# Energy-Efficient Roadside Unit Scheduling for Maintaining Connectivity in Vehicle Ad-hoc Network

Feng Zou, Jiaofei Zhong
Dept. of Computer Science
University of Texas at Dallas
phenixflying@gmail.com,
fayzhong@utdallas.edu

Weili Wu, Ding-Zhu Du
Dept. of Computer Science
University of Texas at Dallas
{weiliwu, dzdu}@utdallas.edu

Junghoon Lee
Dept. of Computer Science
and Statistics
Jeju National University
jhlee@jejunu.ac.kr

## ABSTRACT

Vehicle Ad-hoc Network (VANET) is a type of mobile ad-hoc networks with highly dynamic topology. To address its frequent network partition issue, recently a special kind of infrastructure called the Roadside Unit (RSU) is proposed to be deployed along the road to improve the VANET connectivity. In this paper, we study the energy saving problem in RSU scheduling. Given a set of deployed RSUs, our objective is to find the optimal schedule of turning them on and off within a given time period so that the overall energy consumption of RSUs in the system is minimized while the network connectivity is still maintained. We divide this problem into two subproblems called the *snapshot scheduling* problem and the *snapshot selection* problem. The *snapshot scheduling* problem decides the minimum number of active RSUs needed for a snapshot of the VANET at a given time point, while the *snapshot selection* problem decides a sequence of time points on which the snapshot must be updated. By addressing these two subproblems, we present a complete solution for our energy-efficient RSU scheduling. We present our theoretical analysis and experimental results to show that our algorithms can achieve a significant energy saving while still maintaining the VANET connectivity.

## Keywords

Vehicle ad-hoc network, Energy-efficient, RSU scheduling, Optimization

## 1. INTRODUCTION

As an emerging new type of mobile ad-hoc network, Vehicle Ad-hoc Network (VANET) faces one crucial challenge: its highly dynamic and partitioned characteristic severely limits the speed and accuracy of data dissemination inside the network. Even though each vehicle can exchange information with others when within their communication ranges, in order to successfully disseminate necessary information all over the network, every pair of vehicles within the VANET has to be connected. However, in real-world scenarios, a group of vehicles may need to wait for a long time or even cannot communicate with other groups of vehicles forever in the VANET. This obviously affects the efficiency of information exchange. The situation becomes even worse in large-scale VANET, since the network will have a much higher possibility to be disconnected. This highly partitioned feature raises problems in many other research areas of VANET, such as maintaining the data freshness and monitoring the real-time traffic. The improved network connectivity will also reduce the difficulties of these problems.

Considering this underlying highly-partitioned characteristic of VANET, a special kind of infrastructure called the Roadside Unit(RSU) is introduced recently. RSUs are static devices that can be deployed independently on roadside or installed together with traffic lights, and they are capable of large area sensing and communication. With RSUs such as 802.11 access points, vehicles can either access data stored in them or upload its own data. Previous research work has shown that introducing these RSUs can alleviate the network partition problem, therefore greatly improve the information dissemination and data aggregation in VANET [12]. However, since most of RSUs are battery-powered, how to schedule them to achieve minimum energy consumption while still maintaining the network connectivity is still a challenging problem and has not received sufficient attentions.

In this paper, we assume that all the RSUs are already deployed in the network and a full coverage of the network can be achieved by setting all of them to be active. We formalize the energy saving problem in RSU scheduling as an optimization problem. Our objective is to find the optimal schedule of turning on and off the deployed RSUs within a given time period so that the total energy consumption in that period is minimized while the connectivity of the VANET is still maintained. We divide this optimization problem into two subproblems called the *snapshot scheduling* problem and the *snapshot selection* problem. The *snapshot scheduling* problem is reduced to the *Steiner Tree with Minimum Number of Steiner Points* (STP-MSP) problem and it decides the minimum number of active RSUs needed for a snapshot of the VANET at a given time point; The *snapshot selection* problem decides the sequence of time points on which the snapshot must be updated thus to maintain the network connectivity. The combination of these two subproblems presents a complete framework for our energy-efficient RSU scheduling.

The rest of this paper is organized as follows. Section 2 briefly summarizes the related works, and section 3 describes


*ICUIMC* '11, February 21-23, 2011, Seoul, Korea

the details of our optimization problem from the perspective of graph theory. We explain our models and assumptions in Section 4. Section 5 presents our complete solution and gives the theoretical analysis of the two subproblems. In section 6, we evaluate the performance of our algorithms through experiments in the downtown area in Dallas. We conclude the paper in section 7 and discuss the future work.

# 2. RELATED WORK

Research works related to the RSUs in the literature can be generally classified into the following three categories: the vehicle-RSU data access scheduling, the communication setup between vehicle and RSU, and the RSU placement. In this section, we describe several previous works related to our work and give a brief review of the *Steiner tree with Minimum Number of Steiner Point* (STP-MSP) problem which is the theoretical basis of our snapshot scheduling problem.

## 2.1 RSU Scheduling

The vehicle-RSU data access scheduling problem is to design efficient scheduling algorithms for the servers on RSUs to handle data access requests from moving vehicles in the network. Most of existing schemes do not consider either the time constraints or the data quality. Zhang et al. [17] present a scheduling scheme called $D * S$ considering both of these factors. Moreover, $D * S$ is optimized and a new scheme called $D * S/N$ is proposed. In their paper, they also study the impact of data staleness and present a two-step scheduling scheme to provide a balance between serving download and update requests.

Lots of effort has also been devoted in the study of vehicle-to-RSU communication. Bohm et al. [**?**] target at the handover and connection setup between a vehicle and an RSU, in order to reduce the delay of waiting for access to the wireless communication channel. In their system, RSU can produce a communication schedule for all vehicles in its transmission range and poll these vehicles for data. They show that with limited overhead, this mechanism still allows MAC protocols to support various safe-critical applications in a densely highway scenario.

Considering the highly partitioned nature of VANET, researchers also study the identification of RSU positions. In their paper, Lochert et al.[12] present their solution of RSU placement for a VANET traffic information system using a genetic algorithm. In order to cope with the highly partitioned nature of a VANET in an early deployment stage, they identify good spots for RSUs from a set of possible positions that are initially given. However, their choice for the best set of RSUs highly depends on the traffic situation simulated in the experiments. As different traffic situations might require different sets of best RSU spots, whether the selected set of RSUs could perform well in different traffic situations is not guaranteed in their paper. Moreover, even though the information dissemination in VANET could be improved by deploying a set of RSUs at the selected spots, the network partition problem is still unavoidable. Considering from this perspective, the genetic algorithm only helps alleviate the network partition problem but prevent it.

Different from all aforementioned problems, in this paper we investigate the scheduling of RSUs from the connectivity and energy saving perspectives. We assume that a set of RSUs are deployed initially providing full coverage of the designated area. Our objective is to design an efficient scheduling scheme for all the deployed RSUs so that at each time point within a given time period, the connectivity of all the vehicles in the network is maintained and the energy consumption during this period is minimized. To the best of our knowledge, our work is the first to study this problem in the literature.

## 2.2 STP-MSP Problem

As we have mentioned, the STP-MSP problem is highly related to one of our subproblems, the *snapshot scheduling* problem. Therefore, we briefly describe this problem as part of the related work here. Given a graph $G = (V, E)$ with a weight function $w$ on $E$ and a subset $S$, the *Steiner tree problem* (STP) is to find a minimum weight subgraph of $G$ interconnecting all nodes in $S$. We call the set $S$ as *terminal set*. For any Steiner tree $T$ interconnecting $S$ and a node $u \in V(T)$, we call $u$ as a *terminal node* if $u \in S$. Otherwise, we call it as a *Steiner node*. The Steiner tree problem, which is of great interest nowadays, is a classic connectivity problem in networks. This problem is known to be NP-hard in most metrics[8]. Lots of effort has been devoted to study the approximation algorithms for this problem [1, 9, **?**, 14, **?**] and have successfully achieved constant ratios, the best known result among which is $\rho = 1 + \frac{\ln 3}{2} \approx 1.55$ [14].

The Steiner tree problem with minimum number of Steiner points, denoted by STP-MSP for short, is a variation of Steiner tree problem. It is defined as follows: Given a set $S$ of $n$ terminals, STP-MSP asks for a tree $T$ interconnecting a superset of $S$ such that the number of Steiner points in $T$ is minimized. There are two versions of STP-MSP problem. One is in graphs and another is in plane (e.g. Euclidean plane). The difference between them is that in graphs, all the vertices contained in $T$ should belong to vertex set $V$ of $G$. While in the plane, there is no restriction for that. Steiner points could be any point in the plane. Existing approximation algorithms for the STP-MSP problem [5, 7, 11] all focus on STP-MSP in Euclidean plane and have achieved constant approximation ratios. While few algorithms are designed exactly for the STP-MSP problem in graphs as far as we know.

# 3. PROBLEM DESCRIPTION

The optimization problem we study in this paper can be described as following: Given a time period $[0, T]$ and a set of deployed RSUs $R$, which are deployed along the roads and cover all the interested area, we look for a schedule of mode switch (on/off) for all the RSUs in $R$ during the time period $[0, T]$, so that the vehicles in the network are connected through RSUs at all time in the period and the energy consumption of RSUs during this time period is minimized as well.

We consider this problem from the perspective of graph theory. To begin with, we introduce a *Virtual Control Node(VCN)* into the VANET, which is considered connecting to all the RSUs through wires. If taking the vehicles in the network together with all RSUs and the VCN as the vertex set of a graph, we could derive a graph called *temporal graph* for the VANET. The *temporal graph* is a kind of graph in which each edge has its associated temporal feature. This temporal feature is described using *temporal vector* and *temporal duple*.

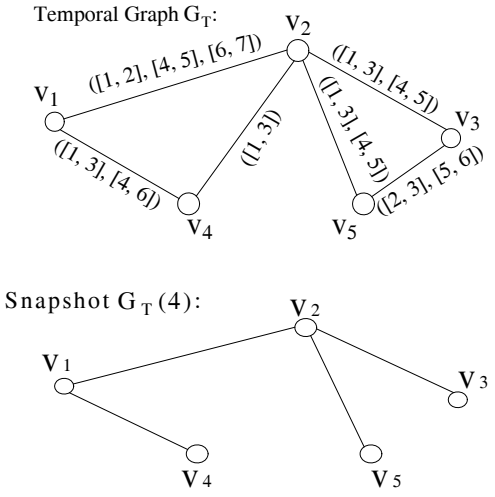


Figure 1: An example of temporal graph and snapshot

*Definition 1.* **Temporal Vector**: A *temporal vector* $s$ is consisted of a set of time intervals $s_i$. Each time interval represents a close time period (e.g., $[t_{i1}, t_{i2}]$) and none of the time-intervals in the same *temporal vector* overlaps with each other. Note that the temporal vector $s$ could be an empty set.

*Definition 2.* **Temporal Duple**: A *temporal duple* $D$ is a duple including one temporal vector. We will use the notation $(u, v) : s$, where $u$ and $v$ represent two distinct objects and $s$ is the temporal vector.

Now, based on the definitions of *temporal vector* and *temporal duple*, we can define the *Temporal Graph* as follows.

*Definition 3.* **Temporal Graph**: A *temporal graph* $G_T$ is a set of temporal duples $\{D_1, D_2, \ldots, D_k\}$. Vertex pairs that are not included in this set are considered to have no edges interconnecting them at any time.

In the temporal graph $G_T$ of the VANET, there exists one edge between two vehicles in the graph at time $t$ if they are within each other's communication range at that time point. This edge is denoted as an *effective edge* at time $t$ otherwise a *noneffective edge*. And there exists one edge connecting a vehicle and an RSU at time $t$ if the vehicle could directly communicate with this RSU. It's obvious that all the edges between RSUs and the VCN have the same temporal vector $[0, T]$.

Further, for each temporal graph $G_T$, we define its *snapshot* at time $t$ as follows.

*Definition 4.* **Snapshot**: A *snapshot* $G_T(t)$ of a temporal graph $G_T$ at time $t$ contains the same vertices as $G_T$, but only effective edges at time $t$ in $G_T$.

Figure 1 presents an illustrative example of the temporal graph and its associated snapshot at time $t = 4$. At time 4, edges ($v_1$,$v_2$),($v_1$,$v_4$),($v_2$,$v_5$) and ($v_2$,$v_3$) are effective edges.

Considering the set containing vertices representing vehicles and the vertex representing VCN as the terminal set $S$ in this temporal graph $G_T$, our optimization problem could be further described as to find the minimum number of RSUs in $R$ to be activated at every time point $t$ in the time period $[0, T]$ such that every vertex in the set $S$ is connected and the energy consumption of the RSUs is minimized. In order to solve this problem, we divide it into two subproblems.

The first subproblem is called *Snapshot Scheduling* problem, which is to minimize the number of active RSUs needed for a snapshot $G_T(t)$, so that every vertex in the set $S$ is connected and the energy consumption of RSUs at time $t$ is minimized. Here we assume that there is no energy consumption during the RSU mode switch. The *Snapshot Scheduling* problem is highly related to the *STP-MSP* problem.

As it is impractical and expensive to adjust the RSUs' mode at every time point $t$, our second subproblem is to decide the sequence of necessary time points on which the snapshot must be updated thus to maintain the network connectivity. We call it *Snapshot Selection* problem.

In the following, we will first describe our system model in Section 4 and present our solutions for the *Snapshot Scheduling* problem and *Snapshot Selection* problem in Section 5.

# 4. SYSTEM MODELS AND ASSUMPTIONS

Before introduce the main idea of our energy-efficient RSU scheduling, in this section, we summarize the assumptions and describe the models we applied in this paper. First, we study the problem in a discrete time system. We assume that all RSUs are powered through batteries and they are homogeneous. All RSUs have the same sensing and communication range and they can communicate with each other as well as the VCN through a backbone network [12]. Each RSU can switch between two different modes: active mode and sleep mode. For simplicity, We assume that there is no energy consumption when the RSU is in the sleep mode and during the mode switch. All possible wireless interferences are also ignored.

## 4.1 Roadside Units Deployment Model

We assume that a certain amount of RSUs has already been deployed along the roads, and the area that we are interested can be fully covered when all these RSUs are active. Each vehicle on the road can communicate with at least one RSU at any time. They could download information from RSUs and also upload information to them. This is quite different from the model used in [12], where the placement of RSUs does not guarantee the full coverage of the area.

Figure 2 gives an illustration of our model taking one road segment as an example. Assume that the communication range of every RSU equals to $r_c$. If the length of this road segment is $L$, then we can deploy the RSUs along the road at points $\{r_c, 3 \cdot r_c, \ldots, k \cdot r_c, \ldots\}$, where $k = 2 \cdot j + 1 (0 \leq j \leq \frac{L}{2 \cdot r_c})$. If the length of the road is not exactly multiples of the value $2 \cdot r_c$, then an extra RSU will be deployed at the ending point of the road. Compared with the deployment model in [12], this grid deployment of RSUs could guarantee the coverage of all the roads as well as vehicles on the roads in the target area. Besides, comparing with those models which deploy RSUs irregularly in the area (e.g. places away from roads), such deployment also reduce the total management cost.

## 4.2 Vehicle Mobility Model

The construction of the temporal graph of the VANET is an essential step in our scheme, and the difficulty of its

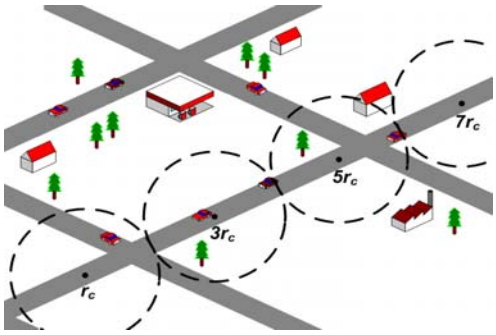


Figure 2: An example of the RSU deployment

construction highly depends on the mobility model we adopt in the system. The more complicated the mobility model is, the more difficult the construction will be.

There are many mobility models that are specifically designed for ad-hoc network like VANET. A brief taxonomy is given in [2], and [4] provides a detailed survey and presents simulation-based comparisons among different mobility models.

Random Waypoint model [3, 10, 15, 6] is one of the most widely used mobility models for ad-hoc networks. It has been proven to be an effective approximation for simulating the realistic motion of vehicles in VANET [**?**, 16, 13]. [16] introduces a variation of the Random Waypoint model and constrains the motion of each vehicle to be along the roads. It calculates the route from source to a random destination using Dijkstra's single source shortest path algorithm, travels along the path until reaches its destination. Once a vehicle reaches its destination, it chooses another random destination and repeats this process. [16] shows through their experimental results that the performance of this model is comparable with the classic Random Waypoint model and is more practical in real-world scenarios as well.

We adopt this mobility model in our paper. We let each vehicle start from a random source $V_s$ to a random destination $V_d$. The route that it follows is calculated before it starts traveling and is based on its knowledge of the area. We assume that every vehicle in our system travels with its chosen speed $v$ all the way until it reaches the destination. After that, it randomly chooses another destination and repeats this process again. It is easy to observe that under this mobility model, as long as each vehicle reports its speed and destination to RSUs that it encounters on the roads, RSUs can easily predicate its movement.

# 5. ENERGY-EFFICIENT ROADSIDE UNIT SCHEDULING

In this section, we shall describe the details of our approach for constructing the RSU schedule to minimize their total energy consumption while still maintaining the connectivity in the VANET. As we have explained, the optimization problem could be divided into two subproblems. The *snapshot selection* problem decides a series of necessary time points for RSUs to change their modes according to the temporal graph of the VANET; The *snapshot scheduling* problem selects the minimum number of RSUs to be activated from the snapshot of the VANET at a specific time point. For the ease of readers, we will first present the complete framework of our approach, and then provide the solutions for the two subproblems in detail.

## 5.1 Framework

In our approach, initially we set all the RSUs to be active. Each vehicle in the network updates its information including the current position, speed, destination and the route to follow, to the nearby RSUs at time 0. Here we assume that all the vehicles have the information of their routes as soon as they decide their destinations. With all these information collected by the RSUs, the VCN could calculate the temporal graph of the VANET for the given time period $[0, T]$. This temporal graph will be updated whenever an existing vehicle starts a new route.

Next, taking the temporal graph as the input for the *snapshot selection* problem, we first find a set of time points $P = \{t_1, t_2, \ldots, t_m\}$ when RSUs adjustment is necessary. If later at some time point $t$ in $[0, T]$, the temporal graph is updated, we will also update the set $P$ at $t$ accordingly. Therefore, the *snapshot selection* involves not only selecting snapshot initially, also updating the set of selected snapshots in the runtime.

for each time point $t_i (1 \leq i \leq m)$ in $P$, we provide an algorithm for determining which RSUs will be turned on or turned off based on the snapshot $G_T(t_i)$. Note that we also construct a schedule for turn off the unnecessary RSUs at the initial time point 0. Only those RSUs selected based on the snapshot $G_T(0)$ will be kept active until the next time point $t_1$ in $P$ if the temporal graph of the network is not updated during $[0, t_1]$. This approach will guarantee the connectivity of the VANET at any time $t$ in the period $[0, T]$, and minimize the total energy consumption for the RSUs.

In the following sections, we will explain in detail how the two subproblems: the *snapshot selection* problem and the *snapshot scheduling* problem are solved.

## 5.2 Snapshot Selection

We need to answer two questions to address the snapshot selection problem. 1) How to select the appropriate time points for necessary RSUs adjustment based on a given temporal graph? and 2) how to modify the selected time points for RSUs adjustment when the temporal graph is updated? Before answering these two questions, let's first study some characteristics of the temporal graph. Given a temporal graph $G_T$, we have the following observations:

1. The set of appropriate time points for RSUs adjustment is a subset of the collection of distinct starting points and ending points of the time intervals contained in the temporal vectors in $G_T$.

2. Not every time point contained in this collection is an appropriate time point for RSUs adjustment.

Intuitively, RSUs adjustment is only necessary when the network partition situation changes. It is obvious that this change happens either when some edges become effective, or some edges become noneffective in the temporal graph. Thus for every time interval $[t_{k1}, t_{k2}]$ of an edge, both the starting time $t_{k1}$ and ending time $t_{k1}$ are possible candidates of time points for necessary RSUs scheduling adjustment, but only those that involve a network partition change will be included in $P$.

**Algorithm 1** Snapshot Selection - Initialization ($G_T$)

Initialize set $P$ as an empty set.
**for** each edge $e$ in the graph $G_T$ **do**
  **for** each time interval $[t_{k_1}, t_{k_2}]$ in the temporal vector associated with $e$ **do**
    **if** $t_{k_1}$ is not contained in the $P$ **then**
      Add $t_{k_1}$ into $P$.
    **end if**
    **if** $t_{k_2}$ is not contained in the $P$ **then**
      Add $t_{k_2}$ into $P$.
    **end if**
  **end for**
**end for**
Sort the set $P$ into a non-decreasing order $\{t_1, \ldots, t_n\}$.
Set $i = 1$, and $N = n$.
**while** $i < N$ **do**
  Partition the subgraph of snapshots $G_T(t_i)$ and $G_T(t_{i+1})$
  REMOVE = TRUE;
  **if** partitions in $G_T(t_i)$ and $G_T(t_{i+1})$ are different **then**
    Set REMOVE = FALSE;
  **end if**
  **if** REMOVE == TRUE **then**
    Remove $t_{i+1}$ from $P$
    Set $i = i - 1$ and $N = n - 1$
  **end if**
  $i = i + 1$
**end while**
Output set $P$.

**Algorithm 2** Snapshot Selection - Update ($P, G'_T$)

Initialize set $P'$ to be empty.
**for** each newly updated edge $e$ in the graph $G'_T$ **do**
  **for** each newly updated time interval $[t_{start}, t_{end}]$ in the temporal vector associated with $e$ **do**
    **if** $t_{start}$ is not contained in the $P$ **then**
      Add $t_{start}$ into $P'$.
    **end if**
    **if** $t_{end}$ is not contained in the $P$ **then**
      Add $t_{end}$ into $P'$.
    **end if**
  **end for**
**end for**
Sort the set $P'$ into a non-decreasing order $\{t_1, \ldots, t_n\}$.
**for** Each time point $t'$ in $P'$ **do**
  // $t$ the largest time point in $P$ satisfying $t \leq t'$.
  Partition the subgraph of snapshots $G'_T(t)$ and $G'_T(t')$
  REMOVE = TRUE;
  **if** partitions in $G'_T(t)$ and $G'_T(t')$ are different **then**
    Set ADD = TRUE;
  **else**
    Set ADD = FALSE
  **end if**
  **if** ADD == TRUE **then**
    Add $t'$ into $P$.
  **end if**
**end for**
Output set $P$.

With these two observations, we propose the following snapshot selection algorithm using graph partitioning strategy. We first select all the different time points $t_i$ from the boundaries of the time intervals in the temporal graph and set it to be $P = \{t_1, t_2, \ldots, t_n\}$. Then for each time point in $P$, we extract its snapshot. By applying bread-first-search(BFS) search on the snapshot, we partition it so that any pair of connected vertices are put in the same partition. Starting from the first pair of consecutive time points $t_1$ and $t_2$ in $P$, we compare whether they have the same the network partition. If their network partitions are the same, $t_2$ will be removed from $P$. If time $t_2$ is removed, then we will further compare $t_1$ with $t_3$; Otherwise, we will keep on comparing $t_2$ with $t_3$. This process continues until we reach the end of the sequence in $P$. The details of the approach is summarized in Algorithm 1.

On the other hand, when an existing vehicle chooses a new route, the temporal vector of existing edges in the temporal graph will be updated. In this case, we need to update the set $P$ as well. The detailed process is presented in Algorithm 2. It follows the similar strategy as in Algorithm 1. In order to minimize the cost of each update, we will fully utilize the set $P$ we have calculated before the update. We use $P'$ to represent the newly introduced candidates of the time points for RSU scheduling adjustment. Instead of comparing each pair of elements in the set $P' \cup P$, we compare each element $t'_i$ in $P'$ with its closest neighbor $t$ in $P$ satisfying $t \leq t'_i$. Based on the graph partition information, we can decide whether to add $t'_i$ into $P$ or not.

## 5.3 Snapshot Scheduling

After determining the sequence of time points when the snapshot of the VANET should be updated, the next step is to decide the mode (active or sleep) of each RSU in the VANET for each snapshot. The snapshot scheduling problem can be formalized as following. Given a *snapshot* $G_T(t) = (V, E, S)$, where $S$ is the set of vertices representing vehicles as well as the VCN, we look for a subset $R' \subseteq V \setminus S$, so that the subgraph induced by the vertex set $R' \bigcup S$ in graph $G(t)$ is connected. It's easy to observe that this subproblem is an instance of the *STP-MSP* problem. Recall that the *STP-MSP* problem is to look for the Steiner tree with the minimum number of steiner nodes interconnecting the given terminal set. As most of the existing work for the *STP-MSP* problem focus on Euclidean plane, in this section, we propose an approximation algorithm for the *STP-MSP* problem in the general graphs and give a theoretical performance bound based on the maximum degree of the final Steiner tree.

In our algorithm, we first introduce a weight function $w : V \rightarrow \{0, 1\}$ into the *snapshot* $G_T(t)$, assigning different weights to different kinds of vertices. Each vertex in the terminal set $S$ is given the same weight of 0, and every other vertex in the graph has the same weight of 1. In this way, we could convert the graph $G_T(t)$ into a node-weighted graph first. Then, we construct an edge-weighted graph $G'_T(t)$ from $G_T(t)$ by initializing $G'_T(t)$ with the same node-set and edge-set but a different edge weight function $w'$. For every edge $(u, v)$ in $G'_T(t)$, let the edge weight $w'(u, v) = \frac{1}{2}(w(u) + w(v))$. Afterwards, we compute a Steiner tree $T(t)$ of $G'_T(t)$ on $S$ using the $\rho$-approximation algorithm for Minimum Steiner Tree [14]. Obviously, this Steiner tree $T(t)$ could be viewed as the *STP-MSP* of $G_T(t)$ on $S$, and it is our final result. The pseudo-code of this algorithm is presented in algorithm 3.

Given a snapshot $G_T(t)$, the following lemma gives the relationship between the number of RSUs in optimal solution and that chosen in our algorithm 3. For the ease of readers, we use $G$ as a short form of $G_T(t)$, and $G'$ as a short form of $G'_T(t)$.

LEMMA 1. *Denote $T_{opt\_G}$ as the optimal Steiner tree in*

**Algorithm 3** Snapshot Scheduling ($G_T(t) = (V, E, S)$)

**for** every vertex $v$ in $V$ **do**
  **if** $v \in S$ **then**
    $w(v) = 0$;
  **end if**
  **if** $v \in V \setminus S$ **then**
    $w(v) = 1$;
  **end if**
**end for**
Initialize an edge-weighted graph $G'_T(t) = (V', E', w', S')$ by setting $V' = V$,$S' = S$ and $E' = E$
**for** each edge $(v_i, v_j)$ in graph $G'_T(t)$ **do**
  $w'(v_i, v_j) = (w(v_i) + w(v_j))/2$.
**end for**
$T = \mathbf{SMT}(G'_T(t), S)$
Output the vertices in $T$ as the set of active RSUs at time $t$.

*node-weighted graph $G$ and $T_{opt\_G'}$ as the optimal Steiner tree in edge-weighted graph $G'$ we constructed. Define a function $C : G \to \mathcal{N}$, calculating the number of Steiner points contained in a tree. We have that $C(T) \leq \frac{max_{v \in T_{OPT\_G}} d(v)}{2} \rho C(T_{opt\_G})$, where $max_{v \in T_{OPT\_G}} d(v)$ is the maximum degree of vertices in the optimal solution $T_{opt\_G}$.*

PROOF. Following the notation in algorithm 3, denote $T$ as the output of the algorithm 3. As we could consider tree $T_{opt\_G}$ as a Steiner tree for terminal set $S$ in graph $G'$, we first have,

$$
\begin{aligned}
w'(T) &\leq \rho \cdot w'(T_{OPT\_G'}) \leq \rho \cdot w'(T_{OPT\_G}) && (1)\\
&= \rho \cdot \sum_{(u,v) \in E(T_{OPT\_G})} \frac{1}{2}(w(u) + w(v)) && (2)\\
&= \rho \cdot \sum_{u \in V(T_{OPT\_G})} \frac{d_{T_{OPT\_G}}(u)}{2} w(u) && (3)\\
&\leq \rho \cdot \frac{max_{v \in T_{OPT\_G}} d(v)}{2} \sum_{u \in V(T_{OPT\_G})} w(u) && (4)\\
&= \rho \cdot \frac{max_{v \in T_{OPT\_G}} d(v)}{2} w(T_{OPT\_G}) && (5)\\
&\leq \rho \cdot \frac{max_{v \in T_{OPT\_G}} d(v)}{2} w(T_{OPT\_G}). && (6)
\end{aligned}
$$

As the weight we assign to each vertex in $V \setminus S$ is exactly 1 and all the Steiner nodes have a degree of no less than 2, we have

$$
\begin{aligned}
C(T) &= w(T) \leq w'(T) && (7)\\
&\leq \rho \cdot \frac{max_{v \in T_{OPT\_G}} d(v)}{2} w(T_{OPT\_G}) && (8)\\
&= \rho \cdot \frac{max_{v \in T_{OPT\_G}} d(v)}{2} C(T_{OPT\_G}). && (9)
\end{aligned}
$$

□

Studying about the snapshots of the temporal graph derived from VANET further, we observe the following properties for the maximum degree of vertices in the optimal Steiner tree.

1. For a given terminal set $S$, the maximum degree of all the vertices in the Steiner tree for the STP-MSP varies among different snapshots.
2. For different terminal sets, the maximum degree of all the vertices in the Steiner tree for the STP-MSP are different as well.

However it is obvious that the maximum degree of vertices representing vehicles in the network will not exceed 2 in the system. As for the maximum degree of vertices representing the RSUs, the maximum number will depend on two factors. First, it depends on the time. With the change of the time, the number of disconnected components of vehicles in the network changes; Second, it depends on the density of the VANET. When the number of vehicles in the VANET is very large, the number of disconnected components will be lowered and the maximum degree of vertices in the Steiner tree will become smaller as well. No matter how different the value of the maximum degree could vary, there is no doubt that it will be smaller than $n + 1$ when the total number of vehicles in the network is $n$, Therefore, our algorithm could achieve an approximation ratio of $\rho * \frac{n+1}{2}$.

# 6. PERFORMANCE EVALUATION

In order to evaluate the performance of our scheduling scheme in practice, we conduct experiments on the Dallas downtown area in the state of Texas. Figure 3 provides an overview of the area we experiment on. It covers approximately $2 \times 2 km^2$ and contains around 20 major roads. Those roads highlighted in dark in figure 3 are the roads we will focus on in the area.

Figure 3: Dallas Downtown Area

## 6.1 System Framework and Environment Setup

Our system mainly contains the following modules, the Map Loader, the RSU Generator, the Vehicle Generator and the RSU Scheduler. They are operated in a sequence as illustrated in Figure 4. The road map is loaded first. Each road in the map is stored in our system as a set of line segments. Next, the RSU Generator generates the coordinates of all the RSUs according to the map. These RSUs are deployed according to our description in section 4 and cover each road in the system. The Vehicle Generator takes control afterwards. A set of vehicles with chosen speeds and calculated routes are introduced into the system in this module.

After all these steps, the system walks into the RSU Scheduler module, the essential module of the whole system. Implementing the snapshot selection algorithm as well as the RSU snapshot scheduling algorithm proposed in section 5, it outputs the set of RSUs that needs to be activated to guarantee the connectivity of the network. In detail, at each time spot, the coordinates for each existing vehicle in the system is updated according to its speed and calculated route.

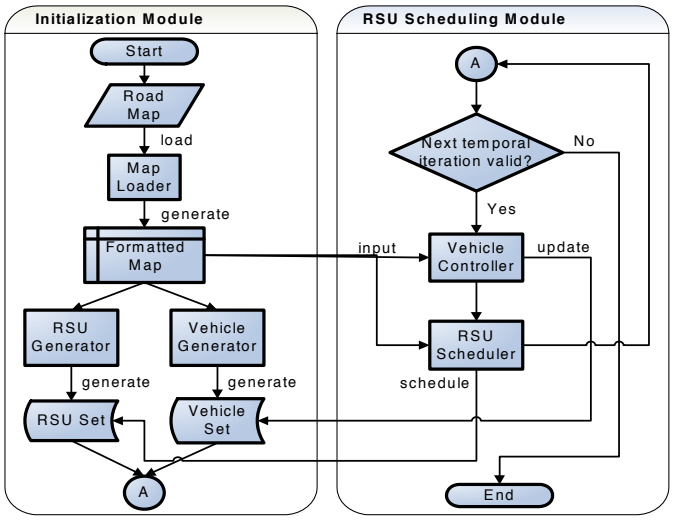


Figure 4: System Flow Diagram

For those vehicles that have reached destinations, they will choose a new destination and start from its current position. These steps are included in the Vehicle Controller submodule in our system. Afterwards, the RSU Scheduler generates the set of RSUs that needs to be activated and provides the appropriate scheduling adjustment.

Based on the size of the area we consider, as well as the real traffic data of downtown area released by the department of public transportation of the city of dallas at Texas [**?**], we set the maximum number of vehicles allowed to appear at the same time in our system to be 2000. Thus the maximum average density of vehicles would be around $500/km^2$. The maximum speed of vehicles is set to be 60mile/hour. As we mentioned earlier, the time in our system is implemented discretely. Every time unit is set equal to 10ms in our system.

In the following, we study the performance of our scheme by investigating the impact of several important vehicles' parameters on the percentage of RSUs chosen in our experiment. The parameters we will consider in our experiment are the total number of vehicles as well as the communication range of vehicles and RSUs.

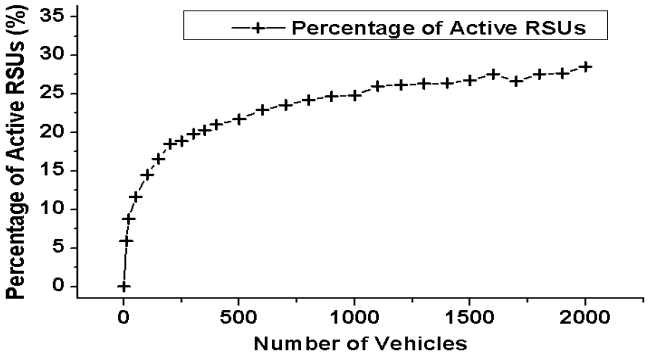


Figure 5: Number of Vehicles w.r.t. the Percentage of Active RSUs

## 6.2 Experiments and Evaluation

In our first set of experiments, we vary the number of Vehicles in the target region, and study how the change of the number of vehicles will affect the number of RSUs selected. We set the vehicle transmission range to be 50 meters and RSU transmission range to be 250 meters. We vary the number of vehicles from 0 to 2,000 and get figure 5. The percentage of RSU selected for each case (e.g. when number of vehicles is 500) is an average value calculated based on 1000 rounds of the same case.

As shown in Figure 5, the total number of RSUs used to connect the vehicles is increasing rapidly before the number of vehicles reaches around 400. After that, it slowly increases and the value is tend to be stabilized as the number of vehicles passing 1,000 and approaching 2,000. Even when the number of vehicles reaches 2,000, the number of active RSUs to connect all the vehicles is on average only around 12. This result indicates that our algorithm effectively selects the minimal RSU set for connecting all the vehicles in the target region and reduces the energy consumption of RSUs in a great extent.

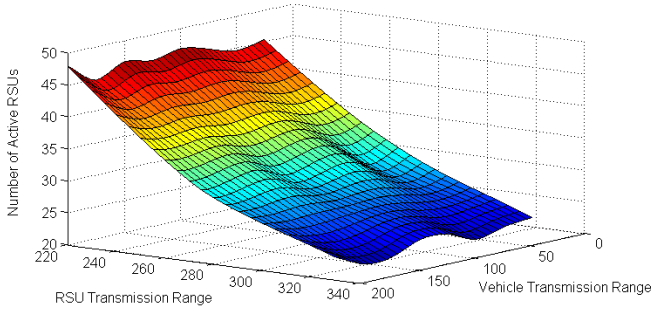


Figure 6: Change of Vehicle Transmission Range & RSU Transmission Range w.r.t. the Number of Active RSUs

In our second set of experiments, we vary the vehicle transmission range as well as the RSU transmission range to study their impacts on the number of active RSUs selected for the target area. In our experiment, the total number of vehicles is set to be 1000. The vehicle transmission range varies from 25 meters to 200 meters, and the RSU transmission range varies from 220 meters to 330 meters. The number of active RSUs we show in the figure 6 is the average value calculated based on 1000 rounds of the same case. Figure 6 provides a three dimensional view of the relationship between vehicle transmission range, RSU transmission range and the number of active RSUs. In this experiment, the size of the RSU set deployed in the target region decreases from 100 to 80 with the increasing of the RSUs' transmission range. The maximum and the minimum number of active RSUs in this figure is around 48 and 22. If fixing the RSU transmission range(e.g. 300), we can observe that the number of active RSUs does not change much. This shows that our RSU scheduling scheme could provide great performance disregarding the change of the vehicle transmission range in the given range. If fixing the vehicle transmission range instead(e.g. 50), the number of active RSUs decreases quickly with the increasing of the RSU transmission range. This result shows that the change of the RSU transmission range has more impact on the number of active RSUs se-

lected than the change of the vehicle transmission range. This phenomenon is caused by the larger transmission capability of RSUs compared with the vehicles in the system.

**Summary of our results.** It is demonstrated in our experimental studies that our RSU scheduling scheme could effectively reduce the energy consumption of RSUs in a great extent by selecting the minimal number of RSUs in various cases, such as the increasing of total number of vehicles as well as the vehicle transmission ranges. We also show in our experiment that the change of the RSU transmission range has a great impact on the total number of active RSUs selected in our scheduling scheme at each time.

# 7. CONCLUSION AND FUTURE WORK

For the purpose of preventing the network partition, we study the problem of the RSU scheduling in a discrete time system. Considering the importance of the network connectivity, our target is to provide a RSU scheduling scheme, which will not only guarantee full coverage in the VANET, but also achieve the minimum energy consumption.

We formalize this problem as an combinatorial optimization problem and divided them into two subproblems namely *Snapshot Selection* and *Snapshot Scheduling*. By providing solutions for these two subproblems, we successfully present a complete scheduling scheme for RSUs in the VANET in this paper. Theoretical analysis shows that our scheduling scheme achieves $\rho * \frac{n+1}{2}$ -approximation ratio when the total number of vehicles in the network is $n$. We further evaluate the performance of our scheme in the dallas downtown area in the state of Texas. Experiment results show that our algorithm can efficiently select the minimal percentage of RSUs to connect up to thousands of vehicles in the target region and successfully reduce the energy consumption of RSUs in the meantime as well.

We are interested in quite a few directions as our future work. First of all, as our current system is centralized, distributed version of it would be a very interesting also challenging research direction. Second, we are looking forward to further study about one drawback of our system in the future. In our system, we require each vehicle updates their information including speed and calculated route to the closest active RSU. For the situation when there are no active RSUs near the vehicle, the system's response time and performance would be affected. Therefore, we are looking forward to design detailed protocols to solve this problems. Last but not least, we are interested in studying about the impact of different vehicle mobility models on the scheme in the future.

# 8. REFERENCES


[1] Berman, P., and Ramaiyer, V. Improved approximations for the steiner tree problem. In *selected papers from the third annual ACM-SIAM symposium on Discrete algorithms* (Orlando, FL, USA, 1994), Academic Press, Inc., pp. 381–408.

[2] Bettstetter, C. Smooth is better than sharp: a random mobility model for simulation of wireless networks. In *Proceedings of the 4th ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems* (New York, NY, USA, 2001), MSWIM '01, ACM, pp. 19–27.

[3] Broch, J., Maltz, D. A., Johnson, D. B., Hu, Y.-C., and Jetcheva, J. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking* (New York, NY, USA, 1998), MobiCom '98, ACM, pp. 85–97.

[4] Camp, T., Boleng, J., and Davies, V. A survey of mobility models for ad hoc network research. *WIRELESS COMMUNICATIONS and MOBILE COMPUTING (WCMC): SPECIAL ISSUE ON MOBILE AD HOC NETWORKING: RESEARCH, TRENDS AND APPLICATIONS 2* (2002), 483–502.

[5] Chen, D., Du, D.-Z., Hu, X.-D., Lin, G.-H., Wang, L., and Xue, G. Approximations for steiner trees with minimum number of steiner points. *J. of Global Optimization 18* (September 2000), 17–33.

[6] Das, S. R., and Perkins, C. E. Performance comparison of two on-demand routing protocols for ad hoc networks. pp. 3–12.

[7] Du, D.-Z., Wang, L., and Xu, B. The euclidean bottleneck steiner tree and steiner tree with minimum number of steiner points. In *Proceedings of the 7th Annual International Conference on Computing and Combinatorics* (London, UK, 2001), COCOON '01, Springer-Verlag, pp. 509–518.

[8] Garey, M. R., and Johnson, D. S. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[9] Hougardy, S., and Prömel, H. J. A 1.598 approximation algorithm for the steiner problem in graphs. In *Proceedings of the tenth annual ACM-SIAM symposium on Discrete algorithms* (Philadelphia, PA, USA, 1999), SODA '99, Society for Industrial and Applied Mathematics, pp. 448–453.

[10] Johnson, D. B., and Maltz, D. A. Dynamic source routing in ad hoc wireless networks. In *Mobile Computing* (1996), Kluwer Academic Publishers, pp. 153–181.

[11] Lin, G.-H., and Xue, G. Steiner tree problem with minimum number of steiner points and bounded edge-length. *Inf. Process. Lett. 69* (January 1999), 53–57.

[12] Lochert, C., Scheuermann, B., Wewetzer, C., Luebke, A., and Mauve, M. Data aggregation and roadside unit placement for a vanet traffic information system. In *Proceedings of the fifth ACM international workshop on VehiculAr Inter-NETworking* (New York, NY, USA, 2008), VANET '08, ACM, pp. 58–65.

[13] Naumov, V., Baumann, R., and Gross, T. An evaluation of inter-vehicle ad hoc networks based on realistic vehicular traces. In *Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing* (New York, NY, USA, 2006), MobiHoc '06, ACM, pp. 108–119.

[14] Robins, G., and Zelikovsky, A. Improved steiner tree approximation in graphs. In *Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms* (Philadelphia, PA, USA, 2000), SODA '00, Society for Industrial and Applied Mathematics, pp. 770–779.

[15] Royer, E. M., and Perkins, C. E. Multicast operation of the ad-hoc on-demand distance vector routing protocol. In *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking* (New York, NY, USA, 1999), MobiCom '99, ACM, pp. 207–218.

[16] Saha, A. K., and Johnson, D. B. Modeling mobility for vehicular ad-hoc networks. In *Proceedings of the 1st ACM international workshop on Vehicular ad hoc networks* (New York, NY, USA, 2004), VANET '04, ACM, pp. 91–92.

[17] Zhang, Y., Zhao, J., and Cao, G. Service scheduling of vehicle-roadside data access. *Mob. Netw. Appl. 15* (February 2010), 83–96.