u_j)/R \rfloor}, u_j >$.

We repeat this procedure for all source vertices. Please note that, paths of two or more source vertices may have common pair of consecutive vertices. Thus, the above procedure is repeated once for each distinct pair of consecutive vertices. Next, we consider the group of vehicles located between two consecutive vertices in the above sequence as a new vertex w_k. These are the vertices that generate data packets. Since each vehicle generates one data packet in each data collection interval, the number of packets generated at vertex w_k is equal to the number of its constituent vehicles. Thus, the new sequence will be $< u_i, w_k, z_1, w_{k+1}, z_2, w_{k+2}, \ldots. z_{\lfloor dist(u_i,u_j)/R \rfloor}, w_{k+\lfloor dist(u_i,u_j)/R \rfloor}, u_j >$.

Packets received by a forwarder located at u_i are forwarded to u_j through forwarders located at $z_1, z_{2..}, z_{\lfloor dist(u_i,u_j)/R \rfloor}$. Similarly, packets originated at vertex w_k are also forwarded until the packets reach u_j. Fig. 7 (a) and (b) show the graph CLG obtained from a sub-graph G_S when $Dist(u_i, u_j) > R$.

We schedule transmissions in a way that interferences from hidden terminals are avoided. The required schedule depends on the number of hidden terminals which varies based on whether the packet is intended for a forwarder along a road segment or is intended for a forwarder at an intersection. Packet collisions in these two cases are described using the CLG as follows:

Case 1: Collisions along a road segment: Along a road segment, collisions occur only at intermediate forwarders. Given three forwarders, z_n, $z_{n+1}$and z_{n+2} along a road segment, z_n and z_{n+2} are hidden to each other. Hence, transmissions by z_n and z_{n+2}, at the same, will result in collisions at z_{n+1}. Furthermore, some vehicles located between z_n and z_{n+1} are hidden to some of the vehicles located between z_{n+1} and z_{n+2}; thus, packets forwarded to z_{n+1} by a vehicle located between z_n and z_{n+1} may be lost due to simultaneous transmission from vehicles located between z_{n+1} and z_{n+2}. Thus in CLG, w_k and w_{k+1} are also considered hidden to each other.

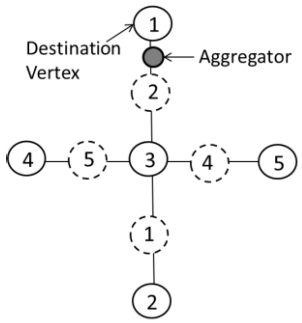

Fig. 8. Scheduling (i.e. time slot assignment).

Case 2: Collisions at intersection: Collisions occur at an intersection, when forwarders or vehicles located on different road segments transmit packets simultaneously to forwarder(s) at the intersection that connects those road segments. Even if forwarders/vehicles of different road segments may be physically located within each other's carrier sense range, obstacles, such as buildings, may prevent them from hearing or detecting each other's transmissions. For example, in CLG shown in Fig.7. (b), z_1, z_2, z_3 and z_4 are hidden to each other.

Definition 3: We define conflict Graph G_C=(V_C, E_C) where V_C= {u: u∈ V_L} and E_C= {(u, v): u and v are hidden to each other}

Definition 4: A time slot, $T_{slot}$is a duration needed to transmit a given load (size of a given number of packets indicated as number of bits).

To achieve perfect scheduling i.e. to avoid collisions, the communication link graph CLG is transformed into a conflict graph CG as follows. All vertices of CLG will remain in CG. However, edges are added between vertices which are hidden from each other according to the collision scenarios illustrated in case 1 and case 2. For each vertex u, an edge is added from u to vertex v, if u and v are hidden to each other. After constructing CG, the schedule is obtained by finding a solution to the vertex coloring problem. Vertex coloring problem is the problem of assigning colors to vertices of a graph in a way that no two adjacent vertices share the same color; each color corresponds to a distinct time slot. It is known that vertex coloring problem is NP-complete. Thus, we adopt iterative greedy heuristic algorithm [56] to find a schedule. Once the schedule is obtained, we assign time slots to the vertices in CG. Fig. 8 shows the assignment of time slots for the graph shown in Fig. 7 (a).

1) Algorithm in Detail: We use DEE algorithm to find the value of N_{recv}. We compute the amount of data received at the destination within a given delay. Transmissions start at source vertices w_k, w_{k+1},..and the packets are forwarded through vertices z_1, $z_{2....}$ along a road segment of the computed shortest paths.

Algorithm 2 shows the pseudo-code of DEE. Using a perfect MAC schedule (see details above), multiple transmission rounds are needed to send packets to the destination, where the duration of a transmission round is given by the total number of time slots in the schedule. In CLG, the source vertices ω_k have different number of packets to send as they might have different number of constituent vehicles. As a result, different forwarders have different load. In each transmission round, the number of packets transmitted by each vertex is equal to the smallest number of packets transmitted by a vertex. We denote the minimum load as n_{min}. For a given transmission round, the duration of time slot i.e. T_{slot} is computed as the time needed to transmit the minimum load. In each transmission round, vertices that have been assigned time slot for transmission transmit exactly n_{min}/s packets (s is the packet size). When a vertex transmits n_{min}/s packet, its load is updated by subtracting n_{min} from its current load; similarly, when n_{min}/s packets transmitted by a vertex is received by its next-hop forwarder, the load of the latter is updated by adding n_{min} to the latter's current load. If a vertex is an aggregator, then the load due to packets received from vehicles or from other aggregators is reduced by the aggregation factor.

In each transmission round, the retransmission delay (see next sub-section for details) is obtained for each vertex that has been assigned time slot to send data. The function FindDelayRet(u) in step 14 of Algorithm 2 uses (12) to obtain the retransmission delay for a vertex u. Retransmission occurs when a packet is not transmitted successfully due to collision. The function FindMaxDelayRet(i) in step 17 of Algorithm 2 is used to find the maximum value of retransmission delay during time slot i. The actual delay of transmission is obtained by adding the maximum retransmission delay to the time slot duration T_{slot}. When the delay is greater than or equal to the required delay T_s, the number of packets received at the destination is returned as N_{recv}. Delivery efficiency is then obtained by using the value of N_{recv} in (3).

At the end of each transmission round, the conflict graph is updated by eliminating the vertices (group of vehicles) that have already transmitted their packets. A new schedule is computed on the basis of the updated conflict graph. The function Update_Contention () in step 23 of Algorithm 2 is used to update the conflict graph as described in Section-IV.A. Once the conflict graph is updated, a collision free schedule is determined using a greedy vertex coloring algorithm. The function Update_Contention () returns the total number of time slots (n_{slot}) in the schedule. In step 24 of Algorithm 2, the function FindMin() is used to determine the minimum load (n_{min}) which is the smallest value of current load among all vertices (i.e. N[u]).

Updating the conflict graph and finding a schedule (using greedy vertex coloring algorithm [56]) are the most expensive operations in Algorithm 2. Both have the time complexity O($|V_L|^2$), where $|V_L|$ is the total number of vertices in the conflict graph. In Algorithm 2, the above two operations are repeated for each hop (i.e. a transmission round). Thus, the time complexity of Algorithm 2 is O($H*|V_L|^2$), where H is the maximum number of hops along the path between a source vertex and a destination vertex. The space complexity of Algorithm 2 is O($|V_L|$).

B. Retransmission Delay

In the following, we present the details to compute retransmission delay for vertex u in the graph CLG. We consider the following assumptions.

1) A collision involves two packets. In [57], it is shown that the number of collisions involving 3 or more packets is negligible.
2) Each packet is transmitted with probability $p = 1/cw$ where cw is the contention window.
3) Vertex u has m hidden terminals. The numbers of packets transmitted by the hidden terminals are denoted by $n_1, n_{2,} n_m$.
4) During a time slot, vertex u sends n packets.
5) $P(i)$ denotes the probability that i packets of vertex u undergo collision.
6) n_{ret} is the average number of times a packet is retransmitted following a collision.
7) t_x is the transmission time of sending a packet in wireless channel. It includes propagation delay, packet transmission time, Distributed Coordinated Function Inter-frame Space (DIFS) and Short Inter-frame Space (SIFS) periods and a small back-off delay.

The expected number of packets sent by vertex u that undergo collisions can be expressed as:

$$N_{col} = \sum_{i=1}^{n} i * P(i) \tag{8}$$

$P(i)$ is closely related to the number of hidden terminals of vertex u and the number of packets transmitted by its hidden terminals. Thus, the probability that only one packet of vertex u undergoes collision is given by:

$$P(1) = \binom{n}{1} * \sum_{i=1}^{m} \binom{n_i}{1} * p^2 \tag{9}$$

$P(2)$ is obtained as:

$$P(2) = P(1) * \binom{n-1}{1} * \sum_{i=1}^{m} \binom{n_i - 1}{1} * p^2 \tag{10}$$

The general expression for P(i) is thus given as:

$$P(i) = P(i-1) * \binom{n-i+1}{1} * \sum_{i=1}^{m} \binom{n_i - i + 1}{1} * p^2 \tag{11}$$

The retransmission delay for vertex u is obtained as follows:

$$T_{ret} = n_{ret} * N_{col} * t_x \tag{12}$$

C. Numerical Results

In this section, we show the limitation of having no hierarchy using numerical results. Let us consider a hierarchy that consists of only the root. In particular, we investigate the impact of required delay, total vehicle count (i.e. total number of vehicles in the network) and size of the region on the delivery efficiency at the root. For evaluation, we consider two scenarios: region 1 which is an 800 × 800 m^2 grid road network with 25 intersections and region 2 which is an 800 × 400 m^2 grid road network with 25 intersections. The length of each road segment is equal to 200m. In order to validate the proposed analytical expression of delivery efficiency, we run simulations to compute delivery efficiency. Details about simulation are provided in Section-V.A.

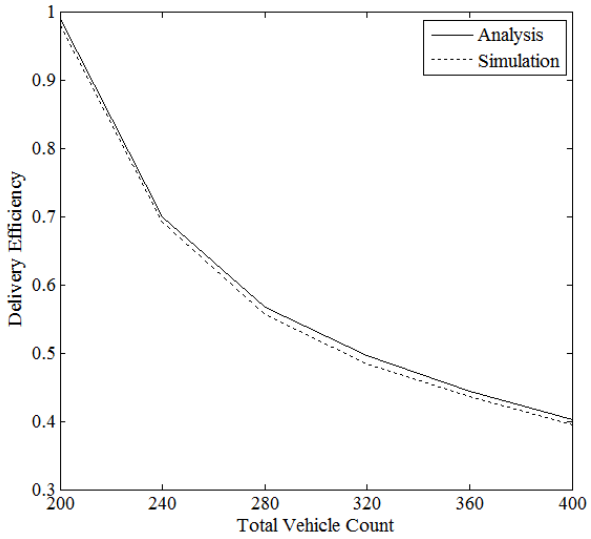

Fig. 9. Delivery efficiency vs Total vehicle count (Region 1, T_s =450 units).

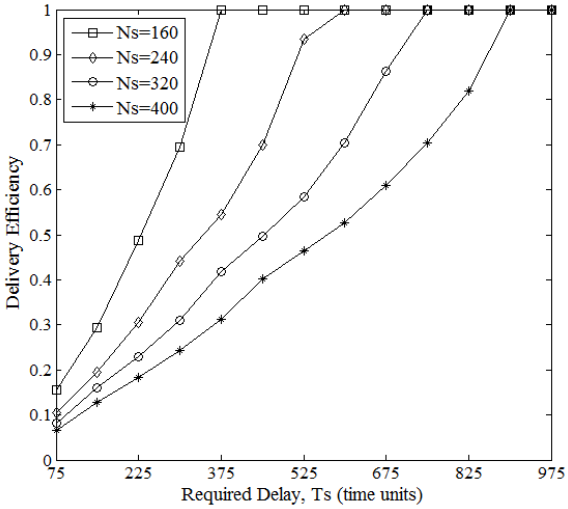

Fig. 10. Delivery Efficiency vs Required Delay (Region 2).

Fig. 9 shows the variation of delivery efficiency with the number of vehicles for region 1. We observe that there is an almost perfect match between the analytical and simulation based results; this proves the accuracy of the proposed analytical expression to compute delivery efficiency. Fig. 9 also shows that delivery efficiency decreases significantly when the number of vehicles increases; for example, increasing the number of vehicles from 200 to 240, in the sensing region, degrades delivery efficiency from 0.98 to 0.7.

Fig. 10 shows delivery efficiency for different values of required delay, T_s for region 2. It also shows delivery efficiency for different numbers of vehicles for each value of T_s. The parameter N_s in Fig. 10 denotes the number of vehicles. We observe that, for a given number of vehicles, delivery efficiency improves as T_s increases. This is expected as packets have a higher lifetime and the packets whose transmissions are

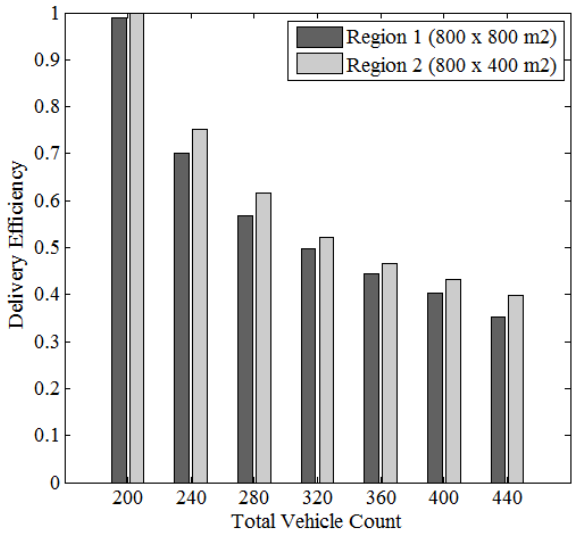

Fig. 11. Delivery Efficiency vs Total vehicle count (T_s =500 units).

delayed in the network, either due to congestion or large hop-counts, will eventually reach aggregator.

Fig. 11 shows the variation of delivery efficiency with the size of sensing region. We observe that delivery efficiency in 800 $\times$ 800m^2 region is smaller than that obtained in 800 $\times$ 400m^2 region for any number of vehicles. This strongly supports the adoption of aggregation hierarchy in which a region is partitioned into smaller regions in order to achieve higher delivery efficiency. A large region attains lower delivery efficiency due to transmission of sensed data over a large number of hops. Instead, if the region is partitioned and the sensed data within each resultant region can be aggregated before forwarded to a final aggregator, the delivery efficiency will improve.

V. Performance Evaluation

A. Simulation Setup

In this section, we investigate the performance of the proposed scheme DHAVS using ns-2 simulator [58]. We used version ns-2.33, which is modelled after the MAC and PHY layer specifications of 802.11p [55]. We compare DHAVS with a baseline scheme in which a static hierarchy is used for data collection and a state-of-the-art aggregation scheme DB-VDG [47]. We refer to baseline scheme as SHAVS (Static Hierarchical Aggregation for Vehicular Sensing). We implemented SHAVS for two hierarchies: one with only a root (i.e. height is 0) and the other with two levels (i.e. height is 1). SHAVS is referred to as SHAVS (h =0) when the hierarchy height is 0 and SHAVS (h =1) when it is 1. For DHAVS, f_{th} and ρ are set to 0.1 and 0.2 respectively.

The simulation scenario represents an urban road topology which consists of 800 x 800 m^2 grid. The grid has five vertical streets, five horizontal streets and 25 intersections. The length of each road segment (i.e. the segment between two adjacent intersections) is 200m. Each street has two lanes, one lane in each direction. We assume presence of one RSU at one of the corners of the simulation area. Note that the position of RSU is independent of the proposed scheme. We used IMPORTANT mobility generator tool [59] to generate vehicular mobility traces according to Manhattan mobility model. The Manhattan mobility model was introduced in [60] to model vehicle movement in a grid topology. This model uses a probabilistic approach to allow vehicle movement at an intersection, where a vehicle continues to move straight with probability 0.5, turn left or right with probability 0.25. The velocity of a vehicle is determined in a periodic manner with a period length of 1s. In particular, the vehicle speed at a time slot depends on the velocity and acceleration of the vehicle in the previous time slot. More details on this model can be found in [59] and [60].

TABLE V
Simulation Parameters

Parameter	Value
Number of Vehicles	200-400
Vehicle Speed	15m/s-25m/s
Vehicle Acceleration	2.5m/s2
Transmission Range	300m
Data Packet Size (Bytes)	1000
Data Collection Interval	5 Sec, 10 Sec, 15 Sec
Channel Propagation	Nakagami-m
Data Rate	6 Mbps
PHY and MAC	IEEE 802.11p
Transmission Range(Vehicles and road-side unit)	300m
Simulation Time	200 sec

Velocity of vehicles varies between 15m/s and 25m/s; the acceleration is set to 10% of maximum velocity. Different scenarios are generated by varying the number of vehicles in the network. We run 10 experiments for each scenario and summarize the results in graphs. We consider data sensing application for the evaluation; it was simulated by generating packets of high payloads. Each vehicle sends the sensed data periodically in each data collection interval. For realistic propagation, we used a Nakagami-m channel model to reflect the wireless channel in an urban setting. The "m" parameter of the Nakagami distribution is responsible for controlling the fading intensities and its value depends on the distance between the transmitter and the receiver [61]. We used dataset 2 provided in [61] for the "m" parameters of Nakagami distribution. Other simulation parameters are listed in TABLE-V.

B. Performance Metrics

1) *Average Delivery Efficiency*: It is defined as the average of delivery efficiency observed at the root aggregators for all data collection intervals.
2) *Maximum Height of Hierarchy*: It is the maximum height attained by the aggregation hierarchy during the simulation period.
3) *Total Number of Bits Transmitted:* It is defined as the total number of bits transmitted during the simulation period. This metric considers the data packets that carry sensed data or aggregated data.

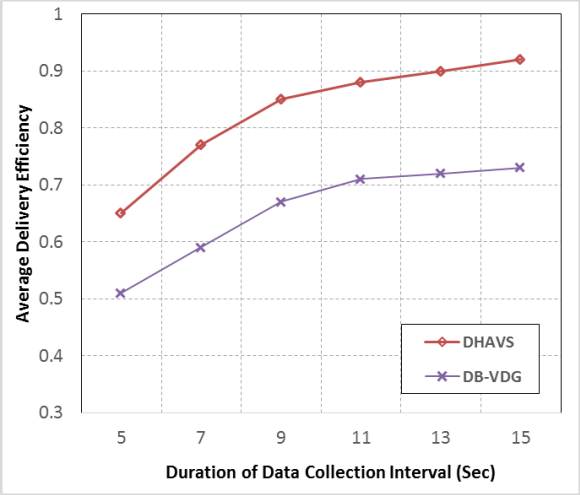

Fig. 12. Average Delivery Efficiency (Number of Vehicles =320).

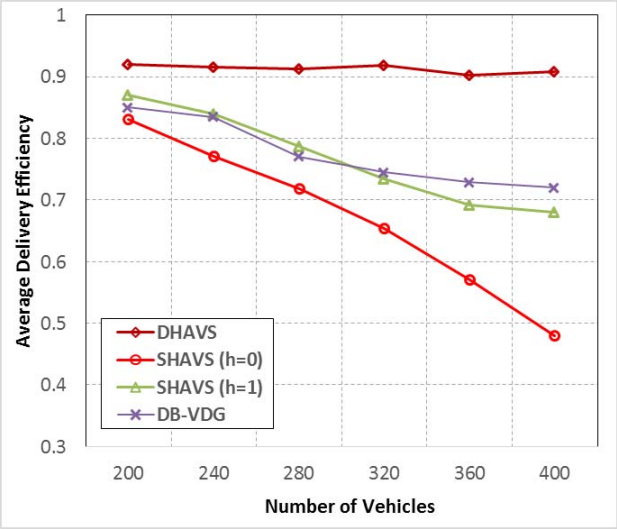

Fig. 13. Average Delivery Efficiency Data Collection Interval=10Sec).

C. Results and Discussions

Fig. 12 shows the average delivery efficiency of DHAVS and DB-VDG with respect to data collection intervals for a scenario of 320 vehicles. We observe that with increase in the duration of data collection interval, an elevation in average delivery efficiency occurs. The reason for this behaviour is that a higher delay threshold allows data packets to reach root aggregator even if using increased number of hops. Overall, DHAVS outperforms DB-VDG irrespective of the duration of data collection interval. This is due to use of aggregation hierarchy in DHAVS which gets updated dynamically in order to improve the delivery efficiency. The reason for lower delivery efficiency in DB-VDG is that it neither employs a hierarchy nor uses a systematic aggregation scheme with meticulously selected aggregators.

Fig. 13 shows the impact of number of vehicles on average delivery efficiency of DHAVS, SHAVS(h=0), SHAVS(h=1) and DB-VDG. For SHAVS (h=0) and SHAVS (h=1), we observe a reduction in average delivery efficiency with

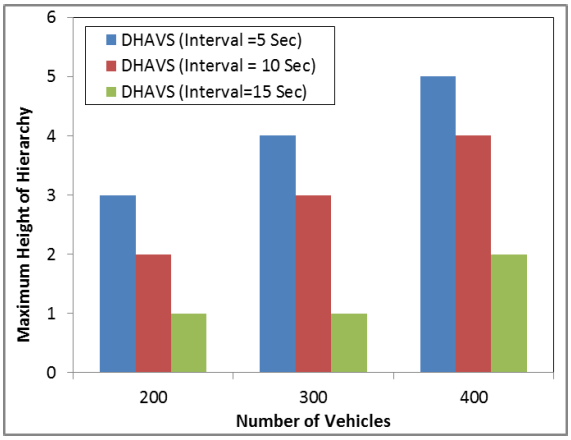

Fig. 14. Maximum Height of Hierarchy in DHAVS Scheme.

increase in the number of vehicles. In SHAVS (h=0), the sharp decrease can be explained by the fact that many-to-one communications result in higher contention with the increase in the number of vehicles; thus, data packets are discarded before they are received at the root aggregator because of significantly higher delays. In SHAVS (h=1), large number vehicles in the leaf regions contend in order to send data to the one of the two leaf aggregators. As a result, some data packets might not reach at the leaf aggregator due to delay violations, thereby leading to less packets received at the root aggregator. SHAVS (h=1), however, performs better than SHAVS (h=0) due to one more aggregation level. In contrast, we observe only a small and steady variation in performance of DHAVS even when the number of vehicles is high, thereby confirming the scalability of DHAVS. When the number of vehicles is 200, the average delivery efficiency of DHAVS is slightly higher than SHAVS (h=1). This is due to the fact that DHAVS (dynamically) decided the hierarchy as time progresses. Thus, when number of vehicles is 200, DHAVS uses a same hierarchy as SHAVS (h=1) sufficient to ensure efficient data collection at the root aggregator.

The average delivery efficiency of DB-VDG is close to SHAVS (h=1) and less than that of DHAVS. In case of DB-VDG, no aggregators are chosen and every vehicles is allowed to perform aggregation on data received from other vehicles. Since aggregation is performed independently by multiple vehicles, the number of data packets is not reduced effectively which results in lower value of average delivery efficiency. Moreover, DB-VDG experiences 15% decrease in performance as the number of vehicles increases from 200 to 400.

Fig. 14 shows the impact of number of vehicles on the maximun height of the aggregation hierrachy used by DHAVS for three different durations of data collection interval: 5 s, 10s and 15s. We observe an increase in the maximum height with increase in number of vehicles. The rationale is that the greater the height, the higher the reduction in number of data packets. Moreover, for any number of vehicles, the maximum height decreases with the increase in the duartion of data collection interval. This is because of the fact that with a higher duration the contention among vehicles has less impact on the

TABLE VI
TOTAL NUMBER OF BITS TRANSMITTED

Aggregation Scheme	Total Number of Bits Transmitted	
	Number of Vehicles=200	Number of Vehicles=400
DHAVS	$324x10^6$	$426x10^6$
SHAVS (h=0)	$640x10^6$	$144x10^7$
SHAVS (h=1)	$416x10^6$	$986x10^7$
DB-VDG	$408x10^6$	$650x10^6$

delivery efficiency. Hence a small height is sufficient to ensure a high value of delivery efficiency at the root aggregator.

Table. VI shows the total number of bits transmitted in the network during the entire simulation period for two different scenarios: one with number of vehicles equal to 200 and the other with number of vehicles equal to 400. We observe that the smallest number of bits is transmitted in case of DHAVS. This confiirms the ability of the proposed scheme to reduce bandwidth usage. As expected, the highest number of bits is transmitted in SHAVs (h=0). DB-VDG transmits 25% and 52% more bits than DHAVS for 200 and 400 vehicles respectively. This huge performance difference is attributed to the lack of a hierarchy in DB-VDG.

VI. CONCLUSION AND FUTURE WORKS

Vehicular sensing applications are becoming more popular as vehicles embed an increasing number of sophisticated sensors. Sensed data when collected over time from a big number of vehicles can generate enormous amounts of data which apart from congesting the network, may fail to arrive in real time to RSUs. In this paper, we address this problem by proposing a scheme for sensed data aggregation in a hierarchical way in order to decrease the volume of data transferred from the road and also to decrease the delay of delivering sensory information. The aggregation hierarchy is dynamically updated in order to ensure an improved performance.

In future, we would like to estimate the delivery efficiency considering network parameters such as channel quality in vehicular networks. Also, we will investigate probabilistic schemes to estimate the quality (e.g. reliability) of aggregated data. In addition, we would like to work on multimodal fusion approaches for urban sensing, where different sensory information such as images, traffic information, and air pollution level of a given area can be combined to provide a complete view of the urban environment.

REFERENCES

[1] U. Lee, E. Magistretti, M. Gerla, P. Bellavista, and A. Corradi, "Dissemination and harvesting of urban data using vehicular sensing platforms," *IEEE Trans. Veh. Technol.*, vol. 58, no. 2, pp. 882–901, Feb. 2009.

[2] K.-C. Lan, C.-M. Chou, and H.-Y. Wang, "Using vehicular sensor networks for mobile surveillance," in *Proc. IEEE VTC-Fall*, Sep. 2012, pp. 1–5.

[3] R. Bruno and M. Nurchis, "Robust and efficient data collection schemes for vehicular multimedia sensor networks," in *Proc. IEEE 14th Int. Symp. Workshops World Wireless, Mobile Multimedia Netw. (WoWMoM)*, Madrid, Spain, Jun. 2013, pp. 1–10.

[4] S. Mathur, S. Kaul, M. Gruteser, and W. Trappe, "ParkNet: A mobile sensor network for harvesting real time vehicular parking information," in *Proc. ACM MobiHoc*, May 2009, pp. 25–28.

[5] S.-C. Hu, Y.-C. Wang, C.-Y. Huang, and Y.-C. Tseng, "Measuring air quality in city areas by vehicular wireless sensor networks," *J. Syst. Softw.*, vol. 84, no. 11, pp. 2005–2012, Nov. 2011.

[6] D. Wu *et al.*, "ADDSEN: Adaptive data processing and dissemination for drone swarms in urban sensing," *IEEE Trans. Comput.*, vol. 66, no. 2, pp. 183–198, Feb. 2017.

[7] D. Wu, Q. Liu, Y. Li, J. A. McCann, A. C. Regan, and N. Venkatasubramanian, "Adaptive lookup of open WiFi using crowdsensing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 6, pp. 3634–3647, Dec. 2016.

[8] D. Wu, D. I. Arkhipov, Y. Zhang, C. H. Liu, and A. C. Regan, "Online war-driving by compressive sensing," *IEEE Trans. Mobile Comput.*, vol. 14, no. 11, pp. 2349–2362, Nov. 2015.

[9] P. Gomes, C. Olaverri-Monreal, and M. Ferreira, "Making vehicles transparent through V2V video streaming," *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 2, pp. 930–938, Jun. 2012.

[10] E. Belyaev, P. Molchanov, A. Vinel, and Y. Koucheryavy, "The use of automotive radars in video-based overtaking assistance applications," *IEEE Trans. Intell. Transp. Syst.*, vol. 14, no. 3, pp. 1035–1042, Sep. 2013.

[11] C. Stiller and J. Ziegler, "3D perception and planning for self-driving and cooperative automobiles," in *Proc. 9th Int. Multi-Conf. Syst., Signals Devices (SSD)*, Mar. 2012, pp. 1–7.

[12] W. van der Mark and D. M. Gavrila, "Real-time dense stereo for intelligent vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 7, no. 1, pp. 38–50, Mar. 2006.

[13] D. Zekri, B. Defude, and T. Delot, "Building, sharing and exploiting spatio-temporal aggregates in vehicular networks," *Mobile Inf. Syst.*, vol. 10, no. 3, pp. 259–285, 2014.

[14] C. Feng, Z. Li, S. Jiang, and R. Zhang, "Data aggregation and routing guidance with QoS guarantee in VANETs," *Int. J. Distrib. Sensor Netw.*, vol. 10, no. 7, pp. 1–11, Jul. 2014.

[15] Y. Zhu, Q. Zhao, and Q. Zhang, "Delay-constrained data aggregation in VANETs," *IEEE Trans. Veh. Technol.*, vol. 64, no. 5, pp. 2097–2107, May 2015.

[16] J. Jiru, L. Bremer, and K. Graffi, "Data aggregation in VANETs a generalized framework for channel load adaptive schemes," in *Proc. 39th Annu. IEEE Conf. Local Comput. Netw.*, Sep. 2014, pp. 394–397.

[17] M. Milojevic and V. Rakocevic, "Location aware data aggregation for efficient message dissemination in vehicular ad hoc networks," *IEEE Trans. Veh. Technol.*, vol. 64, no. 12, pp. 5575–5583, Dec. 2015.

[18] A. S. K. Mammu, J. Jiru, and U. H. Jayo, "Cluster based semantic data aggregation in VANETs," in *Proc. IEEE 29th Int. Conf. Adv. Inf. Netw. Appl. (AINA)*, Mar. 2015, pp. 747–753.

[19] S. Dietzel, J. Petit, F. Kargl, and B. Scheuermann, "In-network aggregation for vehicular ad hoc networks," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 4, pp. 1909–1932, 4th Quart., 2014.

[20] S. Ilarri, T. Delot, and R. Trillo-Lado, "A data management perspective on vehicular networks," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 4, pp. 2420–2460, 4th Quart., 2015.

[21] H. Samet, *Foundations of Multidimensional and Metric Data Structures*. San Francisco, CA, USA: Morgan Kaufmann, 2006.

[22] *Information Technology—Digital Compression and Coding of Continuous-Tone Still Images—Requirements and Guidelines*, document ISO/IEC 10918-1, 1994.

[23] *Information Technology—Generic Coding of Moving Pictures and Associated Audio Information—Part 3: Audio*, document ISO/IEC 13818-3, 1998.

[24] Y. Dieudonné, B. Ducourthial, and S. M. Senouci, "COL: A data collection protocol for VANET," in *Proc. Intell. Vehicles Symp. (IV)*, Jun. 2012, pp. 711–716.

[25] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu, "The broadcast storm problem in a mobile ad hoc network," in *Proc. ACM/IEEE MobiCom*, Aug. 1999, pp. 151–162.

[26] M. D. Dikaiakos, A. Florides, T. Nadeem, and L. Iftode, "Location-aware services over vehicular ad-hoc networks using car-to-car communication," *IEEE J. Sel. Areas Commun.*, vol. 25, no. 8, pp. 1590–1602, Oct. 2007.

[27] S. Peirce and R. Mauri, "Vehicle-infrastructure integration (VII) initiative benefit-cost analysis: Pre-testing estimates," U.S. DoT Draft Rep., Washington, DC, USA, Mar. 2007.

[28] D. Wu, Y. Zhang, L. Bao, and A. C. Regan, "Location-based crowdsourcing for vehicular communication in hybrid networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 14, no. 2, pp. 837–846, Jun. 2013.

[29] I. Catling and F. O. de Beek, "SOCRATES: System of cellular radio for traffic efficiency and safety," in *Proc. IEEE Vehicle Navigat. Inf. Syst. Conf.*, vol. 2. Oct. 1991, pp. 147–150.

[30] C. Sommer, A. Schmidt, Y. Chen, R. German, W. Koch, and F. Dressler, "On the feasibility of UMTS-based traffic information systems," *Ad Hoc Netw.*, vol. 8, no. 5, pp. 506–517, Jul. 2010.

[31] B. Hull *et al.*, "CarTel: A distributed mobile sensor computing system," in *Proc. ACM SenSys*, Nov. 2006, pp. 125–138.

[32] J. Rybicki, B. Scheuermann, M. Koegel, and M. Mauve, "PeerTIS: A peer-to-peer traffic information system," in *Proc. 6th ACM Int. Workshop Veh. Internetwork.*, Sep. 2009, pp. 23–32.

[33] J. Rybicki, B. Pesch, M. Mauve, and B. Scheuermann, "Supporting cooperative traffic information systems through street-graph-based peer-to-peer networks," in *Proc. 17th GI/ITG Conf. Commun. Distrib. Syst. (KiVS)*, Mar. 2011, pp. 121–132.

[34] I. Salhi, M. O. Cherif, and S. M. Senouci, "A new architecture for data collection in vehicular networks," in *Proc. IEEE ICC*, Jun. 2009, pp. 1–6.

[35] M. H. Arbabi and M. Weigle, "Using vehicular networks to collect common traffic data," in *Proc. ACM VANET*, Sep. 2009, pp. 117–118.

[36] A. Miloslavov and M. Veeraraghavan, "Sensor data fusion algorithms for vehicular cyber-physical systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 9, pp. 1762–1774, Sep. 2012.

[37] L. Wischoff, A. Ebner, H. Rohling, M. Lott, and R. Halfmann, "SOTIS—A self-organizing traffic information system," in *Proc. 57th IEEE Semiannu. Veh. Technol. Conf. (VTC-Spring)*, vol. 4. Apr. 2003, pp. 2442–2446.

[38] T. Nadeem, S. Dashtinezhad, C. Liao, and L. Iftode, "TrafficView: Traffic data dissemination using car-to-car communication," *ACM SIGMOBILE Mobile Comput. Commun. Rev.*, vol. 8, no. 3, pp. 6–19, Jul. 2004.

[39] N. Wisitpongphan, O. K. Tonguz, J. S. Parikh, P. Mudalige, F. Bai, and V. Sadekar, "Broadcast storm mitigation techniques in vehicular ad hoc networks," *IEEE Wireless Commun.*, vol. 14, no. 6, pp. 84–94, Dec. 2007.

[40] R. Chen, W.-L. Jin, and A. Regan, "Broadcasting safety information in vehicular networks: Issues and approaches," *IEEE Netw.*, vol. 24, no. 1, pp. 20–25, Jan./Feb. 2010.

[41] W. Zhu, D. Gao, C. H. Foh, W. Zhao, and H. Zhang, "A collision avoidance mechanism for emergency message broadcast in urban VANET," in *Proc. IEEE 83rd Veh. Technol. Conf. (VTC Spring)*, May 2016, pp. 1–5.

[42] W. Benrhaiem, A. S. Hafid, and P. K. Sahu, "Multi-hop reliability for broadcast-based VANET in city environments," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2016, pp. 1–6.

[43] R. Kumar and M. Dave, "A framework for handling local broadcast storm using probabilistic data aggregation in VANET," *Wireless Pers. Commun.*, vol. 72, no. 1, pp. 315–341, Sep. 2013.

[44] T. Saeed, M. Lestas, Y. Mylonas, A. Pitsillides, and V. Papadopoulou, "Analysis of probabilistic flooding in VANETs for optimal rebroadcast probabilities," in *Proc. IEEE/IFIP Netw. Oper. Manage. Symp. (NOMS)*, Apr. 2016, pp. 1181–1186.

[45] U. Lee, J. Lee, J.-S. Park, and M. Gerla, "FleaNet: A virtual market place on vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 59, no. 1, pp. 344–355, Jan. 2010.

[46] J. Sahoo, S. Cherkaoui, and A. Hafid, "Hierarchical aggregation for delay-sensitive vehicular sensing," in *Proc. Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, Aug. 2015, pp. 1365–1370.

[47] C. E. Palazzi, F. Pezzoni, and P. M. Ruiz, "Delay-bounded data gathering in urban vehicular sensor networks," *Pervas. Mobile Comput.*, vol. 8, no. 2, pp. 180–193, Apr. 2012.

[48] D. Delling, A. V. Goldberg, I. Razenshteyn, and R. F. Werneck, "Graph partitioning with natural cuts," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. (IPDPS)*, May 2011, pp. 1135–1146.

[49] J. Sahoo, E. H.-K. Wu, P. K. Sahu, and M. Gerla, "Congestion-controlled-coordinator-based MAC for safety-critical message transmission in VANETs," *IEEE Trans. Intell. Transp. Syst.*, vol. 14, no. 3, pp. 1423–1437, Sep. 2013.

[50] I. Pohl, "Bidirectional search," in *Machine Intelligence*, vol. 6, B. Meltzer and D. Michie, Eds. Edinburgh, Scotland: Edinburgh Univ. Press, 1971, pp. 127–140.

[51] Y.-L. Chou, *Statistical Analysis*, 2nd ed. New York, NY, USA: Holt, Rinehart and Winston, Jul. 1975.

[52] P. K. Sahu, E. H.-K. Wu, J. Sahoo, and M. Gerla, "BAHG: Backbone-assisted hop greedy routing for VANET's city environments," *IEEE Trans. Intell. Transp. Syst.*, vol. 14, no. 1, pp. 199–213, Mar. 2013.

[53] C. Lochert, M. Mauve, H. Füßler, and H. Hartenstein, "Geographic routing in city scenarios," *ACM SIGMOBILE Mobile Comput. Commun. Rev.*, vol. 9, no. 1, pp. 69–72, Jan. 2005.

[54] M. Jerbi, S. M. Senouci, T. Rasheed, and Y. Ghamri-Doudane, "Towards efficient geographic routing in urban vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 58, no. 9, pp. 5048–5059, Nov. 2009.

[55] *IEEE Standard for Information Technology—Local and Metropolitan Area Networks—Specific requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 6: Wireless Access in Vehicular Environments*, IEEE Standard 802.11p, Jul. 2010.

[56] J. C. Culberson and F. Luo, "Exploring the k-colorable landscape with iterated greedy," in *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge* (DIMACS Series in Discrete Mathematics and Theoretical Computer Science), vol. 26, D. S. Johnson and M. A. Trick, Eds. Providence, RI, USA: AMS, 1996, pp. 245–284. [Online]. Available: http://web.cs.ualberta.ca/~/

[57] R. K. Schmidt, B. Kloiber, F. Schüttler, and T. Strang, "Degradation of communication range in VANETs caused by interference 2.0—Real-world experiment," in *Communication Technologies for Vehicles* (Lecture Notes in Computer Science), vol. 6596. Berlin, Germany: Springer, Mar. 2011, pp. 176–188.

[58] *The Network Simulator—ns-2*, accessed Jan. 28, 2017. [Online]. Available: http://www.isi.edu/

[59] *User Manual for IMPORTANT Mobility Tool Generator in ns-2 Simulator*, accessed Jan. 28, 2017. [Online]. Available: https://www.semanticscholar.org/paper/User-Manual-for-Important-Mobility-Tool-Generators-Bai-Sadagopan/f91de61e5b095036ee8b16b458b944370bdf337a/pdf

[60] F. Bai, N. Sadagopan, and A. Helmy, "IMPORTANT: A framework to systematically analyze the impact of mobility on performance of routing protocols for adhoc networks," in *Proc. 22nd Annu. Joint Conf. IEEE Comput. Commun. (INFOCOM)*, vol. 2. Mar./Apr. 2003, pp. 825–835.

[61] L. Cheng, B. E. Henty, D. D. Stancil, F. Bai, and P. Mudalige, "Mobile vehicle-to-vehicle narrow-band channel measurement and characterization of the 5.9 GHz dedicated short range communication (DSRC) frequency band," *IEEE J. Sel. Areas Commun.*, vol. 25, no. 8, pp. 1501–1516, Oct. 2007.

Jagruti Sahoo (M'14) received the Ph.D. degree in computer science and information engineering from National Central University, Taiwan, in 2013. She was a Post-Doctoral Fellow with University of Sherbrooke, Canada, and with the Telecommunication Service Engineering Research Laboratory, CIISE, Concordia University, Canada. She is currently an Assistant Professor of Computer Science with South Carolina State University, South Carolina, USA. Her research interests include wireless sensor networks, vehicular networks, content delivery networks, cloud computing, and network functions virtualization. She served as a member of the Technical Program Committee in many conferences and as a Reviewer for many journals and conferences.

Soumaya Cherkaoui (M'99–SM'15) was with the industry as a Project Leader on projects targeted at the aerospace industry. She was an Invited Professor with University of Toronto, Monahh University, Bell Laboratories, University of California at Berkeley, and University of Montreal. She has over 100 publications in reputable journals, conferences, and workshops in the area of communication networks. She is currently a Full Professor with the Department of Electrical and Computer Engineering, University of Sherbrooke, Sherbrooke, QC, Canada. Since 2005, she has also been an Adjunct Full Professor with Lulea University, Lulea, Sweden, and has been the Director of INTERLAB, a research laboratory that conducts research funded both by government and industry. She has served as a General Chair, an Editor, a Member of Technical Committee, a Session Chair, or a Program Committee Member of many conferences or referenced journals. She is a Professional Engineer of Quebec, Canada.

Abdelhakim Hafid was an Assistant Professor with University of Western Ontario (UWO), London, ON, Canada; as a Research Director with the Advance Communication Engineering Center (venture established by UWO, Bell Canada, and Bay Networks), Canada; as a Researcher with CRIM, Canada; as a Visiting Scientist with GMD-Fokus, Berlin, Germany; and as a Visiting Professor with University of Evry, Evry, France. He spent several years as a Senior Research Scientist with Telcordia Technologies (formerly Bell Communications Research), NJ, USA, where he was involved in major research projects on the management of next-generation networks, including wireless and optical networks. He is currently a Full Professor with University of Montreal, Montreal, QC, Canada, where he founded the Network Research Laboratory in 2005. He is also a Research Fellow with the Interuniversity Research Center on Enterprise Networks, Logistics, and Transportation. He has extensive academic and industrial research experience in the area of the management of next-generation networks, including wireless and optical networks, QoS management, distributed multimedia systems, and communication protocols.

Pratap Kumar Sahu (M'14) received the Ph.D. degree from National Central University, Taiwan. He was a Post-Doctoral Researcher with the University of Montreal, Canada, and National Central University, Taiwan. He is currently a Research Fellow with the CONNECT Centre, Trinity College Dublin, Dublin. His research interests include network coding, vehicular ad hoc networks, and sensor networks. Mostly, he focuses on various aspects of intelligent transportation systems. He has many publications in his credit, such as journals like IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, IEEE TRANSACTIONS ON CLOUD COMPUTING, IEEE SENSORS, *Elsevier Vehicular Communications*, and *Springer Telecommunication Systems*. He also has many publications in various conferences, including IEEE GLOBECOM, IEEE ICC, IEEE LCN, and IEEE ICCCN. He served as Technical Program Committee Member for many IEEE conferences. He currently serves as an Associate Editor of *Journal of Circuits, Systems, and Computers*.