Event Notification in VANET With Capacitated Roadside Units

Joy Chandra Mukherjee, Arobinda Gupta, and Ravella Chaitanya Sreenivas

***Abstract*—Many future vehicular ad hoc network (VANET) applications will be event driven and will require events of different types to be delivered to moving vehicles within some specified time. In this paper, we propose a publish–subscribe based event notification framework that uses roadside units (RSUs) to deliver events to vehicles that subscribe to them within the validity periods of both the subscriptions and the events. Each RSU can disseminate only a finite number of events at a time and has a cost associated with it. Two scheduling problems to schedule the dissemination of events from RSUs are formulated. The first problem aims to maximize only the number of subscriptions that are matched to some events. The second problem, in addition to maximizing the number of subscriptions matched, also aims to minimize the total cost of disseminating the events. We have designed offline and online algorithms for the problems that a service provider can execute to schedule event disseminations from the RSUs. Detailed simulation results are presented to show that the algorithms are able to match a high percentage of subscriptions with low average event dissemination cost for some realistic city traffic scenarios.**

***Index Terms*—Event dissemination, scheduling, publish-subscribe, VANET.**

I. Introduction

A vehicular ad-hoc network (VANET) is a type of mobile ad-hoc network where nodes are moving vehicles and communication can be vehicle-to-vehicle (V2V), or vehicle-to-infrastructure (V2I). The infrastructure is usually deployed on the roadside, also referred to as road-side units (RSUs). The RSUs can be used as broadcasting nodes and are usually connected to the internet through some backbone network. VANETs have become an important area of research with potential applications in various domains such as safety, navigational applications, in-vehicle infotainment etc. [2].

Many future VANET applications will be event-driven and will require events of different types to be delivered to vehicles. For example, a vehicle driver may be interested to know about traffic conditions in his/her route, available parking spots close to the destination, fuel prices posted by gas stations etc. These events are useful to the driver only when they are delivered to the vehicle within some specified time without having the vehicle to deviate from its route. Such event delivery *in time en route* to moving vehicles in VANET is an interesting problem.

Manuscript received February 24, 2015; revised August 15, 2015 and November 9, 2015; accepted December 14, 2015. The Associate Editor for this paper was X. Cheng.

J. C. Mukherjee and A. Gupta are with the Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, Kharagpur-721 302, India (e-mail: joy.cs@cse.iitkgp.ernet.in; agupta@cse.iitkgp.ernet.in).

R. C. Sreenivas is with Microsoft Development Center Serbia, 11070 Belgrade, Serbia (e-mail: rcsreenivas@gmail.com).

Digital Object Identifier 10.1109/TITS.2015.2511145

Events can be delivered to vehicles by using V2V or V2I communication, or by a combination of both. Several works [7], [14], [16] use V2V communication in different ways for event dissemination. However, using only V2V communication for event dissemination causes redundant message delivery, and usually cannot tolerate network partitioning. Several works [3], [9], [10] have explored the problem of delivering the events through RSUs. However, none of these approaches consider the validity periods of events or subscriptions, or the cost of usage of the RSUs. Mukherjee *et al.* [11] propose a publish-subscribe framework based scheme for low cost dissemination of events that consider validity periods of subscriptions and events, and the cost of RSUs. However, their work assumes that a RSU can broadcast an infinite number of events at one time, which is not practical due to limited bandwidth of the RSUs. The assumption of infinite capacity also makes the problem much easier to solve, as this guarantees that all subscriptions can always be matched as long as a subscription has at least one matching event and the vehicle that raises the subscription passes by at least one RSU.

In this paper, we consider a more realistic model where each RSU has finite capacity, i.e., only a finite number of events can be broadcast from a RSU at the same time. Under this model, we propose a publish-subscribe based framework for event dissemination in which vehicles subscribe to a single service provider (SP) for specific types of events through RSUs. The events are also reported to the SP. A vehicle can receive an event if the event is broadcast from a RSU in its route when the vehicle passes by it. The SP schedules the events at RSUs for appropriate time intervals such that the vehicles can receive the events within the validity periods of both the subscriptions and the events. Given the finite capacity of RSUs, it may not be possible to match all subscriptions. Thus it is important for the SP to properly schedule the events to maximize the number of subscriptions matched. At the same time, since each RSU also has a per-unit time cost for event dissemination, the SP should also aim to minimize the total cost of disseminating the events. We formulate two scheduling problems: the *Bounded Maximum Subscription Matching* (*B-MaxSubMatch*) problem aims to maximize only the number of subscriptions matched, while the *Bounded Minimum Cost Maximum Subscription Matching* (*B-MinCostMaxSubMatch*) problem also aims to minimize the total cost of event dissemination while maximizing the number of subscriptions matched.

The main contributions of the paper are as follows.

1) We have formulated the *B-MaxSubMatch* and the *B-MinCostMaxSubMatch* problems for event dissemination through RSUs with finite capacity. To the best of our

knowledge, this is the first work that considers validity period of events and subscriptions, cost of usage of RSUs, and the capacity of the RSUs.

2) The *B-MinCostMaxSubMatch* problem is proved to be NP-complete.
3) An offline algorithm is proposed for the *B-MaxSubMatch* problem. Simulation results in different city traffic scenarios show that the algorithm matches almost 100% of the subscriptions in most scenarios.
4) An offline algorithm is proposed for the *B-MinCostMaxSubMatch* problem. Simulation results show that the algorithm outperforms the offline algorithm for *B-MaxSubMatch* in terms of cost for event dissemination without compromising too much on the percentage of subscriptions matched.
5) An online algorithm is proposed for the *B-MinCostMaxSubMatch* problem that considers the arrival of subscriptions in real-time. Simulation results show that this delaying helps in reducing event dissemination cost for most of the scenarios.

The rest of this paper is organized as follows. Section II discusses some related works. Section III states the *B-MaxSubMatch* and the *B-MinCostMaxSubMatch* problems formally. Section IV presents two offline algorithms for the two problems. Section V describes an online algorithm for the problem. Section VI presents detailed simulation results for some realistic traffic scenarios. Finally, Section VII concludes the paper and discusses some scope for future work.

II. Related Works

Events can be delivered to vehicles by using V2V or V2I communication, or by a combination of both. The work in [15] uses optimized flooding based V2V communication. Similarly, [7] uses V2V communication for a targeted broadcast. Several other works [7], [14], [16] use V2V communication in different ways for event dissemination. For a large number of vehicles, only V2V communication is not enough since it causes redundant message delivery, and usually cannot tolerate network partitioning. Also, a large number of intermediate vehicles that route the messages may not be interested in these messages. Therefore, using the RSUs for event placement and delivery has been explored. The works in [9], [10] address the issue of delivering the events from a moving vehicle to a fixed static destination through RSUs. However, their solutions are not applicable to cases where the destinations are mobile like vehicles in VANET. Chen *et al.* [3] address a problem in which events must be delivered to mobile vehicles en route through wireless base stations on the way, as they travel toward their destinations. Shen *et al.* [13] address the challenge of how best to assign transmission opportunity to nodes with maximum dissemination utility while avoiding the collision problem and minimizing the dissemination delay. However, they do not consider the case when the subscription and the event have validity periods, and the cost of usage of the RSUs. The impact of the vehicular traffic density on the level crossing rate and average fade duration for nonisotropic scattering V2V Ricean fading channels is investigated in [5]. The impact of information and communication systems on transportation infrastructure is described in [4]. Cheng *et al.* [6] investigate the joint coordination of V2V and V2I communications, and carry out a feasibility study of device-to-device (D2D) communication for intelligent transportation systems based on both the features of D2D and the nature of vehicular networks. However, their approach does not consider the case where the events have validity periods associated with them. They also do not consider the cost of usage of the RSUs. Mukherjee *et al.* [12] consider validity periods of subscriptions and events, and use publish-subscribe communication framework for disseminating events. However, their work assumes that a RSU can place and disseminate infinite number of events at a time which is not practical.

III. Problem Formulation

Let E denote the set of events reported to the service provider (SP), and F denote the set of subscriptions issued from vehicles. Let type_f, T_f, and Δ_f denote the type, the issue time, and the validity period of a subscription $f \in F$ respectively. Similarly, type_e, T_e, and Δ_e denote the type, the issue time, and the validity period of an event $e \in E$ respectively. A subscription f matches an event e if $\text{type}_f = \text{type}_e$, and there is an overlap between the validity interval $[T_f, T_f + \Delta_f]$ of the subscription f and the validity interval $[T_e, T_e + \Delta_e]$ of the event e. For simplicity, it is assumed that every subscription has exactly one matching event.

Consider a SP with a set U of m RSUs under its control. For each RSU $u \in U$, let cap_u denote the maximum number of events that can be disseminated from u at a time, and c_u denote the event dissemination cost from u per unit time; thus the cost for disseminating an event e from u is given by c_u times the time for which e is disseminated from u. A vehicle can receive an event notification only when it is in the broadcast range of a RSU broadcasting the event. Due to capacity constraints, all events cannot be disseminated simultaneously from a RSU that potentially restrains matching of all subscriptions. As a result, the dissemination of an event from a RSU is not independent of the dissemination of other events.

A subscription is assumed to carry the route information of the vehicle that raises it. On receiving the subscription, the SP can determine the sequence of RSUs $\langle u_1, \ldots, u_k \rangle$ that the vehicle will pass by on its way to its destination. Also, given locations of the RSUs and a map, assuming an average speed of the vehicle, the SP can estimate the time interval during which the vehicle will pass by the RSUs in its route. Since a subscription f is raised from a vehicle, its spatio-temporal validity for different RSUs in its route can thus be expressed as $\langle (u_1, t_{u_1 f}, d_{u_1 f}), \ldots, (u_k, t_{u_k f}, d_{u_k f}) \rangle$, where $[t_{u_i f}, d_{u_i f}]$ denotes the tentative time interval of the vehicle passing by the RSU u_i. However, a RSU u_i in the route of a vehicle that raises the subscription f can only be qualified as a potential candidate for disseminating the matching event e if $T_f \leq t_{u_i f}, d_{u_i f} \leq T_f + \Delta_f$, and $T_e \leq t_{u_i f}, d_{u_i f} \leq T_e + \Delta_e$. Therefore, the route can be expressed as $\langle (u_1, t_{u_1 f}, d_{u_1 f}), \ldots, (u_r, t_{u_r f}, d_{u_r f}) \rangle$ containing only the RSUs that satisfy the above condition. We will refer to this as the *route of the subscription* in the rest of this paper. It is assumed that every subscription has at least one

TABLE I
VARIABLES IN PROBLEM FORMULATION

Variables	Meaning
U	Set of RSUs
m	Number of RSUs
cap_u	Capacity of u i.e. maximum number of events that can be disseminated at a time from u
c_u	Event dissemination cost from u
E	Set of events
F	Set of subscriptions
$type_{f/e}$	Type of subscription f / Type of event e
$T_{f/e}$	Issue time of f / Issue time of e
$\Delta_{f/e}$	Validity period of f / Validity period of e
t_{uf}	Time at which f enters the broadcast range of u
d_{uf}	Time at which f leaves the broadcast range of u
P	Total event dissemination cost of the SP
N	Total number of subscriptions matched by the SP
T	Total travel time of all vehicles
y_{ueft}	e can be disseminated from u at time t to match f
x_{ueft}	Whether e will be disseminated from u at time t to match f

RSU in its route. Since the time interval $[t_{u_if}, d_{u_if}]$ can only be estimated, we assume that the RSU u_i, if selected to broadcast an event that matches f, must broadcast it for the entire duration $[t_{u_if}, d_{u_if}]$. Note that a single broadcast of an event from a RSU can match more than one subscription if the vehicles pass by the RSU at the same time.

The total travel time of all the vehicles is divided into T discrete timeslots, i.e., any vehicle will start or finish its journey at a timeslot between 1 and T. Let (0, 1] denote the 1st timeslot, ..., and $(T-1, T]$ denote the Tth timeslot. We define a set of input variables $\{y_{\text{ueft}}\}$, where $y_{\text{ueft}} = 1$ indicates that the event $e \in E$ can be disseminated from the RSU $u \in U$ at the tth timeslot to match the subscription $f \in F$, else it is 0. The output is a set of variables $\{x_{\text{ueft}}\}$, where $x_{\text{ueft}} = 1$ indicates that the event $e \in E$ will be disseminated from the RSU $u \in U$ at the tth timeslot to match the subscription $f \in F$, else it is 0. To match f by the event e, it is required that $x_{ueft} = 1$ for at least one such $u \in U$, $\forall\, t \in [t_{uf}, d_{uf}]$. Table I summarizes the variables used in the problem formulation.

We define the following two problems based on the objectives to be achieved.

B-MaxSubMatch: The objective of this problem is to schedule the dissemination of events from the RSUs so as to maximize the number N of subscriptions matched. This is formally specified as follows:

$$\text{maximize } N = \sum_{u\in U}\sum_{e\in E}\sum_{f\in F}\bigwedge_{t=t_{uf}}^{d_{uf}}(y_{\text{ueft}} \times x_{\text{ueft}}) \tag{1}$$

subject to the conditions

$$\forall\, u \in U\ \forall e \in E\ \forall f \in F\ \forall t \in [0,T]\ y_{\text{ueft}}, x_{\text{ueft}} \in \{0,1\} \tag{2}$$

$$\forall\, u \in U\ \forall e \in E\ \forall f \in F\ \forall t \in [0, t_{uf})\ \ y_{\text{ueft}} = x_{\text{ueft}} = 0 \tag{3}$$

$$\forall\, u \in U\ \forall e \in E\ \forall f \in F\ \forall t \in (d_{uf}, T]\ y_{\text{ueft}} = x_{\text{ueft}} = 0 \tag{4}$$

$$\forall\, t \in [0,T]\ \forall\, u \in U \sum_{e\in E}\bigvee_{f\in F}(y_{\text{ueft}} \times x_{\text{ueft}}) \le \text{cap}_u \tag{5}$$

$$\forall\, f \in F\ \forall\, u \in U\ \forall\, e \in E$$

$$\sum_{t=0}^{T}\left((y_{\text{ueft}} \times x_{\text{ueft}}) \bigwedge_{t=t_{uf}}^{d_{uf}}(y_{\text{ueft}} \times x_{\text{ueft}})\right)$$

$$= (d_{uf} - t_{uf}) \bigwedge_{t=t_{uf}}^{d_{uf}}(y_{\text{ueft}} \times x_{\text{ueft}}). \tag{6}$$

Constraints 3 and 4 jointly specify that for a given event, a subscription, and a RSU, specific time duration is used for matching the subscription. Constraint 5 specifies that a RSU can broadcast a maximum number of events at a given point in time. Constraint 6 indicates that for a subscription f, the broadcast of a matching event from a RSU u continues for the entire duration $[t_{uf}, d_{uf}]$ for which the vehicle is in the range of broadcast of the RSU u.

B-MinCostMaxSubMatch: The objective of this problem is to schedule the dissemination of events from the RSUs so as to maximize the number N of subscriptions matched while minimizing the dissemination cost P from RSUs. This is formally specified as follows:

$$\text{maximize } N = \sum_{u\in U}\sum_{e\in E}\sum_{f\in F}\bigwedge_{t=t_{uf}}^{d_{uf}}(y_{\text{ueft}} \times x_{\text{ueft}}) \tag{7}$$

$$\text{minimize } P = \sum_{t=0}^{T}\sum_{u\in U}\sum_{e\in E} c_u \bigvee_{f\in F}(y_{\text{ueft}} \times x_{\text{ueft}}) \tag{8}$$

subject to the same conditions (2)–(6).

In Equations (1) and (7), $\bigwedge$ denotes the *logical and* symbol that indicates that to match a subscription f by broadcasting a matching event e from a RSU u, it is required that x_{ueft} be true for the entire time interval $t \in [t_{uf}, d_{uf}]$. In Equation (8), $\bigvee$ denotes the *logical or* symbol that indicates that if an event e is broadcast from a RSU u at time t to match more than one subscription of the same type, then the event dissemination cost c_u is considered only once for e (and not again and again for all the matching subscriptions) since a single broadcast of e can match all of them.

The *B-MinCostMaxSubMatch* problem is proved to be NP-complete (Proof is given in Appendix A).

IV. OFFLINE ALGORITHMS

In the offline setting, we have assumed that the sets of events and subscriptions are known to the SP in advance. In this section, we propose two offline algorithms, *B-MaxSubMatch-Offline* for the *B-MaxSubMatch* problem and *B-MinCostMaxSubMatch-Offline* for the *B-MinCostMaxSubMatch* problem. Before describing the algorithms, we first define the following:

Definition 1: A chunk k at a RSU u for an event e is a set of subscriptions $\{f_1, \ldots, f_n\}$ with time intervals $\{[t_{uf_1}, d_{uf_1}], \ldots, [t_{uf_n}, d_{uf_n}]\}$ such that (*a*) $\forall\, i \in [1, \ldots, n]$ $\text{type}_{f_i} = \text{type}_e$ (*b*) $\forall\, t \in [\tau^k_{ue}, \delta^k_{ue}] \exists i \in [1, \ldots, n],\ t_{uf_i} \le t \le d_{uf_i}$, where $\tau^k_{ue} = \text{minimum}_{\forall\, i\in[1,\ldots,n]} t_{uf_i}$ and $\delta^k_{ue} = \text{maximum}_{\forall\, i\in[1,\ldots,n]} d_{uf_i}$. The number of subscriptions n is called the size of the chunk. The time interval $[\delta^k_{ue} - \tau^k_{ue}]$ is called the chunk interval.

Note that a chunk at a RSU for a single event can only be formed with subscriptions of identical types. Also, two different chunks may overlap with each other. A *maximal chunk* is defined as a chunk that is not overlapping with any other chunk. To determine maximal chunks from a set of subscriptions at a RSU, the subscriptions are first sorted in non-decreasing order of t_{uf} values. Then all the subscriptions are scanned in the sorted order, and checked whether the current subscription overlaps with any of the earlier subscriptions or not. If it overlaps, it is included in the maximal chunk and the chunk interval is updated accordingly. Otherwise, a new maximal chunk is created with the current subscription.

To determine the set of chunks corresponding to a maximal chunk mcnk, the subscriptions in mcnk are sorted in non-decreasing order of t_{uf} values. For k number of subscriptions in mcnk, there are k number of t_{uf} and d_{uf} values. Therefore the maximum number of chunks is upper bounded by k^2. For each of the k^2 intervals, it is possible to find out the set of subscriptions that belongs to it, and to verify whether the set of subscriptions constitutes a chunk or not.

The algorithm for finding maximal chunks and the algorithm for finding chunks from a maximal chunk are given in Appendix B. The benefit of introducing chunks in solving the problems is as follows. Since a chunk at a RSU consists of a set of overlapping subscriptions of identical types, the broadcast of a matching event from the RSU for the chunk interval is sufficient to match all subscriptions in the chunk. The overlapped subscriptions in a chunk help to minimize the average time for event dissemination to match a subscription in the chunk, thereby minimizing the cost for event dissemination. Also the overlapping subscriptions ensure a more compact packing, and therefore, more events can be scheduled for dissemination from the RSUs resulting in matching of larger number of subscriptions.

A. B-MaxSubMatch-Offline Algorithm

Let F_u denote the set of subscriptions that passes by RSU u, and F_{ue} denote the subset of F_u that matches the event e. Clearly, $F_u = \cup_{e \in E} F_{ue}$. We define the average-subscription-space index (ASSI) of a RSU as follows:

Definition 2: The average-subscription-space index (ASSI) of a RSU u, denoted by ASSI_u is defined as $(\text{cap}_u \times (\text{maxd}_u - \text{mint}_u))/|F_u|$ where, $\text{mint}_u = \text{minimum } \{t_{uf} |\ f \in F_u\}$ and $\text{maxd}_u = \text{maximum } \{d_{uf} |\ f \in F_u\}$.

For example, for a RSU u with $\text{cap}_u = 4$, $F_{ue_1} = \{f_1, f_2, f_3\}$ with time intervals [10, 12], [5, 9], [7, 8] and $F_{ue_2} = \{f_4, f_5\}$ with time intervals [10, 13], [6, 8]. Clearly, $F_u = \{f_1, f_2, f_3, f_4, f_5\}$, $\text{maxd}_u = 13$ and $\text{mint}_u = 5$. Therefore, $\text{ASSI}_u = (4 \times (13 - 5))/5 = 6.4$. A higher ASSI value of a RSU indicates a higher scope for disseminating events from it.

The intuition behind the algorithm is to first select RSUs in the order of non-increasing ASSI values, thereby giving priority to the RSU that has higher space for placing events, followed by a more efficient and compact placement and dissemination of events by applying the *Weighted Interval Scheduling* algorithm (WIS) [8] over the maximal chunks that can be formed for different events. The input to WIS is a set of maximal chunks MCH_u at RSU u each having a chunk interval and a weight that is equal to the size of the chunk. The output of WIS is a subset $\text{MCH}'_u \in \text{MCH}_u$ that gives the maximum weight (i.e., maximum number of subscriptions matched) subset of non-overlapping maximal chunks. The WIS algorithm is executed on MCH_u for cap_u number of times to fully utilize the RSU.

The pseudo code for the algorithm is shown in Algorithm 1. The set of RSUs is sorted in non-increasing order of their ASSI values and stored in U' (Line 1). The algorithm runs for U' rounds (Lines 3–14). In each round, the RSU $u \in U'$ with the highest ASSI value is selected (Line 4). For the RSU u and for each event e, the set F_{ue} is determined from F, and the set of all maximal chunks is computed. MCH_u stores the set of maximal chunks for all events at u (Line 5). For each RSU u, the algorithm iterates for cap_u number of passes (Lines 6–12). In each pass, the WIS algorithm is invoked over MCH_u to determine a subset MCH'_u of maximal chunks (Line 7). All the subscriptions $F_{\text{MCH}'_u}$ in MCH'_u are deleted from F, MCH'_u is removed from MCH_u, and the number of subscriptions $|F_{\text{MCH}'_u}|$ is added to totalMatched (Lines 8–11). After cap_u number of passes, the RSU u is removed from U' (Line 13). The *B-MaxSubMatch-Offline* algorithm then passes to the next round until U' is empty. At the end of the algorithm, totalMatched gives the total number of subscriptions matched (Line 15).

Algorithm 1 B-MaxSubMatch-Offline Algorithm

1: $U' \leftarrow$ Sorted U in non-increasing order of ASSI values
2: totalMatched $\leftarrow 0$
3: **while** $U' \neq \phi$ **do**
4: $u \leftarrow$ RSU with the highest ASSI value in U'
5: $\text{MCH}_u \leftarrow \cup_{e \in E}$ Set of maximal chunks using F_{ue}
6: **for** $(i \leftarrow 1;\ i \leq \text{cap}_u;\ i \leftarrow i + 1)$ **do**
7: Invoke *WIS* algorithm over MCH_u to get MCH'_u
8: $F_{\text{MCH}'_u} \leftarrow$ Subscriptions in MCH'_u
9: totalMatched $\leftarrow$ totalMatched $+ |F_{\text{MCH}'_u}|$
10: $F \leftarrow F \setminus F_{\text{MCH}'_u}$
11: $\text{MCH}_u \leftarrow \text{MCH}_u \setminus \text{MCH}'_u$
12: **end for**
13: $U' \leftarrow U' \setminus \{u\}$
14: **end while**
15: **return** totalMatched

The time required to sort RSUs by their ASSI values is $O(|U| \log_2 |U|)$. The outer loop runs for $O(|U|)$ times. In each iteration, the time complexity of calculating the set of maximal chunks MCH_u is $O(|F| \log_2 |F|)$. Then, the WIS algorithm (time complexity $O(|\text{MCH}_u| \log_2 |\text{MCH}_u|) = O(|F| \log_2 |F|)$) is called for cap_u number of times. Let mcap be the maximum capacity that a RSU can have. Then, given a RSU u, the time complexity of the loop (Lines 4–13) is $O(\text{cap}_u \times |F| \log_2 |F|) = O(\text{mcap} \times |F| \log_2 |F|)$. Therefore, the time complexity of Algorithm 1 is $O(|U| \log_2 |U| + \text{mcap} \times |F| \log_2 |F|) = O(\text{mcap} \times |F| \log_2 |F|))$.

The following example illustrates the working of the algorithm. Consider a scenario with five RSUs u_1, u_2, u_3, u_4, u_5 in a city with $c_{u_1} = 5, c_{u_2} = 6, c_{u_3} = 7, c_{u_4} = 6, c_{u_5} = 5$, and $\text{cap}_{u_1} = \text{cap}_{u_2} = \text{cap}_{u_3} = \text{cap}_{u_4} = \text{cap}_{u_5} = 1$. Also consider two events e_1, e_2 with validity periods of [1, 30], and four subscriptions

f_1, f_2, f_3, and f_4 with validity periods of [1, 25], where $\text{type}_{f_1} = \text{type}_{f_2} = \text{type}_{f_3} = \text{type}_{e_1}$ and $\text{type}_{f_4} = \text{type}_{e_2}$. f_1 passes by u_1 and u_2 between [9, 11] and [12, 14] respectively. f_2 passes by u_3, u_4, and u_5 between [9, 12], [10, 12], and [14, 16] respectively. f_3 and f_4 pass by u_4 and u_2 between [10, 12] and [12, 14] respectively. Thus the *ASSI* for u_1, u_2, u_3, u_4, u_5 are 2, 0.66, 3, 0.66, 2 respectively. The RSUs sorted in non-increasing order of their ASSI values are u_3, u_1, u_5, u_2, u_4. The algorithm selects u_3 with highest *ASSI*, and disseminates e_1 to match f_2 at [9, 12]. Next it selects u_1, and disseminates e_1 to match f_1 at [9, 11]. Then it selects u_5, but no event dissemination takes place. After that it selects u_2, and disseminates e_1 to match f_3 at [12, 14]. At the end it selects u_4, and disseminates e_2 to match f_4 at [10, 12]. Thus the total number of subscriptions matched is 4. However, the total cost incurred is $((12-9) \times 7 + (11 - 9) \times 5 + (14 - 12) \times 6 + (12 - 10) \times 6) = 55$.

B. B-MinCostMaxSubMatch-Offline Algorithm

The *B-MaxSubMatch-Offline* algorithm attempts to maximize only the number of subscriptions. In this section, we propose the *B-MinCostMaxSubMatch-Offline* algorithm for the *B-MinCostMaxSubMatch* problem which also tries to minimize the total cost for event dissemination while maximizing the number of subscriptions matched. Before describing the algorithm, we define the following:

Definition 3: The *Overlap-Cost* (OC_{ue}^k) of a chunk k at RSU u for an event e is defined as the product of c_u and the chunk interval $[\delta_{ue}^k - \tau_{ue}^k]$. The *Average-Overlap-Cost* (AOC_{ue}^k) of the chunk k is OC_{ue}^k divided by the size of chunk k.

Each chunk is represented by a tuple $\langle k, e, u, \text{flist}, \text{CI}, \text{AOC}, s\rangle$, where flist is the set of subscriptions that creates the chunk k of size s and can be matched by disseminating the event e at RSU u for the chunk interval CI. The cost for event placement and dissemination is $\text{OC} = \text{AOC} \times s$.

The pseudo code for the *B-MinCostMaxSubMatch-Offline* algorithm is given in Algorithm 2. The set of all maximal chunks are computed and stored in a list MCH (Line 4). For each chunk in MCH, the set of all chunks are computed and stored in a list CH (Line 5). In each pass of the algorithm, the chunk temp with the minimum AOC is selected (Line 7). Then it is checked whether a matching event e can be placed and disseminated for the chunk interval in the RSU or not (Line 8). If e cannot be placed, then temp is deleted from CH (Line 17). If e can be placed for the entire chunk interval, then the chunk is added to the final solution (Line 9). The RSU capacity is decremented by one for the chunk interval (Line 11). The total cost is updated by adding the OC of the chunk (Line 12). The total number of subscriptions that are matched is updated by adding the size of the chunk (Line 13). Then all the chunks (including temp) that have at least one common subscription with temp are deleted from CH (Lines 14 and 15). The process moves into the next pass until CH is empty.

The time required to determine MCH from F is $O(|F| \log_2 |F|)$, and to calculate CH from MCH is $O(|F|^3)$. The while loop iterates for $|\text{CH}|(= O(|F|^2))$ number of times. Line 11 takes $O(\delta_{ue}^k - \tau_{ue}^k) = O(T)$ time ($T = \text{maxd} - \text{mint}$, where $\text{mint} = \text{minimum } \{t_{uf} |\ u \in U,\ f \in F_u\}$ and $\text{maxd} = \text{maximum } \{d_{uf} |\ u \in U,\ f \in F_u\}$). Lines 14 and 15 takes $O(|F|^2)$ time. Hence, to execute the while loop (Lines 6–19), it takes $O(|F|^2 \times (T + |F|^2))$ time. Therefore, the time complexity of Algorithm 2 is $O(|F|^3 + |F|^2 \times (T + |F|^2)) = O(|F|^2 \times (T + |F|^2))$.

Algorithm 2 B-MinCostMaxSubMatch-Offline Algorithm

1: totalCost $\leftarrow 0$
2: $\text{CH}_{\text{final}} \leftarrow \phi$
3: $x_{\text{ufet}} \leftarrow 0$ for all u, e, f, t
4: MCH $\leftarrow$ Set of maximal chunks using F for all RSUs and for all events
5: CH $\leftarrow$ Set of chunks from MCH
6: **while** (CH $\neq \phi$) **do**
7: temp $\leftarrow$ Chunk k in CH with minimum AOC that can be matched by disseminating e from u for $t \in [\tau_{ue}^k, \delta_{ue}^k]$
8: **if** (e can be placed in u for $[\tau_{ue}^k, \delta_{ue}^k]$ to match temp) **then**
9: $\text{CH}_{\text{final}} \leftarrow \text{CH}_{\text{final}}$.append(temp)
10: $x_{\text{ufet}} \leftarrow 1$, where $u = \text{temp}.u,\ \forall f \in \text{temp}.F_{ue}^k$, $e = \text{temp}.e$ and $\forall t \in [\tau_{ue}^k, \delta_{ue}^k]$
11: $\text{cap}_{ut} \leftarrow \text{cap}_{ut} - 1\ \forall t \in [\tau_{ue}^k, \delta_{ue}^k]$
12: totalCost $\leftarrow$ totalCost $+ oc_{ue}^k$
13: totalMatched $\leftarrow$ totalMatched $+ |F_{ue}^k|$
14: CH$'$ $\leftarrow$ temp $\cup$ chunks in CH having at least one common subscription with temp
15: CH $\leftarrow$ CH $\setminus$ CH$'$
16: **else**
17: CH $\leftarrow$ CH $\setminus$ temp
18: **end if**
19: **end while**
20: **return** totalCost, totalMatched, CH_{final}

Consider the same example mentioned in Section IV-A. The chunks are $\langle 1, e_1, u_4, \{f_2, f_3\}, [10, 12], 6, 2\rangle$, $\langle 2, e_1, u_2, \{f_1, f_3\}, [12, 14], 6, 2\rangle$, $\langle 3, e_1, u_1, \{f_1\}, [9, 11], 10, 1\rangle$, $\langle 4, e_1, u_5, \{f_2\}, [14, 16], 10, 1\rangle$, $\langle 5, e_2, u_2, \{f_4\}, [12, 14], 12, 1\rangle$, $\langle 6, e_2, u_4, \{f_4\}, [10, 12], 12, 1\rangle$, and $\langle 7, e_1, u_3, \{f_2\}, [9, 12], 21, 1\rangle$. The algorithm selects $\langle 1, e_1, u_4, \{f_2, f_3\}, [10, 12], 6, 2\rangle$ with lowest *AOC*, disseminates e_1 from u_4 to match f_2 and f_3 at [10, 12], and removes chunks 1, 2, 4, and 7. Then it selects $\langle 3, e_1, u_1, \{f_1\}, [9, 11], 10, 1\rangle$, disseminates e_1 from u_1 to match f_1 at [9, 11], and removes chunk 3. Next it selects $\langle 5, e_2, u_2, \{f_4\}, [12, 14], 12, 1\rangle$, disseminates e_2 from u_2 to match f_4 at [12, 14], and removes chunks 5 and 6. Thus the total number of subscriptions matched is 4. However, the total cost incurred is $(6 \times 2 + 10 \times 1 + 12 \times 1) = 34$.

V. Online Algorithm

In the online setting, the subscriptions and the events are reported to the SP only when they occur in real time. Given an event e, some of the subscriptions matched to e may come before or at T_e, and the rest may arrive in the interval $(T_e, T_e + \delta_e]$. The online algorithm tries to delay the dissemination of an event from a RSU scheduled by the *B-MinCostMaxSubMatch-Offline* algorithm till its actual time of broadcast from the RSU. This gives the online algorithm the scope to rerun the offline

algorithm with new subscriptions that come later, which may result in a better schedule.

Algorithm 3 B-MinCostMaxSubMatch-Online Algorithm

totalCost $\leftarrow 0$
for $(t \leftarrow 1;\ t \leq T;\ t \leftarrow t+1)$ **do**
  if an event e occurs at t **then**
    e is added to E
  end if
  if a subscription f arrives at t **then**
    f is added to F
  end if
  if new subscriptions arrived at t that have a matching event, or new events in E at t have some matching subscriptions **then**
    $\text{Cost}_{\text{Off}}, \text{Matched}_{\text{Off}}, \text{CH}_{\text{Off}} \leftarrow$ Run *B-MinCostMaxSubMatch-Offline* on F
  end if
  for all chunk k in CH_{Off} **do**
    if $t = \tau_{ue}^k$ **then**
      if e can be disseminated from u for $[\tau_{ue}^k, \delta_{ue}^k]$ **then**
        e is disseminated from RSU u for $[\tau_{ue}^k, \delta_{ue}^k]$
        $F \leftarrow F \setminus F_k$
        Append k to CH_{final}
      end if
    end if
  end for
end for
for all chunk k in CH_{final} **do**
  totalCost $\leftarrow$ totalCost $+ c_u \times (\delta_{ue}^k - \tau_{ue}^k)$
  totalMatched $\leftarrow$ totalMatched $+ F_k$
end for
return totalCost, totalMatched, CH_{final}

The pseudo code for the online algorithm, called *B-MinCostMaxSubMatch-Online*, is shown in Algorithm 3. The online algorithm runs for T timeslots. At each timeslot t $(1 \leq t \leq T)$, if an event e (or a subscription f) occurs (or arrives), it is added to E (or F). Two cases are possible at any timeslot t.

- *Some of the subscriptions that arrived at t have some matching event, or the events that are queued at t have some matching subscriptions*. In this case, *B-MinCostMaxSubMatch-Offline* is invoked, and the set of possible chunks is stored in CH_{Off}. However, all of them will not be scheduled. A chunk k with chunk interval $[\tau_{ue}^k, \delta_{ue}^k]$ will be added to the final solution CH_{final} if and only if $t = \tau_{ue}^k$. The set of subscriptions in k is removed from F. A chunk p with chunk interval $[\tau_{ue}^p, \delta_{ue}^p]$ will remain stored at CH_{Off} if $t < \tau_{ue}^p$.
- *The subscriptions that arrived at t do not have any matching event, and the events that are queued at t do not have any matching subscriptions*. In this case, *B-MinCostMaxSubMatch-Offline* will not be executed at t. *B-MinCostMaxSubMatch-Online* will check whether any chunk in CH_{Off} has $t = \tau_{ue}^k$ or not. If it finds a chunk k with $t = \tau_{ue}^k$, then it will be added to CH_{final}, and the subscriptions in k are removed from F.

TABLE II
SIMULATION PARAMETER SETTINGS

Parameters	Settings
City map	Tokyo
No of zones	105 (= 7 × 15)
No of busy zones	18
No of less-busy zones	87
Maximum no of RSUs	459
RSU Placement across zones	Uniform
RSU broadcast radius	300-500 m
cap_u for RSU u in busy zone	5
cap_u for RSU u in less-busy zone	2
c_u for RSU u in busy zone	25-30 unit
c_u for RSU u in less-busy zone	10-15 unit
No of events	10
Maximum no of subscriptions	20000
Time of issuing of a subscription	1-40 min
$d_{uf} - t_{uf}$	1-3 min
Average vehicle speed	20-60 km/h
Total travel time of a vehicle	45-100 min
Vehicle route from source to destination	Shortest-distance path

At $t = T$, the total cost is calculated from CH_{final} by summing up the cost of disseminating all events from all RSUs. Also, the total number of subscriptions that are matched are calculated by adding up sizes of all the chunks in CH_{final}.

Consider the same example mentioned in Section IV-A with the following additional input. The subscriptions f_1 and f_2 have occurred at $t = 9$ whereas the subscriptions f_3 and f_4 have occurred at $t = 10$. Algorithm 3 works as follows. It invokes Algorithm 2 with subscriptions f_1 and f_2 at $t = 9$, which gives a dissemination of e_1 from u_1 and u_5 for chunk intervals [9, 11] and [14, 16] respectively to match subscriptions f_1 and f_2 respectively. The dissemination of e_1 from u_1 is done immediately as the start time is equal to the current time, $t = 9$. However, the dissemination of e_1 at u_5 is deferred as its start time is 14. At $t = 10$, Algorithm 3 invokes Algorithm 2 with subscriptions f_2, f_3, and f_4, which gives a dissemination of e_1 from RSU u_4 for chunk intervals [10, 12] to match f_2 and f_3, and a dissemination of e_2 from RSU u_2 for chunk intervals [12, 14] to match f_4. The dissemination of e_1 from u_4 is done immediately as the start time is equal to the current time, $t = 10$. However, the dissemination of e_2 from u_2 is deferred as its start time is 12. At $t = 12$, e_2 is actually disseminated from u_2 to match f_4. Thus the total minimum cost incurred is $((11-9) \times 5 + (12-10) \times 6 + (14-12) \times 6) = 34$. The total number of subscriptions matched is 4.

Note that if we run the offline algorithm at each time interval and disseminate the event from the RSUs immediately, e_1 will be disseminated from u_1 and u_5 at $t = 9$, incurring a cost of $((11-9) \times 5 + (16-14) \times 5) = 20$. At time $t = 10$, when f_3 and f_4 occur, e_1 and e_2 will now be disseminated from u_4 and u_2 respectively for time intervals [10, 12] and [12, 14] respectively, and the cost is $((12-10) \times 6) + (14-12) \times 6) = 24$. Therefore, without deferred dissemination, the total cost would be $(20 + 24) = 44$. Thus the deferred dissemination in the online algorithm can help in reducing the total cost incurred by the SP.

TABLE III
TRAFFIC SCENARIOS

Scenario	Scenario Description	Settings
1	Traffic moves from different parts of a city toward a central location like the downtown or the office district	Traffic from 5-7 less-busy zones (source) to one busy zone (destination). From each source zone, 2500-4000 vehicles move toward the destination zone
1a	Traffic moves from different parts of a city toward more than one central locations	Traffic from 5-7 less-busy zones (source) to two busy zones (destination). For each pair of source and destination zones, 1400-2000 vehicles move from the source zone to the destination zone
2	Traffic moves from a central location like the downtown or the office district to different parts of a city	Traffic from one busy zone (source) to 5-7 less-busy zones (destination). From the source zone, 2500-4000 vehicles move toward each of the destination zones
2a	Traffic returns from more than one central locations to different parts of a city	Traffic from two busy zones (source) to 5-7 less-busy zones (destination). For each pair of source and destination zones, 1400-2000 vehicles move from the source zone to the destination zone
3	Traffic is random	Randomly select 4-5 source zones and 4-5 destination zones. For each pair of source and destination zones, 800-1250 vehicles move from the source zone to the destination zone

VI. SIMULATION RESULTS

We have evaluated the performance of the proposed algorithms through simulations over realistic city traffic scenarios using the city map of Tokyo [1]. The city map is divided in a 7×15 grid, where each grid-cell represents a zone in the city. A zone is characterized as a busy zone if the number of four-road junctions in it is above a threshold.

The RSUs are placed in some of these junctions. However, the exact placement of the RSUs varies across the simulations carried out. The zones from where a vehicle starts and ends its journey are referred to as its source zone and destination zone respectively. Given the source and the destination zones of a vehicle, two locations are picked from each zone as its source location and destination location; the route of a vehicle is the set of RSUs in the shortest path from the source location to the destination location.

For a subscription f, its t_{uf} values are generated in the order the RSUs are encountered in the route. For two successive RSUs in the route of f, the difference between t_{uf} values is proportional to the shortest path distance between them. The difference between d_{uf} and t_{uf} for the same u is randomly chosen between 1–3 minutes. These values are chosen based on the underlying assumption that the broadcast range of a RSU has a radius between 300–500 m, and the average speed of the vehicle is between 20–60 km/h. The time of occurrence of a subscription or an event varies randomly between 1–40 minutes from the beginning of simulation (from the simulation data, a vehicle has been seen to take typically between 45–100 minutes to reach its destination). Ten events are considered in the simulation, i.e., any subscription raised can be matched by exactly one of the ten events.

For a RSU u, the c_u values in a less-busy zone and a busy zone are chosen randomly between 10–15 unit and 25–30 unit respectively, and the capacity cap_u values in a less-busy zone and a busy zone are taken as 2 and 5 respectively. The RSUs are uniformly placed across all zones. Table II summarizes the different parameters and settings for the simulation. Table III summarizes the traffic scenarios studied.

For each of the five scenarios stated in Table III, we have plotted the variation of the percentage of subscription matched and the average cost for event dissemination against the number of subscriptions and RSUs. All results reported are the average of 250 runs. The 95% confidence interval is calculated, and is found to be within $\pm 5\%$ of the reported results for all cases. In each graph, three curves are shown: (i) *MaxSub-Off* for the *B-MaxSubMatch-Offline* algorithm, (ii) *MinCostMaxSub-Off* and (iii) *MinCostMaxSub-On* for the *B-MinCostMaxSubMatch-Offline* algorithm and *B-MinCostMaxSubMatch-Online* algorithm respectively.

A. Effect of Number of Subscriptions

In experiments, we have used 459 RSUs which is the total number of four-road, five-road, and six-road junctions in Tokyo. In each zone, the RSUs are placed in the road junctions as per the RSU placement scheme. In this subsection, we study the effect of the number of subscriptions on the percentage of subscriptions matched [Fig. 1(a)–(c)] and the average cost for event placement [Fig. 1(d)–(f)] for traffic scenarios 1, 2, and 3. The effect of the number of subscriptions for Scenario 1a and 2a are similar to Scenario 1 and 2 respectively and are not shown.

In Scenario 1, both the offline algorithms are able to match nearly 100% of the subscriptions. In this scenario, the traffic converges from different less-busy zones toward a single busy zone. The vehicles that start their journey at the same time tend to pass by the RSUs in the source zones almost at the same time with high degree of overlap. This results in the creation of many large sized chunks with smaller chunk intervals at these RSUs. On the other hand, vehicles that start their journey at different times still form a large number of small sized chunks with small chunk intervals at the RSUs near the destination zone. This occurs as all vehicles go to a single destination zone, making them all pass by the same RSUs near the end of their journey. Even if they start at different times, this results in significant overlap between at least some subscriptions. *MinCostMaxSub-Off* algorithm selects the lower AOC chunks at the RSUs near the source zones as much as possible. However, it still finds enough chunks at the RSUs near the destination zone to match most of the remaining subscriptions. On the other hand, *MaxSub-Off* selects RSUs with higher ASSI values, which occurs more in the RSUs near the destination zone, and compactly packs the maximal chunks there. Some other RSUs are also used in the rest of the route thereafter. From the simulation log, it is found that in Scenario 1, the *MaxSub-Off* uses less number of RSUs (25–30) with more compact packing of the chunks whereas *MinCostMaxSub-Off* uses more RSUs (30–40).

In Scenario 2 [Fig. 1(b)], both *MaxSub-Off* and *MinCostMaxSub-Off* match almost 80% subscriptions using RSUs close to the source zone. In Scenario 2, the traffic starts

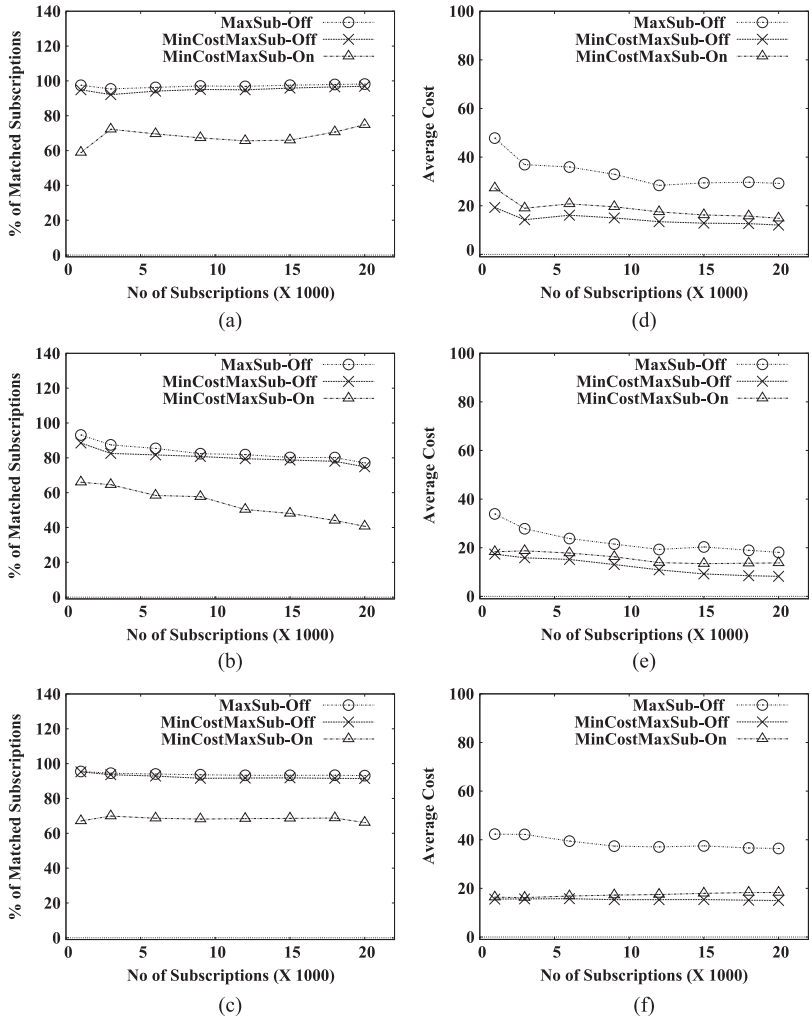

Fig. 1. Effect of the number of Subscriptions. (a) Scenario 1: % of subscriptions matched. (b) Scenario 2: % of subscriptions matched. (c) Scenario 3: % of subscriptions matched. (d) Scenario 1: Average cost. (e) Scenario 2: Average cost. (f) Scenario 3: Average cost.

from a single busy zone and diverges to different less-busy zones. The source zone has RSUs with higher capacity. *MaxSub-Off* selects these RSUs since they have higher ASSI values, and matches most of the subscriptions by disseminating events from those RSUs. However, lower capacity of the RSUs that are situated further in the routes of the vehicles prevents events to be placed in sufficient number thereby causing lower percentage of subscriptions matched in Scenario 2 than in Scenario 1. *MinCostMaxSub-Off* selects most of the RSUs that are situated near the source zone to match subscriptions that are raised from vehicles starting almost at the same time. In these RSUs, larger size chunks with smaller chunk intervals are formed because of the same reason stated earlier. However, some of the RSUs that are situated further in the routes of the vehicles in the less-busy zones are also used to match subscriptions that are raised at different times. Some small sized chunks having larger chunk intervals are formed at these RSUs due to the same reason stated earlier. Since there are lower number of RSUs in the less-busy zones, smaller number of events can be disseminated from these RSUs thereby causing lower percentage of subscriptions matched in Scenario 2 than in Scenario 1. From the simulation log, it is found that in Scenario 2, the *MaxSub-Off* uses less number of RSUs (18–22) with more compact packing of the chunks whereas *MinCostMaxSub-Off* uses more RSUs (25–30).

From simulation log, it is observed that in Scenario 1a and 2a, the routes have some overlap in the middle of their journey as well. In Scenario 1a and 2a, some RSUs are also selected by *MinCostMaxSub-Off* from the zones where overlap happens, in addition to the RSUs that are used in Scenario 1 and 2 respectively. In effect, higher number of RSUs (37–42 RSUs for *MaxSub-Off*, 45–50 RSUs for *MinCostMaxSub-Off* in Scenario 1a and 27–30 RSUs for *MaxSub-Off* and 35–39 RSUs for *MinCostMaxSub-Off* in Scenario 2a) are used to match subscriptions in Scenario 1a and 2a.

In Scenarios 1 and 2 [Fig. 1(d) and (e)], it is observed that *MinCostMaxSub-Off* outperforms *MaxSub-Off* in terms of the average cost for event dissemination, even though they give identical performance in the percentage of subscription matched. The reason is that *MaxSub-Off* gives priority to those RSUs in the busy zones with higher capacity and cost over the RSUs in the less-busy zones with lower capacity and cost thereby increasing the average cost for event dissemination. The difference between the average costs calculated by *MinCostMaxSub-Off* and *MaxSub-Off* is found to be much lower in Scenario 2 than in Scenario 1. In Scenario 2, both

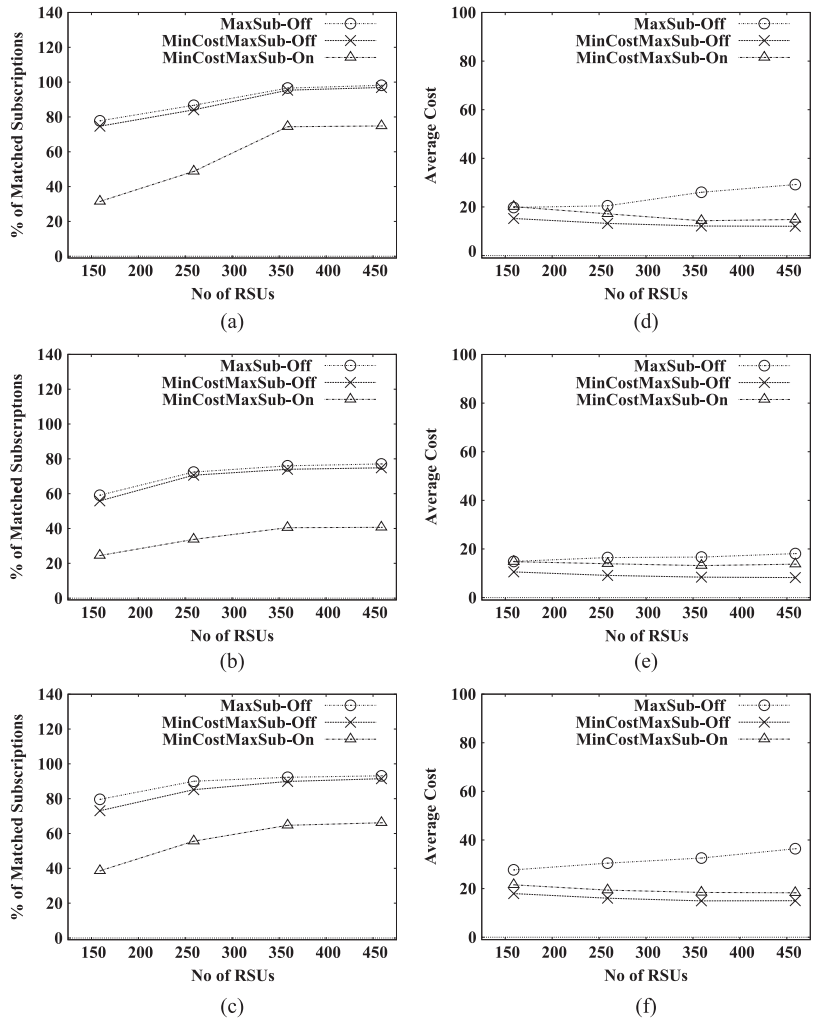

Fig. 2. Effect of the number of RSUs. (a) Scenario 1: % of subscriptions matched. (b) Scenario 2: % of subscriptions matched. (c) Scenario 3: % of subscriptions matched. (d) Scenario 1: Average cost; (e) Scenario 2: Average cost, (f) Scenario 3: Average cost.

offline algorithms select almost the same set of RSUs for matching the subscriptions. The reasons for selecting almost identical set of RSUs by the algorithms are stated earlier.

In all scenarios [Fig. 1(d)–(f)], it is observed that the average cost for matching subscriptions calculated by *MinCostMaxSub-Off* almost remains constant with increasing number of subscriptions. Since the time for passing by a RSU for a subscription is small, it minimizes the amount of overlap with other subscriptions of the same type. Therefore, the chunk size increases linearly with the number of subscriptions thereby increasing the total cost for event placement, and the average cost is found to be constant.

In all scenarios [Fig. 1(d)–(f)], it is observed that the average cost for matching subscriptions calculated by *MinCostMaxSub-On* is almost identical to *MinCostMaxSub-Off* whereas the percentage of subscription matched [Fig. 1(a)–(c)] is lower for *MinCostMaxSub-On* compared to *MinCostMaxSub-Off*. This is because of the fact that deferred placement in the RSUs helps to decrease cost of event placement; however, it minimizes the options for event placement in terms of the number of RSUs in the routes of the vehicles. In addition, the capacity constraints in those RSUs restrict the number of events to be placed.

Fig. 1(c) and (f) show how the percentage of subscription matched and the average cost vary with the number of subscriptions in Scenario 3. As before, both offline algorithms perform better than *MinCostMaxSub-On* in terms of the percentage of subscription matched. Since the source and the destination zones are randomly chosen, the overlaps between the routes do not follow any pattern. The spatial distribution of RSUs for matching the subscriptions is found to be scattered over the entire city.

On analyzing the results, it is observed that both *MaxSub-Off* and *MinCostMaxSub-Off* are able to match more than 80% subscriptions in all scenarios. In particular, in Scenario 1 and 1a, almost 100% subscriptions are matched. However, using *MinCostMaxSub-Off*, subscriptions are matched with reasonably lower cost. In most scenarios, *MinCostMaxSub-On* performs almost identical to *MinCostMaxSub-Off* in terms of the average cost for event dissemination.

B. Effect of Number of RSUs

In this section, we study the effect of the number of RSUs on the percentage of subscriptions matched [Fig. 2(a)–(c)] and the average cost for event placement [Fig. 2(d)–(f)] for traffic

scenarios 1, 2, and 3 while keeping the number of subscriptions constant at 20 000. The effect of the number of RSUs for Scenario 1a and 2a are similar to Scenario 1 and 2 respectively and are not shown.

With lower number of RSUs (around 159) distributed uniformly across zones, each zone has lower number of RSUs. It is observed that 80% of the subscriptions are matched in Scenario 1 [Fig. 2(a)] whereas 60% of the subscriptions are matched in Scenario 2 [Fig. 2(b)] using *MaxSub-Off* and *MinCostMaxSub-Off*. In Scenario 1, *MinCostMaxSub-Off* algorithm selects many lower AOC chunks at the RSUs near the source zones. However, some of the RSUs near the destination zone are also used to match the remaining subscriptions. On the other hand, *MaxSub-Off* selects RSUs with higher ASSI values, which occurs more in the RSUs near the destination zone, and compactly packs the maximal chunks there. Some other RSUs are also used in the rest of the route thereafter. In Scenario 2, both offline algorithms select RSUs only from the source zone and its vicinity to match large sized chunks with low AOC.

With higher number of RSUs (around 359), it is observed that 100% of the subscriptions are matched in Scenario 1 [Fig. 2(a)], whereas 80% of the subscriptions are matched in Scenario 2 [Fig. 2(b)] using *MaxSub-Off* and *MinCostMaxSub-Off*. Beyond 359 RSUs, the percentage of subscription matched almost remains constant. Higher number of RSUs ensures more number of RSUs in each zone with uniform RSU placement scheme. As a result, both offline algorithms perform well in Scenario 1 and Scenario 2.

For higher number of RSUs, the percentage of subscription matched by *MinCostMaxSub-On* is better in Scenario 1 than in Scenario 2. In Scenario 1, the destination zone is a busy zone having RSUs with higher capacity whereas in Scenario 2 the destination zones are less busy zones having RSUs with lower capacity. Since *MinCostMaxSub-On* uses deferred dissemination strategy for minimizing the cost while ensuring better overlap of subscriptions, it chooses RSUs placed further in the routes of the vehicles. As a result, *MinCostMaxSub-On* is able to match higher number of subscriptions in Scenario 1 than in Scenario 2.

In all scenarios [Fig. 2(d)–(f)], it is observed that the average cost remains constant for *MinCostMaxSub-Off*. We have seen that the uniform increase in the number of RSUs in each zone results in an almost linear increase in the number of subscriptions matched. Due to small difference between d_{uf} and t_{uf} values, the overlaps among the subscriptions are less, and the increase in the number of subscriptions increases the chunk interval linearly thereby causing a nearly constant average cost.

On analyzing the results, it is observed that in all scenarios, *MinCostMaxSub-Off* is able to match more than 80% subscriptions with reasonably lower average cost of event dissemination when the number of RSUs are higher than a threshold (> 359 in our case). The delaying by *MinCostMaxSub-On* helps in reducing event dissemination cost for most of the scenarios.

VII. Conclusion

In this paper, we have used a publish-subscribe based communication framework for a service provider (SP) that utilizes its RSUs to deliver relevant events to moving vehicles. We have formulated the *B-MaxSubMatch* and the *B-MinCostMaxSubMatch* problems for event dissemination through RSUs with finite capacity. To the best of our knowledge, this is the first work that considers validity period of events and subscriptions, cost of usage of RSUs, and the capacity of the RSUs. Simulation results show that *MaxSub-Off* and *MinCostMaxSub-Off* match almost 100% of the subscriptions in most scenarios. In addition, *MinCostMaxSub-Off* outperforms *MaxSub-Off* in terms of cost for event dissemination. In most scenarios, *MinCostMaxSub-On* performs almost identical to *MinCostMaxSub-Off* in terms of the average cost for event dissemination. It is also seen that a smaller number of RSUs distributed effectively based on traffic pattern can help in efficient dissemination of events.

The work can be extended to study the problem where V2V communication can be used along with V2I communication to notify vehicles with the help of other vehicles in its vicinity.

Appendix A
NP-Completeness of B-MinCostMaxSubMatch

We first define a restricted version of the *B-MinCostMaxSubMatch* problem, where we relax the condition of minimum cost for event placement in RSUs. Instead of minimum cost, we consider a fixed cost p, and the SP's objective is to maximize the number N of subscriptions matched subject to the constraint that the cost P for event placement in the RSUs is less than or equal to p. We refer to this restricted version of the *B-MinCostMaxSubMatch* problem as the *Bounded Fixed Cost Maximum Subscription Matching (B-FixCostMaxSubMatch)* problem. The output of *B-FixCostMaxSubMatch* is a set of variables $\{x_{\text{ueft}}\}$, where $x_{\text{ueft}} = 1$ if the event $e \in E$ will be placed in the RSU $u \in U$ at tth timeslot to match the subscription $f \in F$, else it is 0.

The decision version of *B-FixCostMaxSubMatch problem (D-B-FixCostMaxSubMatch)* is to decide that given two non-negative integers p and l, whether there exists an assignment of 0 or 1 to the set of variables x_{ueft} such that $N \geq l$ and $P \leq p$. The *D-B-FixCostMaxSubMatch* problem is next proved to be NP-complete by reducing it from the *Maximum Coverage Problem (MCP)* [18] which is known to be an NP-Complete problem. Clearly, this also implies that *B-MinCostMaxSubMatch* problem is NP-Complete.

The decision version of *Maximum Coverage Problem (D-MCP)* is defined as follows:

[**D-MCP**]: Given two non-negative integers n and k, and a finite set of sets $S = \{S_1, \ldots, S_m\}$, Does there exist a subset of $T \subseteq S$ such that $|\cup_{V \in T} V| \geq n$ and $|T| \leq k$?

Theorem 1: The D-B-FixCostMaxSubMatch problem is NP-complete.

Proof: Given a certificate for the problem $\{x_{\text{ueft}}\}$, we can calculate the total number of subscriptions N actually matched and verify whether all other constraints are satisfied or not in polynomial time. Hence, *D-B-FixCostMaxSubMatch* $\in$ NP.

We now show that *D-MCP* $\leq_P$ *D-B-FixCostMaxSubMatch*, by reducing an instance of the *D-MCP* to an instance of the *D-B-FixCostMaxSubMatch* in polynomial time. Suppose,

an input instance M of *D-MCP* consists of a non-negative integer n, a non-negative integer k and a finite collection of sets $S = \{S_1, \ldots, S_m\}$. Then, the input instance D of *D-B-FixCostMaxSubMatch* is defined as follows.

1) $p = k$,
2) $l = n$,
3) $|U| = m$, where each RSU $u_i \in U$ corresponds to subset S_i,
4) $\forall\, u \in U,\ c_u = 1$ and $\text{cap}_u = 1$,
5) $F = S_1 \cup, \ldots, \cup S_m$, where each subscription f_i corresponds to a unique element in the set $S_1 \cup, \ldots, \cup S_m$,
6) $E = \{e\}$, where $\forall\, f \in F$ $\text{type}_f = \text{type}_e$, and
7) $\forall\, i \in \{1, \ldots, m\}\ \forall\, f \in u_i,\ t_{u_i f} = 2 \times i$ and $d_{u_i f} = 2 \times i + 1$, i.e., a subscription f will pass by a RSU u_i for time $[t_{u_i f}, d_{u_i f}]$.

To complete the proof, we have to show that an output instance M^* of *D-MCP* has covered the number of elements greater than or equal to n if and only if the output instance D^* of *D-B-FixCostMaxSubMatch* has matched the number of subscriptions greater than or equal to l. Let M^* consists of a subset $S^* = \{S_1^*, S_2^*, \ldots, S_{|T|}^*\}$. By our reduction, each S_i^* corresponds to a unique RSU $u_j \in U$. The output instance D^* of *D-B-FixCostMaxSubMatch* is constructed as follows:

$$x_{\text{ueft}} = \begin{cases} 1 & \text{if } \exists i \in [1, \ldots, |T|] \text{ such that} \\ & (u = S_i^* \wedge f \in S_i^* \wedge t \in [t_{uf}, d_{uf}]) \\ 0 & \text{otherwise.} \end{cases}$$

Since M^* is a solution of *D-MCP*, $|\bigcup_{S_i^* \in M^*} S_i^*| \geq n$ and $|T| \leq k$. Since each S_i^* corresponds to a unique RSU $u_j \in U$, and each element in S_i^* corresponds to a unique subscription, $|\bigcup_{u_j \in U} F_{u_j}| \geq l$ and $|T| \leq p$, where F_{u_j} denotes the set of subscriptions matched from u_j in time $[t_{u_j f}, d_{u_j f}]$, i.e., the subscriptions matched from u_j in time $[2 \times j,\ 2 \times j + 1]$. Hence, D^* is a valid solution of *D-B-FixCostMaxSubMatch*.

We next show that a solution of *D-B-FixCostMaxSubMatch* gives a solution of *D-MCP*. Suppose that the output instance D^* of *D-B-FixCostMaxSubMatch* $\{x_{\text{ueft}}\}$ has a cost $P \leq p$ with the number of subscriptions matched $N \geq l$. From this, we construct an output instance M^* of *D-MCP* as follows. For each RSU u in D^*, we include the corresponding $S_u \in S$ in M^*. Each S_u contains the elements that correspond to the subscriptions that are matched by event e broadcast from u. Let the sets in M^* be $\{S_1^*, S_2^*, \ldots, S_P^*\}$.

Algorithm 4 Maximal Chunk Finding Algorithm

```
1: mcnkList ← φ
2: F' ← Sort F_ue in non-decreasing order of t_uf values. If
   t_uf_p = t_uf_q for two subscriptions f_p and f_q, then f_p is put
   before f_q in the sorted order provided d_uf_p ≤ d_uf_q
3: f'_1 ← 1st subscription in F'
4: mcnk ← f'_1
5: τ ← t_uf'_1
6: δ ← d_uf'_1
7: for (i ← 2; i ≤ |F_ue|; i ← i + 1) do
8:   if (τ ≤ t_uf'_i ≤ δ) then
9:     mcnk ← mcnk.append(f'_i)
10:    τ ← minimum of τ and t_uf'_i
11:    δ ← maximum of δ and d_uf'_i
12:  else
13:    mcnkList ← mcnkList.append(mcnk)
14:    mcnk ← f'_i
15:    τ ← t_uf'_i
16:    δ ← d_uf'_i
17:  end if
18: end for
19: return mcnkList
```

Then, since for a given u_j, the broadcast of event e is done for the entire duration $[2 \times j,\ 2 \times j + 1]$, all the subscriptions are matched simultaneously. Since the cost $P \leq p$, this implies that the number of RSUs selected is P. These RSUs are used to match $N \geq l$ subscriptions. Therefore, the number of elements covered is N which is greater than or equal to $l = n$ and these elements belong to P sets which is less than or equal to $p = k$. Hence, M^* is a valid solution of *D-MCP*. Therefore, *D-B-FixCostMaxSubMatch* is NP-complete. ■

APPENDIX B
ALGORITHMS TO COMPUTE MAXIMAL CHUNKS AND CHUNKS IN A MAXIMAL CHUNK

As stated earlier, maximal chunks are non-overlapping. The pseudo code for finding all maximal chunks from a set of subscriptions at a RSU is shown in Algorithm 4. Let F_{ue} denote the set of subscriptions at RSU u that matches the event e. Sort all the subscriptions in the RSU u in non-decreasing order of t_{uf} values. If $t_{uf_p} = t_{uf_q}$ for two subscriptions f_p and f_q, then f_p is put before f_q in the sorted order provided $d_{uf_p} \leq d_{uf_q}$. Let, the sorted order be $\{f'_1, \ldots, f'_{|F_{ue}|}\}$. The algorithm runs for $|F_{ue}|$ rounds. In the first round, the maximal chunk mcnk is initialized to $\{f'_1\}$ with the chunk interval $[\tau, \delta]$, where $\tau = t_{uf'_1}$ and $\delta = d_{uf'_1}$. In the second round, If f'_2 overlaps with mcnk having the chunk interval $[\tau, \delta]$ $(\tau \leq t_{uf'_2} \leq \delta)$, then f'_2 is added to mcnk, and the chunk interval of mcnk is updated as follows. τ is set to the minimum of τ and $t_{uf'_2}$, and δ is set to the maximum of δ and $d_{uf'_2}$. If f'_2 does not overlap with the mcnk $(\delta < t_{uf'_2})$, then mcnk is a maximal chunk with f_1 as the only subscription because the subscriptions $f'_3, \ldots, f'_{|F_{ue}|}$ cannot be added to mcnk as the t_{uf} values of $f'_3, \ldots, f'_{|F_{ue}|}$ must be greater than δ. Now, mcnk is initialized to $\{f'_2\}$ with the chunk interval $[\tau, \delta]$, where $\tau = t_{uf'_2}$ and $\delta = d_{uf'_2}$. Then the algorithm goes into the next round. At the end of $|F_{ue}|$ rounds, all maximal chunks are determined. To calculate the time complexity, sorting the subscriptions takes $O(|F_{ue}| \log_2 |F_{ue}|)$ time, followed by an execution of the loop for $|F_{ue}|$ times requires $O(|F_{ue}|)$ time. Therefore, the time complexity of the Algorithm 4 is $O(|F_{ue}| \log_2 |F_{ue}|)$.

For example, if $F_{ue} = \{f_1, f_2, f_3, f_4, f_5\}$ with time intervals of [2, 5], [3, 8], [4, 6], [10, 13], and [11, 14] respectively, the maximal chunks are $\{f_1, f_2, f_3\}$ and $\{f_4, f_5\}$ with chunk intervals [2, 8] and [10, 14] respectively.

As stated earlier, a chunk can only be formed with subscriptions of identical types (i.e., for a single event) for a single RSU. Given a maximal chunk mcnk with the set of subscriptions F_{mcnk}, the pseudo code for finding all possible chunks is shown in Algorithm 5. Sort F_{mcnk} in non-decreasing order of t_{uf} values. If $t_{uf_p} = t_{uf_q}$ for two subscriptions f_p and f_q, then f_p is put before f_q in the sorted order provided $d_{uf_p} \leq d_{uf_q}$. This sorting takes $O(|F_{\text{mcnk}}| \log_2 |F_{\text{mcnk}}|)$ time. For $|F_{\text{mcnk}}|$ number of subscriptions in mcnk, we have $|F_{\text{mcnk}}|$ number of t_{uf} and d_{uf} values. This implies that the maximum number of chunks is upper bounded by $|F_{\text{mcnk}}|^2$. For each of the $|F_{\text{mcnk}}|^2$ intervals, it is possible to find out the set of subscriptions that belongs to it in $O(|F_{\text{mcnk}}|)$ time (Lines 6–11), and to verify whether the set of subscriptions constitutes a chunk or not in $O(|F_{\text{mcnk}}|)$ time (Lines 12–25) [17]. Therefore, the time complexity of the Algorithm 5 is $O(|F_{\text{mcnk}}|^3)$.

Algorithm 5 Chunk Finding Algorithm

```
1: cnkList ← φ
2: F_mcnk ← Subscriptions in the maximal chunk mcnk
3: Sort F_mcnk in non-decreasing order of t_uf values.
   If t_uf_p = t_uf_q for two subscriptions f_p and f_q, then f_p
   is put before f_q in the sorted order provided d_uf_p ≤ d_uf_q
4: for (i ← 1; i ≤ |F_mcnk|; i ← i + 1) do
5:   for (j ← 1; j ≤ |F_mcnk|; j ← j + 1) do
6:     cnk ← φ
7:     for (k ← 1; k ≤ |F_mcnk|; k ← k + 1) do
8:       if (t_uf_i ≤ t_uf_k, d_uf_k ≤ d_uf_j) then
9:         cnk ← cnk.append(f_k)
10:      end if
11:    end for
12:    isValidChunk ← true
13:    stack ← Create an empty stack
14:    Push the first interval of cnk in stack
15:    for all (interval [t', d'] in cnk) do
16:      if ([t', d'] does not overlap with the interval in
         stack.top) then
17:        isValidChunk ← false
18:        break
19:      else
20:        [t, d] ← Pop the interval from stack
21:        τ ← minimum of t and t'
22:        δ ← maximum of d and d'
23:        Push [τ, δ] in stack
24:      end if
25:    end for
26:    if (isValidChunk = true) then
27:      cnkList ← cnkList.append(cnk)
28:    end if
29:  end for
30: end for
31: return cnkList
```

Consider the earlier example where $F_{ue} = \{f_1, f_2, f_3, f_4, f_5\}$ with time intervals of [2, 5], [3, 8], [4, 6], [10, 13], and [11, 14] respectively. The chunks associated to the maximal chunk $\{f_1, f_2, f_3\}$ are $\{f_1\}$, $\{f_3\}$, $\{f_2, f_3\}$, and $\{f_1, f_2, f_3\}$ with chunk intervals [2, 5], [4, 6], [3, 8], and [2, 8] respectively. Similarly, the chunks associated to the maximal chunk $\{f_4, f_5\}$ are $\{f_4\}$, $\{f_5\}$, and $\{f_4, f_5\}$ with chunk intervals [10, 13], [11, 14], and [10, 14] respectively.

REFERENCES

[1] [Online]. Available: http://www.openstreetmap.org/

[2] F. Bai, H. Krishnan, V. Sadekar, G. Holl, and T. Elbatt, "Towards characterizing and classifying communication-based automotive applications from a wireless networking perspective," in *Proc. IEEE Workshop Autom. Netw. Appl.*, 2006, pp. 1–25.

[3] F. Chen, M. Johnson, Y. Alayev, A. Bar-Noy, and T. La Porta, "Who, when, where: Timeslot assignment to mobile clients," *Trans. Mobile Comput.*, vol. 11, no. 1, pp. 73–85, Jan. 2012.

[4] X. Cheng *et al.*, "Electrified vehicles and the smart grid: The ITS perspective," *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 4, pp. 1388–1404, Aug. 2014.

[5] X. Cheng, C.-X. Wang, B. Ai, and H. Aggoune, "Envelope level crossing rate and average fade duration of nonisotropic vehicle-to-vehicle Ricean fading channels," *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 1, pp. 62–72, Feb. 2014.

[6] X. Cheng, L. Yang, and X. Shen, "D2D for intelligent transportation systems: A feasibility study," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 4, pp. 1784–1793, Aug. 2015.

[7] S. Khakbaz and M. Fathy, "A reliable method for disseminating safety information in vehicular ad hoc networks considering fragmentation problem," in *Proc. 4th Int. Conf. Wireless Mobile Commun.*, 2008, pp. 25–30.

[8] J. M. Kleinberg and É. Tardos, *Algorithm Design*. Reading, MA, USA: Addison-Wesley, 2006.

[9] J. Lebrun, C.-N. Chuah, D. Ghosal, and M. Zhang, "Knowledge-based opportunistic forwarding in vehicular wireless ad hoc networks," in *Proc. IEEE VTC-Spring*, 2005, pp. 2289–2293.

[10] I. Leontiadis and C. Mascolo, "GeOpps: Geographical opportunistic routing for vehicular networks," in *Proc. IEEE Int. Symp. World Wireless, Mobile Multimedia Netw.*, 2007, pp. 1–6.

[11] J. C. Mukherjee and A. Gupta, "A publish-subscribe based framework for event notification in vehicular environments," in *Proc. 5th Int. Conf. Commun. Syst. Netw.*, 2013, pp. 1–10.

[12] J. C. Mukherjee and A. Gupta, "Mobility aware event dissemination in VANET," in *Proc. 16th Int. Conf. Distrib. Comput. Netw.*, 2015, p. 22.

[13] X. Shen, X. Cheng, L. Yang, R. Zhang, and B. Jiao, "Data dissemination in VANETs: A scheduling approach," *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 5, pp. 2213–2223, Oct. 2014.

[14] Y. Sung and M. Lee, "Light-weight reliable broadcast message delivery for vehicular ad-hoc networks," in *Proc. IEEE VTC-Spring*, 2012, pp. 1–6.

[15] M. Torrent-Moreno, "Inter-vehicle communications: Assessing information dissemination under safety constraints," in *Proc. 4th Annu. Conf. Wireless Demand Netw. Syst. Serv.*, 2007, pp. 59–64.

[16] L. A. Villas, A. Boukerche, R. B. de Araujo, A. A. F. Loureiro, and J. Ueyama, "Network partition-aware geographical data dissemination," in *Proc. IEEE Int. Conf. Commun.*, 2013, pp. 1439–1443.

[17] [Online]. Available: http://www.geeksforgeeks.org/merging-intervals/

[18] D. S. Hochbaum, *Approximation Algorithms for NP-Hard Problems*. Boston, MA, USA: PWS-Kent, 1997.

Joy Chandra Mukherjee received the M.Tech. degree in computer science and engineering in 2011 from the Indian Institute of Technology Kharagpur, Kharagpur, India, where he has been working toward the Ph.D. degree with the Department of Computer Science and Engineering since July 2011. His research interests include smart grid, mobile computing, and distributed algorithms.

Arobinda Gupta received the Ph.D. degree in computer science from The University of Iowa, Iowa City, IA, USA, in 1997. From 1997 to 1999, he was with the Windows 2000 Distributed Infrastructure Group, Microsoft Corporation, Redmond, WA, USA. Since October 1999, he has been a Faculty Member with the Indian Institute of Technology Kharagpur, Kharagpur, India, where he is currently a Professor with the Department of Computer Science and Engineering. His current research interests include distributed systems and mobile computing.

Ravella Chaitanya Sreenivas received the B.Tech. degree in computer science and engineering from the Indian Institute of Technology Kharagpur, Kharagpur, India, in 2014. He is currently a Software Engineer with Microsoft Development Center Serbia, Belgrade, Serbia. His research interests include mobile computing and distributed algorithms.