DOVE: Data Dissemination to a Desired Number of Receivers in VANET
Tan Yan, Wensheng Zhang, Member, IEEE, and Guiling Wang

Abstract—Efficient data dissemination to a desired number of receivers in a vehicular ad hoc network (VANET) is a new issue and a challenging one considering the dynamic nature of VANETs. We aim to accurately control the number of receivers in a particular area of interest, achieve low dissemination delay, and incur only small communication overhead. To achieve these goals, we designed the data dissemination to a desired number of receivers in VANET (DOVE) scheme. DOVE is inspired by processor scheduling treating roads as processors to optimize the workload assignment and improve the efficiency of on-road dissemination. DOVE reaches the desired number of receivers with little inaccuracy and minimizes the dissemination delay with low communication overhead. We enhance our protocol with workload backup to deal with vehicles that are quitting the network. We utilize the unique characteristics of VANET and propose heuristics accordingly to significantly reduce the dissemination delay and overhead. Simulation results show that our scheme can disseminate data to almost all the pre-given number of receivers with a light overhead and low delay.

Index Terms—Data dissemination, desired number, heuristics, processor scheduling, VANET.

I. INTRODUCTION

A. Motivation

A vehicular ad hoc network (VANET) is an on-road network and is envisioned to be applied in many applications for road safety, driving convenience, and commercial purposes [1]–[3]. Among them, data dissemination is an important category, and intensive research has been conducted to effectively propagate data to a large number of recipients in the network under different application requirements [4]–[12].

However, past research has lacked attention to one kind of data dissemination. We observe that the primary goal of many data dissemination applications is to efficiently reach a desired number of receivers in an area of interest. Here are a few examples: 1) An area always has accidents and congestion. To improve the road condition of the area, the Department of Transportation (DoT) needs to collect the feedback from drivers passing the area and, thus, needs to disseminate a survey to them. As is well known, the DoT provides compensation, e.g., $50, for filling the survey, and it has a budget. In addition, a certain number of responses are more than enough for the DoT to figure out the problem. Thus, the survey should be disseminated to only a desired number of receivers. To achieve this goal, a data dissemination scheme, which can control the number of receivers, is needed. 2) A museum or store wants to distribute a certain number of vouchers determined by the budget to local residents, which can be used as free tickets or discounts. A dissemination scheme that can reach a desired number of receivers is needed so that it does not overflow the budget. 3) In an existing on-road e-Ad dissemination system [13], it provides incentives to ad forwarders based on the receipts generated by ad receivers. A receiver number control mechanism is needed to control the total incentives given out. In fact, any budget-constrained data dissemination services that provide incentives to receivers, such as Digital Billboards [14], Electric Coupon System [15], and FleaNet [16], have the need to reach a certain number of receivers.

Past data dissemination schemes in VANET [4], [13], [17], [18] cannot support these applications since they do not control the number of receivers. Tailoring these schemes by merely adding number control cannot provide an efficient solution, considering that controlling the number of receivers in a distributed fashion when no central server is available is costly. To efficiently address the problem, this paper proposes a new scheme in VANET to reach the desired number of receivers in a particular area of interest.

B. Objective

In addressing the problem of a sender disseminating data or queries to a desired number of recipients, the primary objective is to disseminate data or query to a desired number of receivers with small overhead and little inaccuracy. For example, 0.2% of inaccuracy, i.e., 1002 receivers actually get the data although the desired number is 1000. Note that, to 100% ensure that the actual number of recipients is exactly the same as the desired number, it is likely to result in a big overhead and long delay. A solution has to first identify at least the desired number of recipients, e.g., 1000, and then keep sending data to them until it receives acknowledgements from them. Sending 1000 copies to unknown recipients cannot work even with acknowledgements, considering the distributed network environment with unreliable links and vehicles joining and leaving the network. For example, to ensure that one copy of the data is delivered to any recipient without identifying it, the receiver needs to confirm the reception, and thus, the sender knows that one copy has been

Manuscript received June 5, 2013; revised September 20, 2013; accepted September 25, 2013. Date of publication October 30, 2013; date of current version May 8, 2014. This work was supported by the National Science Foundation under Grant 1128369. The review of this paper was coordinated by Prof. Y. Zhang.

T. Yan and G. Wang are with the Department of Computer Science, New Jersey Institute of Technology, Newark, NJ 07102 USA (e-mail: ty7@njit.edu; gwang@njit.edu).

W. Zhang is with the Department of Computer Science, Iowa State University, Ames, IA 50011 USA (e-mail: wzhang@cs.iastate.edu).

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/TVT.2013.2287692

sent. However, if the confirmation is lost, there is no way to tell whether the receiver has not received the data or it has received the data but the confirmation is lost. Thus, the sender is still unaware whether the remaining copies to be sent are one less or still the same.

This paper focuses on designing a distributed algorithm to disseminate data or a query to a desired number of receivers in a specific area of interest with high accuracy, small overhead, and low delay. To the best of our knowledge, the short conference version of this paper [19] is the first one to address the problem in VANETs.

C. Methodology

The general idea behind this paper is to let the sender distribute the data dissemination tasks to multiple vehicles in different roads to lower the overhead and speed up the dissemination process, each of which is responsible for a portion of the number of copies. Considering that any vehicle in one road can be responsible for disseminating data to all the vehicles on the road as it moves on that road, the key is the traffic or the number of potential receivers on the road. The higher the traffic, the more receivers that can be reached with fewer messages. Therefore, in this paper, we employ the following philosophy: a road is viewed as a virtual processor, and its processing capability is its traffic. The higher the traffic of a road, the more copies should be assigned to the road. Centered at the location of the sender, the surrounding roads are multiple processors. The start working time of a processor is when one portion of the dissemination task is delegated to it, and its job is that portion of the task. The whole dissemination task is finished when all the processors finish their tasks, and the desired number of copies is sent.

To implement the idea, a shortest-path tree rooted at the sender’s road is calculated based on the road map. The tree indicates how to delegate the dissemination task to vehicles on the sender’s surrounding roads. The dissemination task and the tree are assigned to a car running on the road where the sender, from then on, is responsible for the dissemination task. The car moves and disseminates data to the vehicles on its road. Based on the shortest-path tree, it also cuts a branch of the tree and delegates a portion of the dissemination task to a vehicle on another road. The delegation is either by sending messages to that road through multihop routing or by directly sending the task when it physically reaches the intersection with that road. The vehicle being delegated further conducts data dissemination and also delegates a portion of its tasks based on its tree.

In real situations, where road traffic is not known beforehand, we propose a heuristic based on the newly obtained traffic information to dynamically adjust the shortest-path tree. To improve on-road dissemination efficiency, we further propose two heuristics based on the road layout. For example, the disseminators utilize vehicles on the same road but driving in an opposite direction to help them to reach more vehicles. They also relay their tasks to vehicles in front of them to speed up the dissemination. We also propose a backup strategy to deal with disseminating vehicles suddenly stopping and quitting the network. A distributed protocol, which is called DOVE, is designed to implement all the details. We evaluate DOVE under both normal density and sparse traffic density.

Note that our proposed scheme can be easily adopted to fulfill the requirements of many other number-related applications, such as how to reach a certain number of receivers that belong to a specific group, how to reach at most a certain number of receivers, and how to reach at least a certain number of receivers.

D. Assumptions

We assume that vehicles communicate with each other through a short-range wireless channel (dedicated short-range communication; 100–250 m) in VANETs. A vehicle knows its location and velocity through a global positioning system or various localization mechanisms [20]. We assume that vehicles are equipped with preloaded digital maps providing a street-level map and traffic statistics (i.e., traffic density and vehicle speed on each road), which can be obtained through various services such as Google Maps [21] and Bing Maps [22]. Vehicles can find their neighbors through periodical exchange of beacon messages, which can be efficiently done with various protocols such as S-Aloha [23], [24].

E. Organization

The rest of this paper is organized as follows. The theoretical solution is presented in Section II. Section III focuses on the distributed protocol DOVE. An evaluation of DOVE is given in Section IV. Section V presents the state of the art. Finally, this paper is concluded in Section VI.

II. THEORETICAL SOLUTION

Here, we discuss the solution in an ideal case, where the road traffic is known beforehand.

A. Problem Formulation

The problem to be addressed is as follows: A sender needs to disseminate data to $M$ recipients and collect $M$ receipts in an interested area consisting of $k$ roads. Given the traffic density in each road and the position of the sender, how many recipients should be disseminated in each road at what time and in what sequence, so that $M$ recipients in total can be reached with small overhead and the dissemination delay is minimized?

Before we proceed, we first introduce the following notations. We use $RD_i$ ($i = 1, \ldots, k$) or road $A, B, \ldots, K$ to denote the $k$ roads interchangeably. Please note that $RD_s$ specifically represents the road where the sender is located. We treat each road as a virtual processor. $R_i$ represents the traffic density of road $RD_i$, which is the number of new recipients that a disseminator can reach on $RD_i$ per time unit. $R_i$ can also be viewed as the processing capability of virtual processor $RD_i$. $m_i$ is the number of recipients that should be reached in $RD_i$. $m_i$ can be viewed as workload assigned to virtual processor $RD_i$. It is what we need to calculate. We use $D_i$ to denote the starting working time of virtual processor $RD_i$, where the time is when $RD_i$ is assigned the dissemination task.
Fig. 1. Data dissemination chart.

\( C_i \) represents when a virtual processor can finish its job, which is the time when recipients on road \( RD_i \) have received the data. The following is the matrix format of the data structure:

\[
R = \begin{bmatrix}
R_1 & \cdots & R_s \\
R_2 & \cdots & R_k
\end{bmatrix}
\quad D = \begin{bmatrix}
D_1 \\
\vdots \\
D_i \\
\vdots \\
D_k
\end{bmatrix}
\quad m = \begin{bmatrix}
m_1 \\
\vdots \\
m_i \\
\vdots \\
m_k
\end{bmatrix}
\quad C = \begin{bmatrix}
C_1 \\
\vdots \\
C_i \\
\vdots \\
C_k
\end{bmatrix}.
\] (1)

We plot the given items in a processor scheduling-like chart in Fig. 1. In Fig. 1, each road \( RD_i \) is a virtual processor, which is represented by a rectangle. The width represents \( R_i \), the processing capability, and the length represents time. The blank space represents the idle time of the processor, which means that the dissemination task has not been assigned to road \( RD_i \) because there is a certain distance between \( RD_i \) and the sender, and it takes some time for \( RD_i \) to be assigned a task. The shadow space is the working time of the processors, and its area is \( m_i \). \( m_i \) is the product of the processing capability \( R_i \) and its working time \( (C_i - D_i) \). The formula is shown as follows:

\[
\begin{aligned}
\{ m = R \cdot (C - D) \\
\sum m_i = M.
\end{aligned}
\] (2)

In addition, we use \( C_{\text{max}} \) to denote the time that the last processor finishes its job. \( C_{\text{max}} \) is the dissemination delay of the whole task. Our objective is to minimize \( C_{\text{max}} \).

B. Workload Assignment

**Theorem 2.1:** \( C_{\text{max}} = \max\{C_1, C_2, \ldots, C_k\} \) is minimized if and only if

\[
C_1 = C_2 = \ldots = C_k = C_{\text{max}}
\] (3)

when we ignore the roundoff.

*Proof of Theorem 2.1 (sketch):* The proof can be done by contradiction. If the optimal solution makes not all \( c_i \in C \) the same, suppose \( C_{\text{max}} = C_i \) is the largest value in \( C \) produced by that optimal solution, we can always cut a small piece \( \delta_c \) of \( C_{\text{max}} \) and add it to \( c_j \), which is the second largest one in \( C \), to make \( C_i - \delta_c = C_j + \delta_c \). Hence, we have \( C_{\text{max}} = C_i - \delta_c = C_j + \delta_c < C_i \), which contradicts that \( C_{\text{max}} = C_i \) is the optimal one.

From Theorem 2.1, we can see that, theoretically, the dissemination delay \( C_{\text{max}} \) will be minimized if the dissemination tasks on all the roads are finished at the same time. To minimize \( C_{\text{max}} \), \( C_i (i = 1, \ldots, k) \) must be equal to \( C_{\text{max}} \). To plug (3) into (2), we have

\[
C_{\text{max}} \cdot \sum_{i=1}^{k} R_i - \sum_{i=1}^{k} (D_i \cdot R_i) = M.
\] (4)

Then, the dissemination delay can be calculated as

\[
C_{\text{max}} = \frac{M + \sum_{i=1}^{k} (D_i \cdot R_i)}{\sum_{i=1}^{k} R_i}.
\] (5)

From the given formula, it can be seen that to minimize dissemination delay \( C_{\text{max}} \), \( D_i \) needs to be minimized. In the following, we first present how to calculate the minimized \( D_i \) and, based on the result, how to calculate \( m_i \). Then, we use a delegation tree data structure to describe how to conduct the workload assignments.

1) Optimum Parameter and Data Structure: \( D_i \) is the time it takes for a delegation message to reach a vehicle on road \( RD_i \), and thus, dissemination can start in \( RD_i \). The earliest possible delegation can only occur in the intersections since either routing or driving follows the road layout in vehicular networks and the intersections are the closest delegation places. Therefore, for a road \( RD_i \), which has an intersection with \( RD_s \), \( D_i \) can be the time in which a delegation message reaches intersection \( I_{ij} \) directly from \( RD_i \). For a road \( RD_j \), which has no intersection with \( RD_s \) but \( RD_k \), \( D_j \) then can be \( D_k \) plus the time it takes for a delegation message to reach \( I_{jk} \) from the disseminator in \( RD_k \).

Before we formally describe how to calculate the minimized \( D_i \), we first abstract the road map shown in Fig. 2(a) into a graph shown in Fig. 2(b). Each node in Fig. 2(b) represents a road. There is an edge from \( RD_i \) to \( RD_j \) if \( RD_i \) intersects \( RD_j \). Each edge is assigned a weight, which is the delegation message delivery delay from the source road to the destination road. Same as in [25] and [26], the delay can be calculated based on the historical statistical traffic information on each road segment through various methods, such as the one derived in [27]. The delegation starts from \( RD_s \) (road \( A \)), where the sender is. Therefore, to calculate the minimized \( D_i \), we need to find the shortest paths from node \( A \) to all the other nodes. The calculation can be done by many standard shortest-path algorithms such as Dijkstra’s algorithm. After the calculation, we obtain a shortest-path tree, as shown in Fig. 2(c). Then, for each node \( RD_i \) in the shortest-path tree, an edge is added from it to a new leaf node, as shown in Fig. 2(d). The extended edge represents the dissemination time to finish the task \( m_i \) assigned to \( RD_i \). The distance from the root to the new leaf node \( RD_i \) represents the task finishing time \( C_i \). All the tasks finish at the same time, and thus, \( C_{\text{max}} \) is minimized.

After obtaining the minimized \( D_i \), we calculate the workload \( m_i \) to be assigned to each road \( RD_i \), to minimize \( C_{\text{max}} \). Plugging (5) into (2), we get

\[
m_i = R_i \cdot \frac{M + \sum_{i=1}^{k} (D_i \cdot R_i)}{\sum_{i=1}^{k} R_i} - D_i.
\] (6)
We use Fig. 3 shows how the delegation tree data structure is utilized.

The delegation tree data structure and becomes responsible for only \( M - m_A - m_E \) copies, among which, \( m_D \) is its own task, and \( m_B + m_C \) belongs to its subtrees. As for \( v_i \), it updates its data structure and becomes responsible for only \( M - m_A - m_E \) copies, among which \( m_A \) is its own task, and \( m_E \) is what it will delegate to road \( E \). \( v_j \) will do the same job as \( v_i \) and sends a branch of its own tree to two vehicles on road \( B \) and \( C \), respectively. After that, \( v_j \) is only responsible for \( m_D \) copies.

**III. DISTRIBUTED SCHEME**

In the real world, road traffic cannot be known beforehand. The dissemination starting time on each road has a big variance depending on several factors. For example, in the off-peak time, the multihop communication link may be broken, and it may take a long time for a disseminating vehicle to assign the job of another road to one of its vehicles. Even reaching the intersection physically, a disseminator may fail to find a vehicle in the intersecting road to delegate a task to. In addition, a disseminating vehicle may stop and quit the network. Taking these real issues into consideration and further improving the dissemination efficiency, we propose several optimizations and a distributed protocol, i.e., *DOVE*, to implement them and deal with the real challenges.

**A. Workload Delegation**

An example of the workload delegation procedure is presented in Section II-B2 and shown in Fig. 3. In a distributed manner, there are two ways for disseminator \( v_i \) on one road, e.g., road \( A \), to look for a disseminator in its child road, e.g., road \( D \), to delegate the task. One is to send a delegation message through multihop communication to the intersection of road \( A \) and road \( D \) to search a disseminator. The other is to find a vehicle in the intersection when \( v_i \) physically drives through the intersection. Apparently, the former way is much more efficient, and we propose first using multihop communication to send the delegation message and search. It is only when no disseminator can be found in this way that \( v_i \) searches a disseminator on road \( D \) when it is physically at the intersection.

Specifically, disseminator \( v_i \) sets a timer and sends messages to search workload receivers. If it cannot find a vehicle to delegate a task to after the timer expires, it terminates the current session and relaunches a new session. The repeated procedure terminates if \( v_i \) finds a workload receiver or it already drives to the intersection physically.

The detailed procedure is as follows: Upon becoming a disseminator of road \( A \), \( v_i \) sends a trial message, asking for workload delegation, toward intersection \( I_{AD} \) of road \( D \). Vehicles on the routing path forward the received messages toward \( I_{AD} \) using geographic routing, which is shown as the solid arrow in Fig. 4. Suppose that vehicle \( v_j \) receives the message and detects that its distance to \( I_{AD} \) is shorter than the communication range, then it can reach vehicles in the intersection. If \( v_j \) itself turns onto road \( D \) at this intersection, then it accepts the workload. If not, \( v_j \) records its previous hop vehicle \( v_m \), sends it \( (M - m_A - m_E, m_D, T_D) \), and removes the branches from its own tree. \( v_j \) learns that it needs to disseminate data to \( m_D \) vehicles on road \( D \), and in total, it is responsible for \( M - m_A - m_E \) copies, among which, \( m_D \) is its own task, and \( m_B + m_C \) belongs to its subtrees. As for \( v_i \), it updates its data structure and becomes responsible for only \( M - m_A - m_E \) copies, among which \( m_A \) is its own task, and \( m_E \) is what it will delegate to road \( E \). \( v_j \) will do the same job as \( v_i \) and sends a branch of its own tree to two vehicles on road \( B \) and \( C \), respectively. After that, \( v_j \) is only responsible for \( m_D \) copies.

**Fig. 2.** Data structure in the dissemination.

**Fig. 3.** Parsing shortest-path delegation tree.
and then starts searching for the workload receiver, considering that it can directly reach the intersection. $v_j$ broadcasts in the intersection looking for the workload receivers.

If no vehicle in the intersection can accept the workload, $v_j$ informs its previous hop vehicle $v_m$ to continue the search, since it is leaving the intersection. $v_m$ will enter the search phase as $v_j$. If $v_j$ finds vehicle $v_k$ in the intersection and $v_k$ can be a workload receiver, $v_k$ sends an acceptance message to initiator $v_i$. Upon receiving the acceptance from $v_k$, $v_i$ calculates and assigns the workload along with the delegation tree to $v_k$. Note that it is possible that $v_j$ finds multiple vehicles in the intersection, and they all send an acknowledgement to the delegator. In this situation, the delegator simply picks the one with the least communication delay to delegate the task.

To reduce communication overhead in workload assignment, we let the disseminator pack the delegation messages according to the delegation tree. Before sending delegation messages, disseminator $v_i$ first classifies intersections according to their directions. Then, the delegation messages are routed to the intersections of the same directions together to avoid redundant forwarding. An example is shown in Fig. 5. Road $A$ needs to delegate tasks to its child roads $D$ and $E$. The disseminator $v_i$ on road $A$ then needs to send delegation messages to intersections $\{I_{AD}, I_{AE}\}$. Considering that the two intersections are in the same direction, instead of sending two separate messages, $v_i$ packs the two messages into one message and propagates the message along the road toward roads $D$ and $E$. When vehicle $v_j$ approaching intersection $I_{AD}$ receives the message, it first checks whether $I_{AD}$ is in the list. If yes, then $v_j$ searches for the workload receiver as previously described. At the intersection, the trial message will be further propagated to $I_{AE}$ until there is no intersection listed in the delegation message.

B. Relay the Disseminator Role Forward and Share it Backward

To speed up the dissemination and reduce the overhead, we employ two optimizations: to relay the dissemination role forward and to share it backward. The first one is motivated by an observation: Vehicles driving in the same direction are relatively static to each other. Even if a disseminator keeps broadcasting the message, a few new vehicles can be reached. Therefore, we propose relaying the disseminator role to the farthest receiver ahead of it after each broadcast.

To implement the idea, after receiving a message, each vehicle sends to the disseminator a confirmation along with its own location and driving direction. Hence, after disseminating a message, based on the received confirmation, disseminator $v_i$ can choose vehicle $v_m$ that drives in the same direction as $v_i$ ahead of it and is farthest to $v_i$ as the next disseminator. After $v_m$ gets the notification from $v_i$, it becomes the new disseminator. When $v_m$ finishes a dissemination, it repeats the same procedure of $v_i$. If $v_m$ reaches the boundary of the dissemination map, it relays the workload to a vehicle in the opposite driving direction. The procedure is shown in Fig. 6.

As to the second optimization, to share the disseminator role backward is to let the disseminator reach vehicles that drive in the opposite direction and will never meet it. Load sharing is only done once when the first disseminator is assigned the task. Suppose that vehicle $v_i$ is the first disseminator of a road. After its first broadcasting, $v_i$ finds vehicle $v_n$ that is the farthest message receiver at its back. Then, $v_i$ informs $v_n$ to be a disseminator and shares half of the workload to $v_n$. Note that $v_i$ can never reach the vehicles that $v_n$ is going to meet as it drives further.

C. Dynamic Workload Reassignment

1) General Idea: We have proved that if workload is assigned to each road in such a way that all the roads finish at the same time, the scheme achieves the minimum dissemination delay theoretically. However, in reality, with the changing traffic condition, no one-time workload assignment can be so perfect that all the roads finish their tasks at the same time. On the contrary, some roads finish while others are still working. The finishing time of each road follows a zigzag fashion as shown in Fig. 1, where $C_j$ is the job finishing time on road $RD_j$. The task delegation tree has a shape as shown in Fig. 7(a), instead of the desired one shown in Fig. 2(d).

To mitigate the issue, we propose conducting workload reassignment after the dissemination starts for a while. A portion of the unaccomplished jobs of the roads that have not finished dissemination will be assigned to the roads that have finished their originally assigned task. Reassignment is based on the amount of remaining workload and up-to-date traffic information of each road.
In detail, we will employ the delegation tree data structure, as shown in Fig. 7(a), to facilitate the reassignment. In the tree, each node has a level, which represents the distance to the root road and the approximate sequence of delegation. Lower level nodes delegate workload to higher level nodes. Before the reassignment, higher level nodes report to their parent node the up-to-date traffic information and how the workload has been finished in its branch. Using the collected information, parent nodes can recalculate and reassign the tasks. There is a tradeoff between the reassignment quality and overhead. For example, if every node reports its current status to its parent node, the parent nodes report further to their parent nodes, and eventually, the root node has all the information, a best quality reassignment can be calculated and distributed to every node. However, in the situation, communication overhead is large. Therefore, we introduce system parameter \( L \) to balance the assignment quality and overhead. \( L \) indicates which roads are involved in the task reallocation. Only roads at level \( L \) or higher are involved in the reallocation. For example, in Fig. 7(a), \( L = 2 \). Disseminators on roads \( D \) and \( E \) at level 2 and roads \( B \) and \( C \) at level 3 are involved in workload reallocation, whereas road \( A \) at level 1 is not involved.

Another system parameter \( t_{re} \) is also introduced, which is the time to start the reassignment procedure. At \( t_{re} \), all roads with a level higher than \( L \) reports to their parent nodes the current situation of their branch. After the nodes at \( L \) collect the necessary information, they redo the calculation according to (6) based on updated information and then distribute the new workload to the children nodes. The setting of \( t_{re} \) cannot be too small or too large. If it is too small, the dissemination has just started, and reassignment may not be necessary. If it is too large, many roads may have been idle for a long time, and it is a waste not to utilize their processing power.

According to the heuristics in Section III-B, the disseminator role on each road changes during the dissemination. When higher level disseminators report their workloads to lower level disseminators, the current disseminator of the corresponding road may not be the original delegator who delegates the workload. The higher level disseminators do not know the ID of the current disseminator on that road and, thus, cannot reach it. To solve this issue, when reassignment begins, we first let all the disseminators stop delegations. Then, on each road involved in the reassignment, the disseminator periodically sends its ID to intersections that intersect its higher level roads, so that vehicles on these intersections can cache the disseminator’s ID. If the disseminator leaves the current road and delegates workload to another vehicle, that vehicle will become a new disseminator. It sends its own ID and continues to wait for the report. When reporting workloads, higher level disseminators route the message to intersections of the corresponding lower level road and broadcast there. Upon receiving the message, vehicles on lower level road will check the cached disseminator’s ID and forward the message to the disseminator.

2) Example: A detailed example with \( L = 2 \) is shown in Figs. 7 and 8. At time \( t_{re} \), disseminators on roads \( B, D, C, \) and \( E \) start the workload refinement. They temporarily stop relaying their disseminator roles. Disseminator \( v_i \) on road \( D \) sends its ID to the intersections that intersect roads \( B \) and \( C \). Since road \( E \) does not have any child node, no one reports to the disseminator on road \( E \) although it is in level 2. Disseminators on roads \( B \) and \( C \) report the remaining workload, shown as the red line in Fig. 7(a), to \( v_i \) on road \( D \). Messages from roads \( B \) and \( C \) will be routed and broadcasted at intersections \( I_{BD} \) and \( I_{CD} \), respectively. Vehicles (e.g., \( v_j \) and \( v_k \) in \( I_{BD} \)) will receive the message and forward it to \( v_i \), as shown in Fig. 8. Upon receiving workload reports, \( v_i \) computes a new workload assignment according to the report and distributes back to disseminators on roads \( B \) and \( C \). Disseminators on roads \( B, D, \) and \( C \) then restart the dissemination according to the new workload, as shown in Fig. 7(b).

### D. Protocols

This section summarizes the protocols running on disseminators and ordinary vehicles, respectively.

1) Protocol of a Disseminator: The job of a disseminator is to disseminate data and delegate workload to other roads. When a disseminator gets the workload for its current road and the delegation tree, it shares half of its assigned workload to a backward vehicle based on the optimization described in Section III-B. Then, it sets a timer to periodically disseminate messages and counts the number of repliers. If the current road is not a leaf node of the delegation tree, the disseminator delegates the job according to the aforementioned procedure in Section III-A. In the case when the disseminator fails to find a vehicle on the intersected road to delegate the task even after arriving at the intersection, it recalculates the shortest-path tree rooted at its current position and the new workload for each road in the tree. After the calculation, it disseminates the data and delegates the workload based on the new tree. Otherwise, if the road is a leaf node, the disseminator simply takes over the task and disseminates by itself. After finishing the workload delegation, the disseminator relays...
the disseminator role forward to speed up the dissemination. When it is time for workload reassignment, the disseminator checks the level of its road and launches the dynamic reassignment. The dissemination procedure on this road continues until the required number is reached. The detailed protocol is illustrated in Algorithm 1.

### Algorithm 1 Protocol of disseminator \( v_i \) on road \( RD_i \)

**Notations:**
- \( T_i \): delegation tree rooted at road \( RD_i \)
- \( m_i \): workload of road \( RD_i \)
- \( t_r \): time for dynamic workload refinement
- \( l_i \): level of \( RD_i \) in the delegation tree

**Upon receiving** \( \langle t_r, m_i, T_i \rangle \)
1. Shares \( m_i/2 \) backward according to Section III-B.
2. Periodically disseminates messages and collect receipts.
3. if \( RD_i \) is not a leaf node in \( T_i \) then
4. Delegate workload to other roads on \( T_i \) according to Section III-A.
5. if Fail to find a workload receiver after passing the intersection then
6. Recalculate the workload and delegation tree.
7. Disseminate and delegate according to the new workload and tree.
8. end if
9. else
10. Take over the task and disseminate on \( RD_i \).
11. end if

**Upon finishing the workload delegation**
12. Relay the disseminator role forward according to Section III-B.

**Upon time = \( t_r \)**
13. if \( l_i \leq L \) then
14. Dynamic workload refinement according to Section III-C.
15. end if

2) **Protocol of an Ordinary Vehicle:** The job of an ordinary vehicle is simple. Upon receiving the disseminated data, it first checks whether it has received the data before or not. If not, it replies with a confirmation. An ordinary vehicle becomes a disseminator if it is assigned workload in one of the aforementioned cases.

E. **Enhancement With Workload Backup**

In a real situation, vehicles stop and leave the network. To deal with the situation that a disseminator leaves the network, we propose a backup strategy where a backup vehicle takes over the job of a disseminator who leaves.

1) **Backup Vehicle Selection:** Consider two vehicles on a road, if they have been almost relatively static for a while, it is highly possible that their relative positions will not change much in the short future. Based on this consideration, we let the disseminator keep track of its neighbors. Each neighbor is associated with a counter that indicates for how long the vehicle has been its neighbor. The backup vehicle is one of the disseminator’s neighbors with the highest counter value. When the selection is done, the disseminator informs the backup vehicle.

Every time, after accepting workload assignment, the disseminator also informs the backup vehicle its dissemination workload and a timer value. Since the backup vehicle is the neighbor of the disseminator, it receives the disseminated messages from the disseminator periodically. Once the backup vehicle does not receive the message at the expected time, it sets a timer and sends a message to that disseminator to check whether it is still alive.

2) **Working Procedure of Backup:** Upon receiving a message from its backup vehicle for an alive check, the disseminator replies. Then, the disseminator indicates to the backup vehicle to switch to an ordinary vehicle. After that, it reselects a new backup vehicle closer to it according to the selection procedure and informs it of the workload and remaining timer value.

If the disseminator leaves the network, the backup vehicle cannot receive the reply, and it determines that its disseminator has gone. The backup vehicle becomes a disseminator with the last updated remaining workload of its disseminator.

Note that, due to a hidden terminal issue, it is possible that the backup vehicle cannot overhear some broadcasts from the disseminator even if it is in the disseminator’s communication range. This particularly happens when our scheme is deployed in the network together with other communication schemes, where packets are sent out by many applications at the same time and cause collision. In this case, we can apply existing mechanisms, such as VeMac [28], to mitigate the issue.

3) **Redundant Backup Cancelation:** As a disseminator and its backup vehicle may have different speeds and paths, the backup vehicle may not be close to the disseminator after a while. In this case, the backup vehicle will become a disseminator even if the disseminator is still alive. To eliminate redundant disseminators, we also design a backup cancelation strategy. Consider that \( v_i \) is a backup vehicle for \( v_s \). It becomes a disseminator at time \( t_i \) when it believes that \( v_s \) has left the network. If a message receiver \( v_j \) receives data from \( v_i \) and has received the data disseminated from \( v_s \) at a time later than \( t_i \), \( v_j \) replies to \( v_i \), notifying it to switch back to an ordinary vehicle considering that \( v_s \) is still alive at \( t_i \).

If a disseminator finishes its task and switches to an ordinary vehicle, it also sends messages to its backup vehicle and let it switch to an ordinary vehicle. If the backup vehicle turns to other roads, it switches to an ordinary vehicle and sends a message to inform its disseminator.

IV. **Performance Evaluation**

A. **Evaluation Methodology**

DOVE is evaluated through simulation implemented in the NS-2 simulator. The evaluation is under both normal traffic density and sparse traffic density. The objectives of the evaluation are fourfold: 1) testing the effectiveness of DOVE in reaching a desired number of receivers; 2) evaluating the efficiency of DOVE by comparing it with other protocols; 3) studying the...
performance of DOVE under different dissemination intervals (i.e., the interval between each two broadcasts); and 4) evaluating the effectiveness of several scheme optimizations, including dissemination relay in Section III-B and workload refinement in Section III-C. We employ three metrics to do the evaluation. We choose the total number of messages transmitted as a measure of overhead and choose the dissemination delay and the actual number of vehicles reached as measures of effectiveness.

Since we are the first to address the problem of disseminating data to a desired number of receivers, there are not many comparable protocols. SSD [13] disseminates e-Ads using a fixed number of forwarding levels, and it is our closest related work. In addition, flooding is considered as a dissemination protocol with the least dissemination delay, whereas one-hop broadcast is considered to have the least communication cost. We choose one-hop broadcasting as a baseline protocol with the lowest overhead to compare and flooding as a baseline protocol of the fastest dissemination to compare. Thus, we compare the performance of DOVE with the following three protocols: 1) flooding; 2) one-hop broadcasting; and 3) SSD. The original version of SSD only specifies the number of forwarding levels and does not have any restriction on the total number of receivers. To have a fair comparison, we extend SSD by deleting the security messages to save overhead and evenly allocating workload for each car in one forwarding level. In addition, in our simulation, SSD and flooding stop when the required number of receivers is reached, assuming there is a “God’s hand” controlling the stop of two protocols. For SSD, we evaluate both SSD:2-Level and SSD:3-Level.

To evaluate the impact of the dissemination interval, we vary it from 5 to 20 s. To test the performance of our workload backup heuristics, we simulate the scenario where vehicles join and leave the network. Each vehicle has a lifetime following the Gaussian distribution. Once a vehicle reaches its lifetime, its memory will be flushed to simulate the case where it quits the network. Then, it reenters the area as a new vehicle. We test the impact of the reassignment level and the reassignment time. Since a normal area map, e.g., Fig. 9, usually has no more than four levels in the shortest-path tree, when evaluating the impact of the workload reassignment level, we only configure DOVE with reassignment levels 1, 2, and 3. We evaluate the effectiveness of the proposed optimizations, including workload reassignment, and relaying the disseminator role forward and sharing it backward.

In our simulation, the map is a 2000 m × 2000 m area centered at (Latitude: 40.64548, Longitude: −73.942795) in Brooklyn, NY, USA, extracted from the Topologically Integrated Geographic Encoding and Referencing (TIGER)/Line database of the U.S. Census Bureau [29]. Fig. 9 shows the map we use. Based on the guidelines provided in [30], for the normal traffic scenario, we deploy 600 vehicles in the area to generate median-light density of traffic, and our goal is to reach 400 of them, whereas for the sparse traffic scenario, we deploy 300 vehicles and aim to reach 200 of them. For the moving trace of vehicles, we employ the open-source microscopic space-continuous time-discrete vehicular traffic generator package SUMO [31] to generate the movements of vehicle nodes. SUMO uses a collision-free car-following model to determine the speeds and positions of the vehicles. After integrating the SUMO trace into the map from TIGER, we discard the first 2000 s to obtain more accurate node movements. The output from SUMO is converted into input files for the movement of nodes in the NS-2 simulator. We configure the simulation according to the WAVE protocol [32]. We use a two-way ground model to model the vehicle communication channel. In our setting, two vehicles on different streets cannot communicate directly because we assume that there are obstacles in between. Unless evaluating the dissemination interval in Section IV-C, we set the dissemination interval to be 10 s. Other system parameters are listed in Table I.

### B. Comparing With Other Schemes

Figs. 10 and 11 show the dissemination delay and overhead of all the aforementioned schemes, in the scenario with normal traffic density. Shown in Fig. 10, the delay of our scheme is 580 s, which is almost the same as that of the flooding scheme, whereas SSD’s delay is more than 2000 s. In terms of communication overhead, as shown in Fig. 11, the cost of our scheme is very close to that of the one-hop broadcasting scheme, which is very low. Note that although the overhead of one-hop broadcast is low, it cannot reach 400 receivers even after 5000 s. The communication overhead of SSD is almost twice as much as ours, and the flooding scheme is more than four times larger than ours. The comparison shows that our scheme is effective in disseminating data to a desired number of receivers and is efficient in achieving it.
C. Impact of Dissemination Interval

This section evaluates the impact of the dissemination interval on the performance of DOVE. Fig. 14(a) shows dissemination delay under different dissemination intervals in a normal traffic scenario. As expected, when the dissemination interval is smaller than 10 s, the dissemination delay is less than 500 s. When the dissemination interval increases, the dissemination delay increases from 253 s to almost 1200 s. Fig. 14(b) shows the total number of transmitted messages in a normal traffic scenario, which includes messages of broadcasting data, receiver acknowledgment, and workload delegation, under different dissemination intervals. The number of transmitted messages sharply decreases with the increase in the dissemination interval. For example, when the interval is 5 s, about 2600 messages are transmitted for disseminating 400 copies; however, when the interval is 20 s, the total number of transmitted messages reduces to about 1250.

In Fig. 14(a) and (b), we can see that a higher interval means lower overhead but longer delay. Hence, in Fig. 14(c), we show the ratio of the number of transmitted messages to the number of vehicles reached in a normal traffic scenario. The metric shows the average message overhead for reaching each receiver. In this figure, we can see that the ratio is very high at the beginning of dissemination, because at that time, the disseminator has not yet delegated workload to other vehicles, which results in “temporarily low” efficiency. As the dissemination goes on, disseminators from other roads receive the workload and start dissemination, and thus, the ratio decreases. When the dissemination interval is 5 s, the ratio is higher than 6. As the interval increases to 10 s, the ratio decreases to about 4 and keeps decreasing slowly to 3 when the interval increases to 20 s. From that we can see, when the dissemination interval is larger than a threshold (e.g., 15 s), the costs are similar, because at any time the disseminator could only reach the vehicles within its range. The number of receivers in each dissemination is limited due to the limited communication range (about 200 m). Hence, the cost has a lower bound in such situations.

Fig. 15(a)–(c) shows the impact of the dissemination interval on the performance, cost, and the ratio of cost and performance of DOVE in the sparse traffic scenario, respectively. All the figures in the sparse traffic case show similar trends as that in the normal traffic case. To reach the same number of vehicles, the time DOVE spends in the sparse traffic scenario is about 1.2 times that spent in the normal traffic scenario, and the message it sends in sparse traffic is about 1.28 times as that in normal traffic. This is because when the traffic density gets lower, the number of vehicles that a disseminator can reach after each broadcast decreases. When dissemination starts for a while, the ratio of the number of transmitted messages to the number of
Fig. 14. Impact of the dissemination interval in normal traffic scenario.

Fig. 15. Impact of the dissemination interval in sparse traffic scenario.

Fig. 16. Comparisons between DOVE with and without workload backup in normal traffic scenario. (a) Dissemination interval = 10 s. (b) Mean value of vehicles’ lifetime = 500 s. (c) Vehicles’ lifetime = infinity.

vehicles reached keeps at about 5 to 7, which is a bit higher than that in the normal traffic case.

D. Evaluating the Backup Scheme

Considering that the simulation map is 2000 m × 2000 m and that the average vehicle speed is 8–15 m/s (18–35 mi/h), normally, it takes a vehicle 200–800 s to pass from one side to another. We vary the mean value of lifetime from 200 to 800 s and set the dissemination interval to be 10 s.

Fig. 16(a) shows the total number of the reached receivers in the disseminations by DOVE with and without backup in the normal traffic scenario. In this figure, we can see that if the mean lifetime is 200 s, without the workload backup, only 270 receivers receive the data copies due to the disseminators frequently leaving. However, the 400 receivers are guaranteed by the enhanced DOVE with workload backup, which reaches 407 vehicles in that case. When the length of lifetime increases, the total number of disseminated data by DOVE without backup grows closer to 400, because more disseminators can finish their workload before leaving. The number of receivers reaches an exact 400 if the lifetime is larger than 600 s, because in this case, few disseminators leave the network before finishing their dissemination task.

We evaluate the impact of the dissemination interval on the number of reached receivers for DOVE with and without backup in the normal traffic scenario in Fig. 16(b). The mean value of vehicles’ lifetime is set to be 500 s. In Fig. 16(b), we
can see that the number of reached receivers decreases from 385 to 234 when the dissemination interval increases from 5 to 20 s. When the dissemination interval is large, disseminators need a long time to finish the dissemination task, which may be longer than their lifetime. Hence, in this case, for DOVE without backup, a lot of workload is lost. However, for DOVE with backup, the total number of reached receivers almost does not change when the dissemination interval varies.

To compare the overhead of DOVE with and without backup, we set the vehicle’s lifetime to be infinite to have a fair comparison. We vary the dissemination interval from 5 to 20 s. In this figure, we can see that when the dissemination interval increases from 5 to 10 s, the number of messages decreases from 2000 to 1200 for DOVE without backup and from 2200 to 1500 for DOVE with backup. When the interval is greater than 15 s, the message overhead almost does not change, and messages sent by DOVE with backup are always about 18% more than that sent by DOVE without backup.

Fig. 17 shows the effectiveness of the backup scheme in the sparse traffic scenario. In the figure, we can see that the performance and cost of the backup scheme in the sparse traffic scenario is similar to that in the normal traffic scenario. For example, when the mean value of the vehicle’s lifetime is 250 s, as shown in Fig. 17(b), the DOVE with backup scheme reaches 191–205 vehicles, which works with only less than 5% inaccuracy, whereas the DOVE without backup scheme may reach 40% less than the required number.

E. Impact of the Workload Reassignment

We fix the dissemination interval at 10 s. The dissemination time and message overhead of DOVE under different workload reassignment levels are shown in Fig. 18. In the figure, we can see that the dissemination time greatly decreases from level 3 to level 2 and slightly decreases from level 2 to level 1. Compared with the scheme without workload reassignment, the workload assignment significantly reduces the dissemination time. The message overhead sharply decreases from level 1 to level 2 and mildly decreases from level 2 to level 3. Thus, for the dissemination map, as shown in Fig. 9, if we set the reassignment level to 2, the workload reassignment scheme can significantly reduce the disseminate time while only introduce a little message overhead. For any given map, we can first run such simulation on the map to find the best reassignment level and then use this as a guideline to deploy our application to the area.

F. Comparing Different Version of our Scheme

We compare the dissemination delay and message overhead of DOVE with and without efficiency optimizations in the normal traffic scenario in Figs. 19 and 20, respectively. In Fig. 19, we can see that the dissemination delay of DOVE is
V. Related Work

To the best of our knowledge, we are the first to address data dissemination to a desired number of receivers. There are not many comparable related works. SSD [13] disseminates advertisements following a tree structure of a fixed level and is the closest research to us. By restricting the level of dissemination, SSD can guarantee that the number of receivers is within a range and, thus, aims to satisfy the budget constraint; however, it can only control the number in a very coarse level.

Protocols for data dissemination in VANET can be broadly classified into two categories: One requires data to be broadcasted in a local or remote area of interest and the other category does not. In the first category, geocast mechanisms [18], [33] deliver data to an area of interest, where location-aware nodes broadcast and selectively rebroadcast packets based on local-related rules. Gohari and Rodoplu [34] study the features of the interested area and aim to discover reliable geographic routes in the most statistically reliable region. To improve forwarding efficiency, a set of heuristics is proposed in [17] to suppress redundant transmissions in the forwarding zone. The forwarding link may often suffer from disconnection [35] in urban areas with obstacles and high vehicle mobility. To address the problem, Jiang and Zhu [36] select vehicles with high coverage capability to forward packets so as to improve packet delivery ratio. To persistently disseminate data in a specific area, abiding-geocast [37] lets vehicles relay the dissemination in the network boundary. Relay strategies are well studied in [38] and [39] to select vehicles in the interested region to continue the dissemination. Because of the highly dynamic network topology in VANETs, opportunistic dissemination is employed in [40], which generates message replicas dynamically to improve link reliability. To improve dissemination persistency, Leonidatis et al. [39] utilize vehicles’ driving route to select relay vehicles.

In the second category, flooding is the simplest approach to broadcast messages but has huge message overhead. Many techniques [7]–[10], [41] have been studied to improve the flooding mechanism to improve broadcast efficacy. To improve the reliability, efficiency, and effectiveness of broadcast-based schemes, many on-road characteristics, such as traffic density, traffic flow, and road topology, have been leveraged [42]–[44]. Neighbor information is used in [4] to determine whether a vehicle needs to forward or continue to carry the cached message. Sommer et al. [45] employ adaptive beaconing messages to dynamically adjust the data rate to fit the channel quality and message priority in the network.

The aforementioned schemes together improve reliability, efficiency, and persistency in data dissemination. However, none of them considers controlling the number of message receivers. Without a delegated workload management system to dynamically allocate, trace, and backup the workloads, these mechanisms cannot be applied to achieve the goals of this paper.

VI. Conclusion

In this paper, we have proposed DOVE, which is an efficient data dissemination protocol for disseminating data to a desired number of receivers in a VANET. Inspired by processor
scheduling, we utilized the road layout and traffic information of roads as two key factors to transform the given problem to the processor scheduling problem, subject to minimizing the dissemination delay. We have proposed a theoretical optimal algorithm to solve the problem in the ideal case. In dealing with real issues, such as vehicles leaving the network, an efficient distributed protocol has been presented in this paper. To improve on-road dissemination efficiency, we have provided three heuristics that significantly reduce the total dissemination relay. The simulation result shows that our protocol is both efficient and effective.

ACKNOWLEDGMENT

The authors would like to thank Dr. Y. Zhang from the Chinese Science Academy for the stimulating discussions.

REFERENCES


Tan Yan received the B.E. degree in information science and technology from Southeast University, Nanjing, China, and the M.E. degree from New Jersey Institute of Technology, Newark, NJ, USA, where he is currently working toward the Ph.D. degree with the Department of Computer Science under the supervision of Dr. G. Wang.

Wensheng Zhang (M’09) received the B.S. degree from Tongji University, Shanghai, China; the M.S. degree from the Chinese Academy of Sciences, Beijing, China; and the Ph.D. degree from The Pennsylvania State University, State College, PA, USA, all in computer science and engineering.

Since 2005, he has been a faculty member with the Department of Computer Science, Iowa State University, Ames, IA, USA, where he is currently an Associate Professor.

Dr. Zhang is a member of the ACM.

Guiling Wang received the B.S. degree in software from Nankai University, Tianjin, China, and the Ph.D. degree in computer science and engineering with a minor in statistics from The Pennsylvania State University, State College, PA, USA, in 2006.

She joined the New Jersey Institute of Technology, Newark, NJ, USA, in the fall of 2006 and was promoted to Associate Professor with tenure in June 2011.