Allocation of Roadside Units for Certificate Update in Vehicular Ad Hoc Network Environments

Sheng-Wei Wang
Department of Applied Informatics
Fo Guang University
Yilan 26247, TAIWAN

***Abstract*—The roadside unit (RSU) plays an important role in VANET environments for privacy preservation. In order to conserve the privacy of a vehicle, the issued certificate must be updated frequently via RSUs. If a certificate expires without being updated, the services for the vehicle will be terminated. Therefore, deploying as more as possible RSUs ensures that the certificate can be updated before it expires. However, the cost for allocating an RSU is very high. In this paper, we consider the roadside unit allocating problem such that the certificates can be updated before it expired. Previous researches focus on the roadside unit placement problem in a small city in which for any origination-destination pair the certificate is limited to update at most once. The RSU placement problem in which more than once certificate updates are required is discussed in this paper. The RSU allocation problem is formulated and the decision problem of the RSUs allocation problem is proved as an NP-complete problem. We proposed three roadside unit placement algorithms which works well for a large city. In order to reduce the number of required RSUs for certificate update, we also proposed three backward removing methods to remove the intersections found by the RSU allocation methods. Simulation results show that the proposed algorithms yields lower number of required RSUs than the simple method named the most driving routes first method. One backward removing method named the least driving routes first backward removing method was shown to be able to further reduce the number of required RSUs.**

***Keywords*—Roadside units allocation, VANET, certificate update, privacy conservation, NP-complete**

I. Introduction

A large number of services such as driving route planning, on-line maps and instant accident notifications require the vehicles to communicate with each other or to connect to the Internet [2]. The number of such services is still increasing. With the growing number of the services, vehicular ad hoc networks(VANETs) are used as the infrastructure of service platform [3].

VANET is an instantiation of mobile ad hoc networks (MANETs) [3]. The main difference between VANETs and MANETS is the components of the networks. In MANETs, the mobile nodes move randomly and no fixed base station is established. However, VANETs consist of vehicles and a number of fixed roadside units (RSUs) to support message exchange. A typical VANET includes three major components, namely, trust authority(TA), on-board units(OBUs) and the roadside units(RSUs) [2]. Fig. 1 shows the relationships between TA, OBUs and RSUs in VANET environments. The functions of the three components are described as follows:

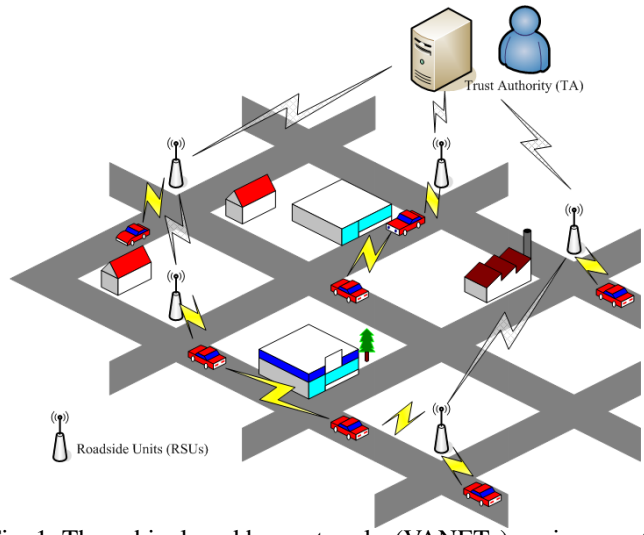

Fig. 1: The vehicular ad hoc networks (VANETs) environments

- Trust Authority (TA): The trust authority is a server which is managed by a service provider or the government. The function of a trust authority is to maintain the service, to keep the records of each vehicles or to issue the certificate for each vehicles.
- On-Board Unit (OBU): The on board unit(OBU) is equipped on a vehicle for inter-vehicles communications or communications between the vehicle and roadside units. An antenna is equipped in an OBU such that the vehicle communications with each other or the roadside units can be made.
- Roadside Unit (RSU): The roadside units are deployed on the traffic signs or along the roads. The main function of the roadside units is to bypass the messages between the vehicles and trust authority. Even though the function provided by the RSU is simple, RSU is a very important component in the VANET environments. If the number of RSU is small or if the RSUs are allocated inappropriately, the performances of the VANETs will become degraded.

In VANET environments, privacy conservation is an important issue in providing services to vehicles [4]. A number of mechanisms for conserving privacy have been proposed in [4]–[8]. Among the mechanisms, using certificate to identify the

owner of a message is an efficient way for secure service providing [7], [8]. However, if a certificate can not be updated for a long time, the certificate may be stolen by a potential eavesdropper. In order to prevent the certificate from being stolen, the certificate of each vehicle should be updated frequently. If the certificate can be updated more frequently, the services in which the authentication is made via the certificate will be more secure.

In VANET environments, a certificate update request is sent to trust authority via roadside units. The roadside unit receiving the certificate update request will pass the request to the trust authority. If the certificate is valid, the authority will issued a new certificate to the vehicle. The certificate which is no longer used will be put into the Certificate Revocation Lists (CRL) which disable the validity of the certificate [9]. When the vehicle receives a new certificate, the services can be provided to the vehicle continuously.

Although the RSU is important for certificate update, the cost for allocating an RSU is very high. To save the cost, only a small number of RSUs can be deployed in a city. How to allocate the RSUs in the VANET environment such that some objectives can be optimized is referred to as the **RSU allocation problem**. In this paper, we consider the problem of deploying a small number of RSUs such that the certificate can be efficiently updated in this city without expiration of a certificate.

In this paper, we directly show that the decision problem of RSU allocation problem is NP-complete. Three allocation methods, namely, *the most driving routes first* and *the most satisfied intersection pairs first*, and *the critical intersections first* methods, are proposed to find the locations for RSU deployment. Since the proposed algorithms are greedy based algorithms, we also proposed three backward removing methods to remove the some RSUs in the solutions found by the proposed RSU allocation methods. The proposed methods can be applied in a large city or under a short certificate updating interval environment. Simulation results show that our proposed the most satisfied intersection pairs first method and the critical intersections first method both yield lower number of required RSUs than the most driving routes first algorithm. We also show that one backward removing method named the least driving routes first method performs better than the other two backward removing methods.

The rest of this paper is as follows. Section II studies the related works in RSU allocation problems. In Section III, the RSU allocation problem is formulated and the NP-completeness of the problem is proved. Section IV provides three allocating methods and three backward removing methods for RSU allocation. Simulation results are shown and discussed in Section V. Finally, some concluding remarks are given.

II. Related Works

A number of researches have been proposed to find the placement for RSUs in the VANET environments [4], [10], [11]. In [11], the authors proposed an analytical model to estimate the minimum number of required RSUs where the packet delay between the vehicles and RSUs are bounded. The RSU placement problem which minimizes the disconnection time and maximizes the connectivity between the vehicles and RSUs are studied in [10] and a placement scheme was proposed.The RSU allocation problems focused in [11] and [10] are not the same as the problem focused in this paper.

In [4], the RSU problem for certificate update is studied and is transformed into a set cover problem. In [4], the transformed problem is proved as an NP-Hard problem and a greedy algorithm is proposed. However, since the instances transformed from RSU placement problem to the set cover problem are not proved as the general cases for set cover problem, the fact that the optimal set cover problem is NP-hard does not ensure that the RSU placement algorithm is NP-hard [12]. Besides, the algorithm proposed in [4] only applies to a small city in which only at most once certificate updating is allowed when driving along every shortest path between an origination-destination pair. Therefore, the authors do not take a large city or a short certificate valid interval into account.

TABLE I: Table of Notations of RSU Allocation Problem

Notation	Definition
M	a graph represents a city map
I	the set of intersections in M
R	the set of roads in M
T	the set of driving times in M
r_{ij}	the road from intersection i to j
t_{ij}	the driving time from intersection i to j
(s,d)	a origination-destination pair with origination intersection s and destination intersection d
$P(s,d)$	the driving route from intersection s to d
$T(s,d)$	the total driving time from intersection s to d
$S(s,d,k)$	the kth segment between (s,d)
$N(s,d)$	the number of segments in driving route $P(s,d)$
$T(s,d,k)$	the driving time on segment $S(s,d,k)$
n_{sd}	the number of all segments between (s,d)
A_i	an indicator to indicate whether an RSU is allocated on intersection i
C_T	the length of certificate valid interval
N	the required number of RSUs in city M

III. The RSU Allocation Problem

In this section, we first formulated the RSU allocation problem. The notations used in this section are summarized in Table I. We also proved that the decision problem of RSU allocation problem is NP-complete in this section.

A. Problem Formulation

In the RSU allocation problem, a map of a city is denoted as $M = (I, R, T)$ where I is the set of the intersections, R is the set of roads, and T is the driving time on each road in this city. Each road in this city is directional. If there is a road from one intersection to another, there is also a road in the reverse direction. The driving time on both directions are not necessarily the same.

The road and the driving time from from intersection i to its neighborhood intersection j are denoted as the r_{ij} and t_{ij} respectively. Let (s,d) be denoted as the origination(source)-destination intersection pair from intersection s to intersection d. The driving route between intersections s and d is denoted as $P(s,d)$ and the driving time on $P(s,d)$ is denoted as $T(s,d)$.

We assume that the RSUs can only be allocated in the intersections since the RSUs are always deployed on the traffic

signs and the certificate can be updated only when the vehicle passed the intersections. When some RSUs are allocated in the city, each driving route in the city may be divided into several sub-routes, namely, segments. The endpoints of a segment is the origination intersection, destination intersection or the intersections with RSU allocated. Any non-endpoint in a segment is an intersection without RSU allocated. We let the segment $S(s,d,k)$ be the kth segment on driving route $P(s,d)$. The number of segments in driving route $P(s,d)$ is $N(s,d)$. The driving time on segment $S(s,d,k)$ is $T(s,d,k)$.

The indicator A_i indicates whether the intersection i is allocated an RSU or not. That is,

$$A_i = \begin{cases} 1 & \text{if an RSU is allocated at intersection } i, \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

The number of RSU allocated is denoted as N. That is,

$$N = \sum_{i \in I} A_i. \quad (2)$$

The length of a certificate valid interval is denoted as C_T which is a fixed value. To update the certificate before it expires, the length of the certificate valid interval should be longer than or equal to the driving time over each segment. That is,

$$C_T \geq T(s,d,k),\ \forall i,j \in I,\ i \neq j, \text{ and } k = 1,2,\cdots,N(s,d). \quad (3)$$

If driving time on a road r_{ij} is larger than C_T, the certificate can not be updated no matter how the RSUs are allocated. To ensure that the certificate can be updated on all driving route, we assume that the driving time on each road r_{ij}, t_{ij}, is less than or equal to the length of certificate valid interval C_T. That is,

$$C_T \geq t_{ij}, \quad \forall i,j \in I, \quad i \neq j. \quad (4)$$

The objective function for the RSU allocation problem is to minimize the number of required RSUs in the city such that the certificate can be updated before it expires on all driving routes.

The RSU allocation problem can be described as follows. *Given a city map $M = (I,R,T)$, the driving route $P(s,d)$ for all intersections s and d, and the length of certificate interval C_T, the objective is to find a subset of intersections A such that each segment $S(s,d,k)$ on $P(s,d)$, the driving time $T(s,d,k)$ is shorter than or equal to the length of certificate valid interval C_T and the number of intersections in A is minimized.*

B. NP-Completeness of The RSU Allocation Problem

To prove the NP-completeness of the RSU allocation problem, we transform the problem into a decision problem.

> *Given a city map $M = (I,R,T)$, the driving route $P(s,d)$ for all intersections s and d, and the length of certificate interval C_T, we want to find a subset of intersections A such that each segment $S(s,d,k)$ on $P(s,d)$, the driving time $T(s,d,k)$ is shorter than or equal to the length of certificate valid interval C_T and the number of intersections in A is less than or equal to N?*

TABLE II: Table of Notations for Hitting Set Problem

Notation	Definition
F	a finite set
f_i	a subset of F
e_i^j	the jth element in f_i after sorting
C	the set of f_i
F'	a hitting set for C
M'	corresponding map for hitting set problem
I'	corresponding set of intersections
R'	corresponding set of roads
T'	corresponding set of driving times
P_i'	the driving route in M' corresponding to f_i
P'	the set of driving routes in M'

We proved the decision problem of RSU allocation problem is NP-complete in the following theorem.

Theorem 1 (NP-Completeness): The RSU-ALLOCATION-PROBLEM is NP-complete.

Proof: First, we will show the RSU allocation problem is in NP. Given a set intersections on the city map, it is easy to check whether the allocations for RSU placement is sufficient for certificate update and the number of intersections is less than or equal to N. Time complexity of the verification is polynomial time where the computation time is proportional with the number of origination-destination pairs in the given city. A solution for the RSU allocation problem can be verified in polynomial time; that is, the RSU allocation problem is in NP.

In order to prove that the RSU allocation problem is NP-complete, we first find an existing NP-complete problem to reduced to the RSU allocation problem. We transform the HITTING SET problem [12] to the RSU-ALLOCATION-PROBLEM. The MINIMUM-HITTING-SET is described as follows [12].

> *HITTING-SET PROBLEM: Given a collection C of subsets of a finite set F, we want to find a hitting set for F, i.e., a subset $F' \subseteq F$ such that F' contains at least one element from each element in C and the cardinality of the hitting set, i.e., $|F'|$ is less than or equal to K.*

The notations used for HITTING SET problem is summarized in Table II.

We then construct the corresponding city maps $M' = (I',R',T')$ for an instance of the hitting set problem as follows. Let each intersection in I' represent each element in F and for each element f_i in C, add two intersections s_i and d_i into the set of intersections I'.

For each element f_i in C, sort the elements in f_i in ascending order first. The elements in f_i can be denoted as $f_i = \{e_i^1, e_i^2, \cdots, e_i^{k_i}\}$ where k_i is the number of elements in f_i. Note that if $a < b$, e_i^a is smaller than e_i^b. For each element e_i^j, $j = 1,2,\cdots,k_i - 1$ in sorted set f_i, add a road from intersection e_i^j to e_i^{j+1} and the driving time on the road is set to 0. Next, for each sorted set f_i, add two roads from s_i to e_i^1 and from $e_i^{k_i}$ to d_i with driving time T where T is an arbitrary positive real number. The driving routes P_i' can be constructed from each s_i to d_i along the intersections specified

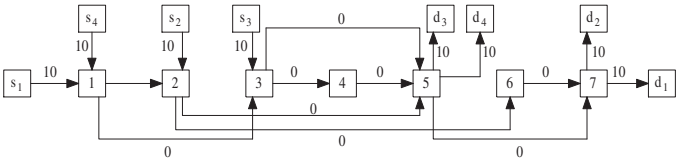

Fig. 2: Example for proof of Theorem 1

in f_i. Finally, we set the length of certificate valid interval C_T to $\frac{3T}{2}$ and set the value of N to the value of K.

For example, given a finite set $S = \{1, 2, 3, 4, 5, 6, 7\}$ and a collection $C = \{\{1, 3, 5, 7\}, \{2, 6, 7\}, \{3, 4, 5\}, \{1, 2, 5\}\}$, we construct the map as follows. The intersections from 1 to 7 are first constructed and the four pairs of intersections $s_1, d_1, s_2, d_2, s_3, d_3, s_4$ and d_4, are then generated. The set of driving routes P' corresponding to C consists of four components where $P'_1 = \{s_1, 1, 3, 5, 7, d_1\}$, $P'_2 = \{s_2, 2, 6, 7, d_2\}$, $P'_3 = \{s_3, 3, 4, 5, d_3\}$, and $P'_4 = \{s_4, 1, 2, 5, d_4\}$. The roads are added according to the set in collection C and the driving time are added on the roads. We let $T = 10$ and the $C_T = 15$. Fig. 2 shows the map generated after transformation.

For each driving route P'_i, it is obvious that the total driving time is $2T$. From intersection s_i to d_i, the driving time is larger than the length of certificate valid interval $C_T = \frac{3T}{2}$ such that the certificate is required to update at least once. Since the driving time is 0 on the roads not connected to intersection s_i or d_i on driving route P'_i, updating the certificate at each intersection on P'_i excluding s_i and d_i may made the certificate not expired during the driving time on P'_i. Finding an intersection on P'_i is equivalent to find an element in set f_i. Therefore, if the number of intersections in map M' such that certificate can be update before it expires on all driving routes is less than or equal to N, the cardinality of the hitting set for collection C is less than or equal to K. This completes the proof of Theorem 1. ■

IV. The RSU Allocation Methods

In this section, we first proposed three greedy based RSU allocation methods for certificate update. Since the proposed methods are all greedy based methods, the found allocations are not necessarily the optimal solution. We then proposed three backward removing methods to remove some locations of RSUs to obtain a better solution.

A. RSU allocation methods

We proposed three RSU allocation methods, namely, *the most driving routes first* and *the most satisfied intersection pairs first*, and *the critical intersections first* methods, in VANET environments. The three methods are described in the following.

1) The most driving routes first method: The idea of this method is that an intersection with more driving routes passed is more likely to become the location of RSU for certificate update. Hence, allocating an RSU on the intersection is expected to be effective for certificate update on the driving routes. Therefore, this method sorts the intersections in the city in descending order according to the number of driving routes passed the intersection and allocate the RSUs on the intersections one by one until the certificate can be update in time on all driving routes.

The computation time of this method includes the time for checking the number of traversed driving routes through each of the interconnection, the time for sorting the interconnections according the number of passed driving routes of the interconnections, and checking if the placed RSUs are enough for certificate update in this city. The computational complexity of counting the number of traversed driving routes is $O(|I|^2)$ since there are $|I| \times (|I| - 1)$ driving routes in the city, the computational complexity of sorting is $O(|I| \log |I|)$ and the computational complexity of checking if the RSUs allocated are enough for certificate update is $O(|I|)$. Therefore, the computational complexity of this method is $O(|I|^2)$.

2) The most satisfied intersection pairs first method: The idea of this method is as follow. If the location of an RSU on an intersection yields more source-destination pairs between which the certificate can be updated before it expires, it is more beneficial to allocate the RSU on the intersection. The details are described in the following.

Recall that $P(s, d)$ be the driving routes between source-destination intersection pair s and d. Let $A_\emptyset$ be the allocation pattern in which no RSU is allocated in the city. Let $h(A_\emptyset, s, d)$ be an indicator that whether the source-destination intersection pairs between which the certificate can be updated or not before it expires when no RSU is allocated in the city. Note that when no RSU is allocated in the city, only the source-destination intersection pair (s, d) with driving time $T(s, d)$ less than or equal to C_T satisfies the constraint described in equation (3). That is,

$$h(A_\emptyset, s, d) = \begin{cases} 1 & \text{if } T(s, d) \leq C_T, \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

The computational complexity for checking whether the driving time $T(s, d)$ is less than or equal to C_T or not is constant time, i.e. $O(1)$.

Let h be the total number of the source-destination pairs between which the certificate can be updated before it expires when no RSU is allocated in the city. Then h is calculated as follows:

$$h = \sum_{s,d \in I,\ s \neq d} h(A_\emptyset, s, d) \ . \tag{6}$$

The computational complexity for calculating h is $O(|I|^2)$.

Let $h(A_i, s, d)$ denote if the source-destination intersection pairs between which the certificate can be updated before it expires when no RSU is allocated in the city. Let $f(i)$ denote the total number of source-destination intersection pairs between which the certificate can be updated before it expires when a single RSU is allocated at intersection i. Then $f(i)$ is obtained as follows:

$$f(i) = \sum_{s,d \in I,\ s \neq d} h(A_i, s, d) \ . \tag{7}$$

The computational complexity for calculating each $f(i)$, is also $O(|I|^2)$. The computational complexity for calculating all $f(i),\ i \in I$, is $|I| \times O(|I|^2) = O(|I|^3)$.

Let $R(i)$ be the difference of the numbers of source-destination pairs which can be successfully updated when no RSU is allocated and when a single RSU is allocated at intersection i. Then $R(i)$ is given as follows:

$$R(i) = h - f(i) \ . \tag{8}$$

The intersection with highest $R(i)$ is selected for RSU allocation. When the first intersection is found and an RSU is allocated, the next intersection is found by the same procedure which can maximize the difference of the numbers of source-destination pairs between which the certificate can be successfully updated. The intersection with highest difference is selected. The procedure repeats until the certificate on all driving routes in the city can be updated in time. Since the procedure repeats at most $|I|$ times where each intersection us allocated an RSU, the overall computational complexity of this method is $|I| \times O(|I|^3) = O(|I|^4)$.

3) The critical intersections first method: Since the most satisfied intersection pairs first method requires high computational complexity, we devise the following method to reduce the computational time. We first find the *critical intersections* at which an RSU is required for certificate update in all feasible allocation patterns. The critical intersections can be found as follows. For each driving route with more than 2 roads, the non-endpoints on the driving route is checked if it is a critical intersection or not. For each consecutive roads r_{ij} and r_{jk}, intersection j is a critical intersection if the sum of driving time on the two roads, $t_{ij} + t_{jk}$, is larger than C_T. That is, if no RSU is allocated at intersection j, the certificate cannot be updated on the driving route. The computational finding the critical intersection is $O(|I|^3)$ since each intersection on all $|I| \times (|I| - 1)$ driving routes should be checked.

After finding the critical intersections, the RSUs are first allocated on the critical intersections. The most satisfied intersection pairs first method are then employed to find other intersections until the certificate on all driving routes can be updated successfully. The computational complexity of the most satisfied intersection pairs first method is $O(|I|^4)$ which is also the overall computational complexity.

Although the critical intersections first method has the same computational complexity as the most satisfied intersection pairs method, the computation time of the method is significantly less than that of the most satisfied intersection pairs first method under some values of C_T. The computation time of the critical intersections depends on how many critical intersections are found in the city. When C_T is not large, more critical intersections can be found in the first part of the method which reduced the search time. However, when C_T is large, only a few critical intersections can be found and the time for searching the other intersections is long.

B. Backward removing methods

Since there will be some redundant RSUs after allocating the RSUs by the three greedy based methods, we proposed three backward removing algorithms to remove the redundant RSUs. Note that the backward removing methods are applied when a set of intersections which is available for successfully certificate update in a city map is obtained by the RSU allocation methods. The three backward removing methods are described in the following.

1) The random backward removing method: Given a set of intersections which is available for successfully certificate update, the random backward removing method removes the intersections one by one in a random order. If an intersection can be removed such that the set of intersections after removing is also available for certificate update, the intersection will be removed. The random backward removing method removes the intersections one by one until all intersections are checked.

2) The most driving routes first backward removing method: The most driving routes first backward removing method works similar as the random method. The main difference between the two methods is in the list of intersections to be removed. The list for removing in the most driving routes first backward removing method is the same as the list in the most driving routes first RSU allocation method.

3) The least driving routes first backward removing method: The least driving routes first backward removing method works similar as the previous two methods. The list for removing in the least driving routes first backward removing method is the reverse list used in the most driving routes first backward removing method.

V. Simulation Study

Simulations are performed to study the performances of the proposed RSU allocation methods. First of all, the percentage of city maps in which the method proposed in [4] can apply successfully are discussed. The performances of the proposed methods are compared with each other. In addition, the locations found by the most driving routes first method are also investigated. We also study the the average driving time in each segment with different length of the certificate valid intervals. Finally, the performance of the three proposed backward removing methods are compared with the allocation method without backward removing method.

A. Simulation Model

Random city maps are used to represent the networks. The type of city maps is a square with 10 intersections each side. That is, the number of intersections in a city is 100. For each intersection i to a neighborhood intersection j, the driving time e_{ij} is selected from 20 to 100 with uniform distribution. The driving time from intersection i to j is not necessarily the same as that from intersection j to i. The number of origination-destination intersection pairs is 100×99.

For each simulation run, 100 random city maps are generated. Since most navigation systems use shortest path algorithm as the route planning algorithm [13], [14], the driving route between each source-destination pair is obtained by Dijkstra's shortest path algorithm [15] in the simulations. The length of certificate valid interval C_T ranges from 100 to 800.

For each city map, the numbers of required RSUs obtained by the allocation methods are first calculated. We also calculate the average length of segments to compare with the length of certificate valid interval C_T. Each data point in our graph is the average values over the 100 city maps.

In the figures to be presented in the following, the simulation results corresponding the most driving routes first method are labeled as *Driving Routes*. The most satisfied intersection

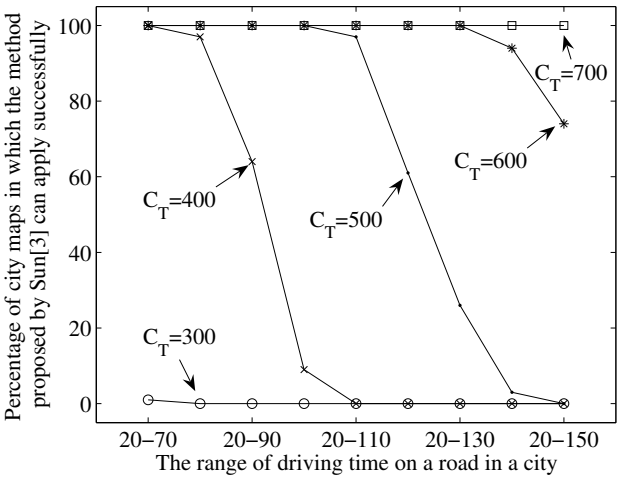

Fig. 3: Percentage of of city maps in which the method proposed in [4] for allocating RSUs such that the certificate can be updated before it expires

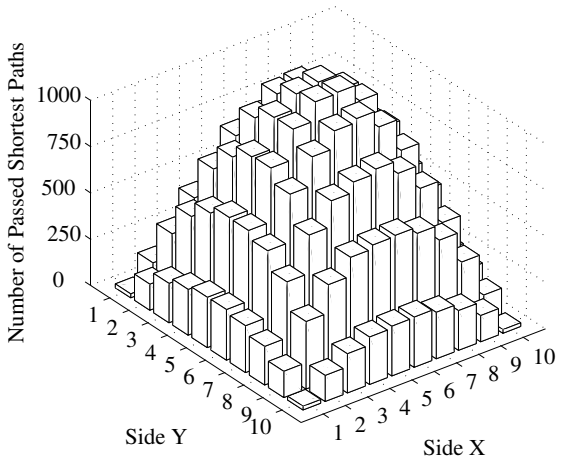

Fig. 4: The number of driving routes passed

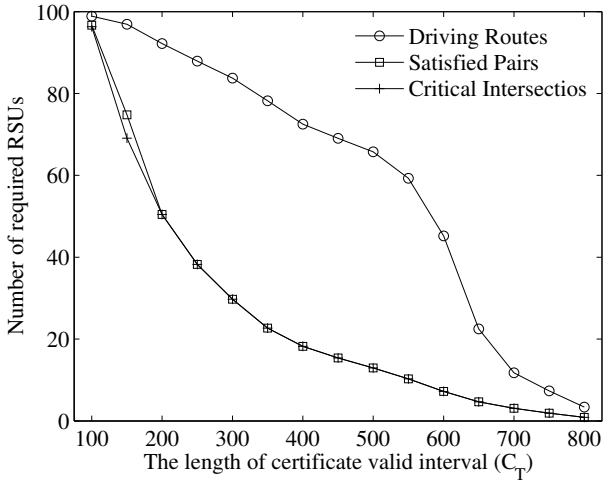

Fig. 5: The number of required RSUs in the city

pairs first method and the critical intersections first methods are labeled as *Satisfied Pairs* and *Critical Intersections* respectively. In the figures 7 and 8 , the allocation methods without backward removing methods are labeled as *No BR*. The three backward removing methods, the random backward removing method, the most driving routes first backward removing method, and the least driving routes first backward removing method are labeled as *Random*, *Most Driving Routes*, and *Least Driving Routes* respectively.

B. Simulation Results

First of all, we will show that the proposed methods are able to solve the RSU allocation problem in a large city or small certificate update interval compare with the proposed method in [4]. Figure 3 shows the percentage of the city maps in which the methods can successfully allocate the RSUs such that the certificate can be updated between all origination-destination pair among 100 city maps. From the figure, we can observe that the proposed methods achieves 100% success rate while success rate of the method in [4] decreased with the increasing certificate update interval.

Next, we are interested in some properties of the most driving routes first method performs when the driving routes are shortest paths between the source-destination pairs. The intersections are located in a square with 10×10 intersections. Fig. 4 shows the average number of shortest paths passed at each intersection from 100 city maps. From the figure, we found the following properties of the most driving routes first method:

- The intersections with minimum number of passed driving routes are located at the corners of the city. This is because if an intersection is at the corners or nearby, the number of shortest paths in the city passed the intersection is small.
- The intersections with maximum number of passed driving routes are located at the center of the city. When an intersection is located at the center of the city, many driving routes may traversed the intersection since the probability that the intersection is on the shortest path between a source-destination pair is high.
- The numbers of passed shortest paths of the nearby intersections are close. When an intersection is around the center of the city, the number of passed driving routes will be large. The value will be small when the intersection is near the boundary of the city.

The properties observed in fig. 4 are able to provide some explanations for the following simulation results.

We next compare the number of required RSUs obtained by the proposed methods. Fig. 5 shows the number of required RSUs with respect to different length of certificate valid intervals. From the figures, we can make the following observations:

- The numbers of required RSUs obtained by the three

methods decreased when the length of certificate valid interval increased. This result is trivial since that when the length of certificate valid interval is large, the need for certificate update will be less as well as the number of required RSUs in the city is small too.

- The number of RSUs obtained by the most driving routes is much larger than the other two methods. This is because that locations of the intersections with large number of passed driving routes are close. When the intersections are close, allocating the RSUs at the these locations will not be effective for certificate update. It is expected that allocating the RSUs more evenly in the city will yield lower number of required RSUs.
- The number of RSUs obtained by the most satisfied intersection pairs first method and the critical intersections first method are the same except when the length of certificate valid interval is smaller than 200. This is because that the maximum driving time on a road is set to 100 in this simulation such that it is impossible to find any critical intersection when C_T is set to larger than or equal to 200. When C_T is larger than or equal to 200, the critical intersections first method is equivalent to the most satisfied intersection pairs method because the critical intersections first method will not find any critical intersection in the first part of the method.
- When the certificate valid interval is smaller than 200, the critical intersections first method yields lower number of required RSUs than the most satisfied intersection pairs first method. Since the most satisfied intersection pairs first method does not find the critical intersections in the beginning, the result implies that some intersections which are able to maximize the number of satisfied intersection pairs but not a critical intersection will be found before some critical intersections. However, when the intersections are found, some of the non-critical intersections may be redundant.
- When the number of required RSUs increased, it is obvious that the average number of segments will decrease.

We are also interested in the average driving time on each segment compared with the length of certificate valid interval. Fig. 6 shows the results of the proposed methods. From the figure, we can made the following observations:

- The average driving time on each segment from the most driving routes first method is less than the other two methods. Since the number of allocated RSUs is large when the most driving routes first method is employed, the number of segment is also large which made the average driving time on each segment be small.
- The average driving time on each segment is much less than the length of certificate valid interval. Given a length of certificate valid interval C_T, it is expected that the driving time on all segments on all driving routes is less than C_T. Since all the driving time on a segment must be less than C_T, some RSUs are allocated for certificate update in a small number of driving routes. However, when the RSUs are allocated, the number of segments on all driving routes which go through the intersection will increase which further decreased the average driving time on each segment.

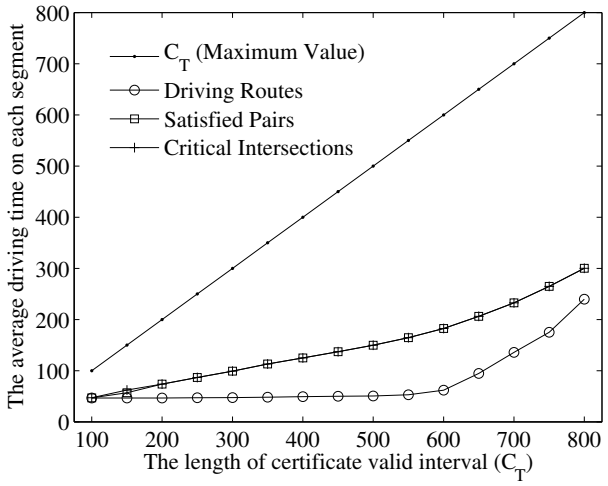

Fig. 6: The average driving time on each segment of driving route

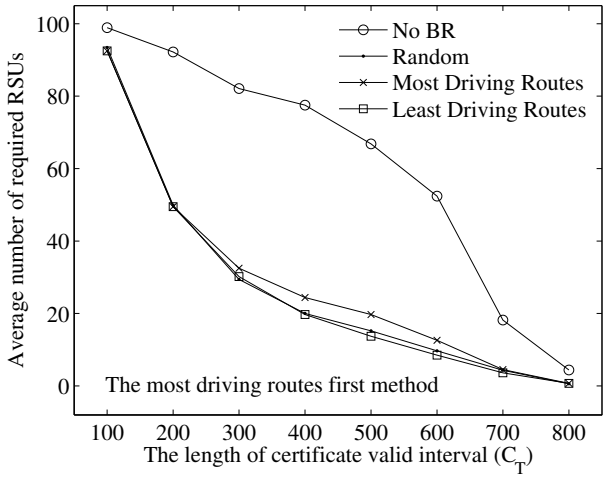

Fig. 7: The average number of required RSUs when the most driving routes first RSU allocation method is employed

Finally, we will discuss the performances of backward removing methods when the placement methods are the most driving routes first method and the most satisfied intersection pairs first method. The performances with three backward removing methods are compared with that without backward removing. Figures 7 and 8 show the average required RSUs when the most driving routes first method and the most satisfied intersection pairs first method are employed with the three backward removing methods.

From the figures, we can make the following observations:

- In Figure 7, the backward removing methods signifi-

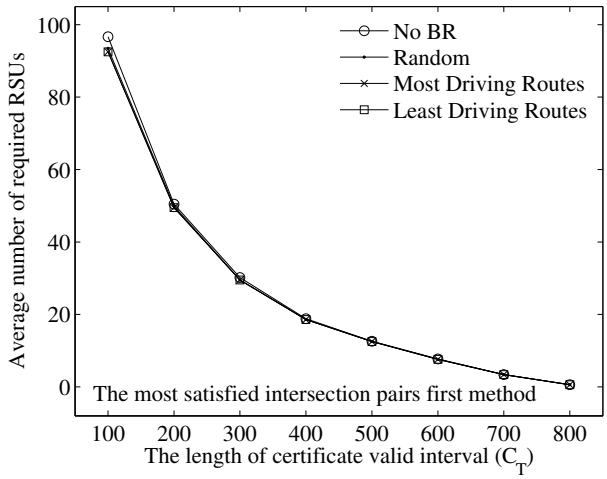

Fig. 8: The average number of required RSUs when the most satisfied intersection pairs first RSU allocation method is employed

cantly reduce the required RSUs compared with that without backward removing methods. This is because the most driving routes first RSUs allocation allocation method find too many redundant intersections for certificate update.

- In Figure 8, the performances of the proposed backward removing methods is similar as that without backward removing. The reason is that the most satisfied intersection pairs method finds the intersections which is indeed required and cannot be removed.
- Among the three backward removing methods, the least driving routes first method performs slightly better than other two methods.

VI. Conclusions

In this paper, we consider the roadside units allocation problem such that the certificates can be updated before it expired. The decision problem of the RSU allocation problem is shown as an NP-complete problem. We also proposed three RSU allocation algorithms which works for a large city. Simulation results show that one proposed method named the critical intersections first method yields lower number of required RSUs than the other two RSUs allocation method. We also show that a backward removing method named the least driving routes first method performs better than the other two backward methods if the RSU allocation method does not find good locations for roadside units. If the route planning algorithm rather than the shortest paths can be developed, the required number of RSUs can be further reduced. Developing the routing planning algorithms is left for the future researches.

Acknowledgements

This research was supported by the National Science Council, Taiwan, under grant NSC99-2218-E-431-001-MY3 .

References

[1] S.W. Wang & M.Y. Chang(2010). Roadside Units Allocation Algorithms for Certificate Update in VANET Environments. In *the 17th Asia-Pacific Conference on Communications (APCC 2011)*, Kota Kinabalu, Malaysia, October 2-5, 2011

[2] J. Zhao & G. Cao(2006). VADD: vehicle-assisted data delivery in vehicular ad hoc networks. In *IEEE INFOCOM 2006,* Barcelona, Spain.

[3] H. Hartenstein & K.P. Laberteaux(2010). *VANET: vehicular applications and inter-networking technologies,* John Wiley & Sons Inc.

[4] Y. Sun, X. Lin, R. Lu, X. Shen, & J. Su(2010). Roadside units deployment for efficient short-time certificate updating in VANETs. In *IEEE International Conferences on Communications (ICC) 2010,* Cape Town, South Africa.

[5] J.L. Huang, L.Y. Yeh, and H.Y. Chien(2011). ABAKA: an anonymous batch authenticated and key agreement scheme for value-added services in vehicular ad hoc networks. In *IEEE Transactions on Vehicular Technology,* 60(1), 248-262.

[6] S. Mahajan & A. Jindal(2010). Security and privacy in VANET to reduce authentication overhead for rapid roaming networks. In *International Journal of Computer Applications,* 1(20), 17-21.

[7] B. Aslam & C.C. Zou (2009). Distributed certificate architecture for VANETs. In *Proceedings of ACM SIGCOMM 2009,* , Barcelona, Spain.

[8] F. Dtzer(2006) Privacy issues in vehicular ad hoc networks . In *Privacy Enhancing Technologies,* Lecture Notes in Computer Science, 2006.

[9] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, & R. Housley, W. Polk (2008), Internet X.509 public key infrastructure certificate and certificate revocation list (CRL) profile. In *IETF RFC 5280,* May 2008.

[10] J. Lee & C. Kim(2010). A roadside unit placement scheme for vehicular telematics networks. In *AST/UCMA/ISA/ACN 2010*, Miyazaki, Japan.

[11] A. Abdrabou & W. Zhuang (2011). Probabilistic delay control and road side unit placement for vehicular ad hoc networks with disrupted connectivity. In *IEEE Journal on Selected Areas in Communications,* 29(1), 129-139.

[12] M.R. Garey & D.S. Johnson(1979). *Computers and intractability: a guide to the theory of NP-completeness,* Bell Laboratories, Murray Hill, New Jersey.

[13] I. Flinsenberg(2009). In *Route planning algorithms for car navigation,* VDM Verlag, September 2009.

[14] C.C. Martin, P.R. Thrift, & M.C. Lineberry(1994). Systems and methods for planning the scheduling travel routes. U.S. Patent 5 272 638, December 21, 1994.

[15] E.W. Dijkstra(1959). A note on two problems in connexion with graphs. in *Numerische Mathematik 1*, 269-271.