A Framework for Implementing Mobile Cloud Services in VANETs

Khaleel Mershad and Hassan Artail
Electrical and Computer Engineering Department
American University of Beirut, Beirut, Lebanon
e-mails: {kwm03, hartail}@aub.edu.lb

Abstract—**The strong potentials that exist in Vehicular Ad hoc Networks (VANETs) has given birth to the concept of Vehicular Clouds, in which cloud computing services are hosted by vehicles that have sufficient resources to act as mobile cloud servers. In this paper, we propose a protocol that enables vehicles to search for mobile cloud servers that are moving nearby and discover their services and resources. The design depends on roadside units (RSUs) that act as cloud directories with which mobile cloud servers register. We evaluate the performance of our protocol using ns2 and explain through comparing the results to another scheme the feasibility and efficiency of our protocol in terms of service discovery and service consuming delays and packet success ratio.**

Keywords-VANETs; Cloud Computing; Cloud services; service discovery; Vehicular Clouds; roadside units

I. INTRODUCTION

The advancement and wide deployment of wireless communication systems have transformed human lifestyles. Researchers have abstracted the idea of Vehicular Ad hoc Networks (VANETs), in which vehicles, equipped with wireless devices, communicate for safety and luxury purposes. VANETs allow vehicles to connect to roadside units (RSUs), which are fixed infrastructure equipped with powerful computing devices. RSUs connect with vehicles via wireless communications and with each other via a wired network. It is expected that future vehicles will be equipped with advanced resources such as powerful computing and storage devices, and sensor nodes [1]. Such vehicles will become powerful computing machines roaming the streets. This fact motivates the idea of making use of these vehicles' resources in a cloud computing environment that exploits the capabilities of such vehicles as mobile cloud servers.

The basic idea behind cloud computing is to allow users to make use of the idle resources that reside on powerful network servers. Businesses may rent the infrastructure and sometimes the needed software to run their applications [2]. In VANETs, Vehicular Clouds are distinctive in that cloud servers are mobile vehicles that have high resources and/or Internet access capabilities, whereas consumers are vehicles that desire to rent these resources or gain Internet access. To achieve this, consumers need to discover the mobile cloud servers, know their resources, and to communicate and request resources from them. In this paper, we propose a protocol that exploits RSUs as cloud directories which store information about mobile cloud servers. We give a mobile cloud server the name of tranSporTAtion seRver, or STAR. An RSU will store for each STAR the type of resources it offers, the attributes of each resource, and the price per resource unit. RSUs will constitute a distributed dynamic index of STARs. We call our protocol disCoveRing and cOnsuming services WithiN vehicular clouds or CROWN.

Note that vehicles might be able to gain Internet access via RSUs. However, connecting via STARs would be preferred for the following reasons: First, a STAR will offer higher Quality of Service (QoS). Suppose that 50 vehicles exist within an RSU coverage area, then the RSU will experience huge load if all 50 vehicles try to connect to the Internet at the same time. However, if 5 STARs exist among the 50 vehicles, then the 5 STARs will cooperate to serve the remaining 45 vehicles which will give each vehicle higher QoS. Second, since the nearest RSU R_1 might be sometimes far from a vehicle V_1, the delay of exchanging packets between V_1 and R_1 such that V_1 could connect to the Internet via R_1 will be high. In such cases, V_1 will prefer to connect via a nearby STAR. Finally, RSUs might not be installed in many areas such as on some highways.

The rest of the paper is organized as follows: A brief overview of cloud computing in VANETs is presented in Section II. The detailed description of CROWN is in Section III, while Section IV evaluates our approach via simulations. Finally, concluding remarks are presented in Section V.

II. BACKGROUND

Cloud computing is increasingly becoming a desirable and foundational element in international enterprise computing. There are many companies which design, develop, and offer cloud technologies. Cloud computing started from the realization that instead of buying and installing necessary software/hardware, businesses may rent the needed resources to run their applications [2]. Moreover, cloud computing can save considerable time since businesses will not have to install or upgrade applications [3].

In VANETs, the idea of making use of smart vehicles as cloud servers in a mobile cloud computing environment was proposed in [1] assuming that these vehicles have Internet access and will feature on-board computational, storage, and sensing capabilities. Owners of vehicles may rent out their vehicles' capabilities on demand, or per-period basis. Also, public vehicles, such as taxies and busses, may benefit from putting their vehicles' computing resources into business. The work in [2] extends that in [1] by providing the possible architectural designs and the resource allocation and load balancing solutions for Vehicular Clouds. The basic feature that distinguishes vehicular clouds is mobility. Most vehicles spend a large amount of time on the road and are involved in several dynamically changing situations [2]. The mobility of vehicles might be thought of as a factor that reduces their ability to serve as cloud servers, since they contact other vehicles for very short periods of time. Hence, they do not have time to exchange sufficient data as required by the cloud. This fact affected vehicular cloud designs ([1-2]), and made them use smart vehicles as cloud servers, only when they are stationary (e.g., when parked, or in a traffic jam).

In this paper, we extend the concept of vehicular clouds by enabling STARs to offer their various resources via the RSU network. RSUSs have an extended transmission range and can relay packets between cloud consumer vehicles and STARs. To the best of our knowledge, we are the first to define the concept of a STAR that offers its cloud services while it is on the move. Our contributions also include identifying the services that could be offered by a STAR and their attributes. Fig. 1 shows a general overview of CROWN.

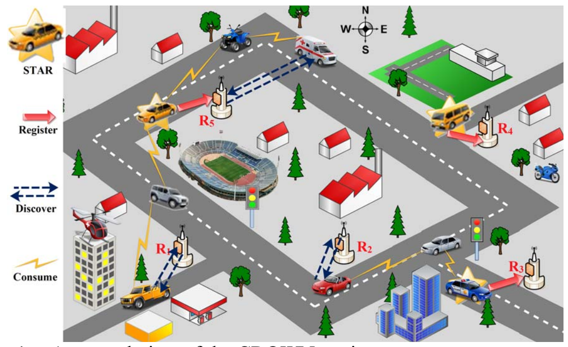


Figure 1. A general view of the CROWN environment.

Note that there are many scenarios in which the existence of a STAR is very important. Consider for example the case of a social network in which vehicles form different groups. Members of each group exchange private information and members of different groups cooperate to trade different traffic data and sensor readings of their vehicles. In previous designs of vehicular social networks (like the one in [4]), each vehicle frequently sends certain data (e.g., traffic status) to members of other groups. In such cases, STARs could serve as social network servers that securely save the exchanged data between different members. The members could use data mining to infer very important information they need from the data that the STAR will save for a certain period of time (e.g., 1 day). Another application for STARs is real-time monitoring. For example, in Healthcare monitoring service, special sensors frequently measure and report the different health aspects of a driver. In such application, the freshness of data is very important. Hence, a STAR that is near to a vehicle being monitored (V_m) plays an important role in receiving the data generated by V_m, analyzing them using applications that run as part of the STAR's cloud services, and taking relevant actions as fast as possible (for example, broadcasting a message to all nearby vehicles to alert them of the state of V_m, sending a fast signal to the OBU in V_m to stop it and then moving fast towards V_m to take a certain action, etc.). In addition to these scenarios, many users in their vehicles might prefer consuming the resources of STARs that are nearby and quickly respond to them than connecting to the Internet to consume online resources which may take a considerable amount of time.

III. THE CROWN PROTOCOL

The deployment of Vehicular Networks is anticipated to start in the near future. It is expected that DSRC-enabled cars will have varying levels of resources, with some having extra storage and processing capabilities that may want to rent out to other vehicles that need additional resources. Also, vehicles that are roaming the streets, like taxies and buses, might welcome the idea of becoming STARs in the vehicular cloud. They may need however to install additional resources in order to engage in the vehicular cloud market.

A. *Vehicular Cloud Services*

There are three fundamental types of computing services that can be made possible by vehicular clouds. We describe here these services that become the focus of this paper, while pointing out that there may be other services that could be added later according to consumers' needs:

1. Network as a Service or NaaS (Internet access): some smart vehicles will have an Internet connection (e.g., via a cellular network), but others will not. This opens the door for vehicles with internet access to offer their extra bandwidth for a certain fee.
2. Storage as a service or STaaS (virtual network hard-disk): some smart vehicles will have high on-board storage capacity, but others may need additional storage. This case can occur if several users are using a vehicle's hardware at the same time, or if the users of a vehicle prefer to keep a backup copy of their data on an external repository. The consumer and the STAR can negotiate for the period of renting the storage capacity and the price per storage unit.
3. Data as a Service or DaaS (virtual data provider): a user in a smart vehicle may require specific data: for example, a video file, a city map, latest news, road conditions, etc. A STAR may define part of its storage as a data cache, and use it to store data that it acquires for consumers. Besides requesting data on the spot, a user can specify to the STAR his daily interests, thus allowing the STAR to acquire these interests and cache them until the consumer requests them. It is expected that STARs which offer DaaS will charge consumers based on the data size.

B. *Registration of a STAR at its Nearest RSU*

When a smart vehicle decides to participate as a STAR in the vehicular cloud, it defines the resources that it can offer and the attributes of each resource. If it can offer Internet access (NaaS), the STAR should define its Quality of Service (QoS) parameters, for example, its bandwidth (BW) and its access delay (D_a). The STAR also sets the price it will charge from its consumers, for example, the access cost per hour (C_a). If the STAR can offer STaaS, it defines the maximum amount of storage per consumer it can offer (S_m), and the times during which its resources will be available

(T_{STAR}) (which is an array that contains the times of each day during which the STAR is expected to connect to the VANET). In addition, the STAR defines the maximum overall time during which it can lend its storage capacity to a consumer (T_t) and the price per storage unit (C_s). Finally, if it can supply DaaS, the STAR defines the types of data it can cache (*DT*), the maximum storage capacity per user it can offer (D_c), and the cost per data unit (C_d). Table I summarizes the attributes of each resource.

TABLE I. ATTRIBUTES OF EACH OF THE THREE RESOURCES THAT CAN BE OFFERED BY A STAR

Parameter	Definition	Type
	NaaS	
BW	access bandwidth	double (kbits/sec)
D_a	access delay	double (msec)
C_a	access cost per hour	double ($/h)
	STaaS	
T_{STAR}	times during which the STAR will connect to the VANET	array of strings (time format)
S_m	maximum storage per customer	double (Mb)
T_t	maximum overall storage time	double (days)
C_s	cost per storage unit	double ($/Mb)
	DaaS	
DT	types of data the STAR caches	array of strings
D_c	maximum data capacity per user	double (Mb)
C_d	cost per data unit	double ($/Mb)

After the STAR defines its resources, it formulates a registration packet that contains the resources it offers and the attributes of each resource and sends it to its nearest RSU (R_1). The registration packet also contains the geographic coordinates of the STAR, and the expected time at which it will leave the VANET (T_l). When R_1 receives the registration packet from the STAR, it stores its location, the resources it can offer, the attributes of each resource, and T_l. R_1 then defines an area which we call the STAR's influence area or A_I, which represents the area in which the STAR can efficiently offer its resources to its consumers. In other words, consumers that are outside A_I could experience degraded results (e.g., long delays) if they try to consume the STAR's resources. A_I will be a circle centered at the STAR, and whose radius can be set either to a constant value that is agreed on by all RSUs or to a variable that depends on certain conditions (e.g., consumers requiring Internet access with average access delay less than 5 seconds). In general, a message that is sent from a vehicle to another will take more time as the distance between them increases. In [5], we studied the relation between the end-to-end delay and the source-to-destination distance ($D_{S\text{-}D}$). In an experiment we did in [5], 20 RSUs were deployed in a 9×9 km^2 area and 300 vehicles were roaming that area. The delay between the source and the destination reached 5 seconds when the distance between the two is around 4 km. Hence, the radius of A_I could be set to an approximate value of 4 km to ensure acceptable Internet access delay. More studies could relate the value of A_I to other parameters such as the speeds of vehicles. We plan to continue on this issue as a future work.

Next, R_1 calculates, using the digital map, the IDs of all RSUs that are within A_I and sends the registration data (*RD*) of the STAR to these RSUs. Each RSU adds the *RD* to its cache, thus allowing the STAR to be known to all RSUs that are within its A_I. Note that A_I will be calculated by the RSU only when it receives the *RD* from the STAR. As the STAR moves and becomes much closer to another RSU R_2 (e.g., by a factor of 30%), it sends its *RD* to R_2, which in turn calculates a new A_I and sends the *RD* to all RSUs within A_I. All RSUs that receive the *RD* for the first time will add the data to their cache. Other RSUs that already cache the *RD* will update the data if the attributes of the STAR changed.

Fig. 2 shows an example of a STAR and its A_I. The STAR (police car) connects at instant t_1 to the RSU R_1, which in turn calculates A_I of the STAR (right dotted circle: $A_{I(1)}$). The RSUs that are within $A_{I(1)}$ are R_2 and R_4. Hence, R_1 sends the *RD* to R_2 and R_4 (light-blue arrows). When the STAR moves, it becomes nearer to R_3 at instant t_2, and hence the STAR sends to R_3 its *RD*. Next, R_3 calculates the new A_I of the STAR ($A_{I(2)}$), which is shown as the left dotted circle. The RSUs which are within $A_{I(2)}$ (i.e., R_1 and R_4) will be getting the updated *RD* of the STAR from R_3.

The STAR periodically sends to its nearest RSU (for example, every 2 sec) a beacon that contains the geographic coordinates of the STAR. The beacon duration depends on the average load in the network. The beacons will let the RSU track the STAR's movement, but since the beacons might experience delays to reach the RSU, the latter will not have precise knowledge about the STAR's location. For this, the RSU will calculate each time it needs to know the location of the STAR an area called estimated STAR area, or A_E. The center of A_E will be the coordinates of the STAR from the last beacon, while the radius of A_E will be the maximum distance d_s that the STAR could have moved between the instance at which it sent the beacon (t_1) and the current time (t_2). If the average speed of the STAR is v, then the radius of A_E will be set to $r_E = 1.3 \times v \times (t_2 - t_1)$, considering a 30% error factor. The area A_E will be used by the RSU to define to the consumers the STAR's estimated location.

As a STAR rents or consumes some of its resources, its attributes will change. In such cases, the STAR adds the new values of its attributes which changed to the next beacon that it sends to its nearest RSU R_1, which in turn updates the values of these attributes in its cache and forwards the data to other RSUs within A_I that update their cached data. An RSU that stops receiving beacons from a STAR for a given time threshold τ (e.g., $\tau = 10 \times$ beacon interval) will delete its *RD*. If an RSU receives requests that are targeted to a STAR that it deleted its RD, it sends these requests to its neighbors, and so on until the requests reach an RSU R_I that is within the A_I of the STAR. Hence, R_I will send the requests to the STAR.

C. *Requesting to Rent the STAR Resources*

When a user in a smart vehicle needs to connect to the Internet, requires additional resources that his vehicle does not have, or is interested in certain data, he can exploit the services of one or more nearby STARs. First, he formulates a request packet and sends it to his nearest RSU. In addition to the desired resources and their attributes, the request packet contains the geographic coordinates of the user's vehicle.

If the user is searching for a STAR that can grant him Internet access, he will need to specify the minimum NaaS attributes that he desires, such as the channel bandwidth (BW), access delay (D_a), access duration (T_a), maximum distance to the STAR ($D_{to\text{-}STAR}$), and maximum cost per hour (C_a) the user is willing to pay. If it is storage resources that are desired, the user defines the storage capacity he needs (S_m), the total storage time (T_t), the maximum storage cost (C_s), and an array that contains the times of each day of the week during which he requires the storage (T_{user}). Finally, if specific data are needed, the user defines the data type or the data identifier (DT), e.g., if the user needs an e-book, he specifies the book name and its author. If he wants certain type of news, he states the type, the time, and the source. Similarly, the user also specifies the maximum cost per unit of data (C_d), and the maximum size of the data he requires (D_c). Note that some attributes are fixed for each resource, while others might be changed according to the requirements, including adding to the DT attribute fields that are specific to the search criteria. Table II summarizes the attributes of each resource included in the request packet.

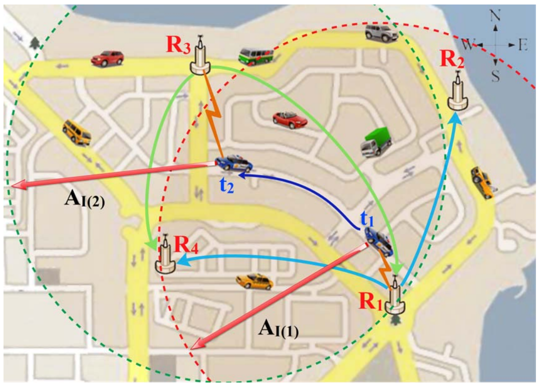


Figure 2. An example of how the Influence Area A_I of a STAR changes as the STAR connects to a new RSU.

TABLE II. ATTRIBUTES SPECIFIED BY THE USER FOR EACH OF THE THREE CLOUD RESOURCES

Parameter	Definition	Type
	NaaS	
BW	minimum access bandwidth	double (kbits/sec)
D_a	maximum access delay	double (msec)
C_a	maximum access cost per hour	double ($/h)
T_a	maximum access duration	double (hours)
$D_{to\text{-}STAR}$	maximum distance to the STAR	double (m)
	STaaS	
T_{user}	times during which the user will require the storage of the STAR	array of strings (time format)
S_m	maximum required storage capacity	double (Mb)
T_t	maximum overall storage time	double (days)
C_s	maximum cost per storage unit	double ($/Mb)
	DaaS	
DT	types (or name and details) of data the user requires	array of strings
D_c	maximum data capacity required	double (Mb)
C_d	maximum cost per data unit	double ($/Mb)

D. Finding STARs that Satisfy a Request

When an RSU receives a request packet from a user, it searches in its cache for one or more STARs that could satisfy the user requirements. If Internet access is being demanded, the RSU looks for STARs that offer NaaS, have a bandwidth greater than or equal to what the user specified, have an access delay that is below the specified, will remain in the VANET for a period greater than T_a, and demand prices that are below the one specified by the user. The RSU calculates the estimated distance between the center of A_E and the location of the user (from the request packet) and excludes STARs whose distance to the user is greater than $D_{to\text{-}STAR}$. In order to know whether the STAR will remain in the VANET for a period of time greater than T_a, the RSU adds T_a to the current time and checks whether the result is less than T_l. Equation (1) shows the condition the STAR should satisfy to meet the user's Internet access requisites:

$$\begin{aligned} Condition_{NaaS} = {} & (BW_{STAR} > BW_{user})\,\text{AND}\,(D_{a-STAR} < D_{a-user}) \\ & \text{AND}\,(C_{a-STAR} < C_{a-user}) \\ & \text{AND}\,(\text{Distance}_{STAR-user} < D_{to-STAR}) \\ & \text{AND}\,(T_{current} + T_a < T_l) \end{aligned} \tag{1}$$

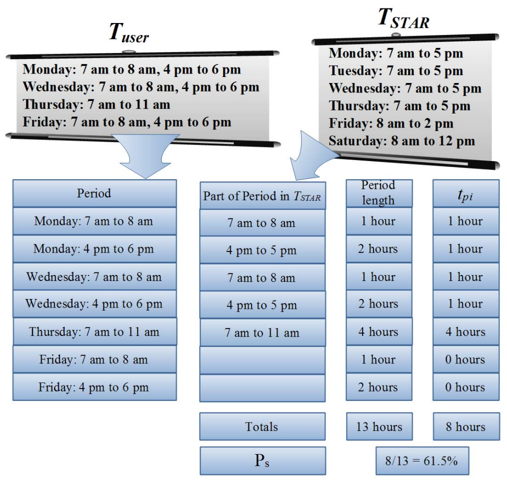
If the user requires storage capacity, the RSU looks for STARs that offer STaaS, have a storage capacity per consumer greater than that required by the user, provide a total storage time greater than the total storage time wanted by the user, and charge a rate less than what is specified. Finally, the RSU calculates the percentage of time during which the user can use the storage of the STAR (P_s), as follows: for each time period p_i in the array T_{user}, the RSU calculates a time variable t_{pi}, which is the portion of p_i that is included in the times of the array T_{STAR}. The RSU then adds t_{pi}'s of all periods in T_{user} and divides the result by the sum of all p_i's (i.e., total time of T_{user}) to get P_s. The RSU chooses among the STARs that satisfy the first three conditions, the ones that have a P_s greater than an agreed-on threshold Th_{Ps} (for example, 70%), to ensure that the selected STARs will serve the user for acceptable times during his connections to the VANET. Fig. 3 shows samples of T_{STAR}, T_{user}, how the RSU calculates t_{pi} for each period in T_{user}, and also how P_s is calculated. The total condition that the STAR should satisfy to meet the user storage requirement is:

$$\begin{aligned} Condition_{SaaS} = {} & (S_{m-STAR} > S_{m-user})\,\text{AND}\,(T_{t-STAR} > T_{t-user}) \\ & \text{AND}\,(C_{s-STAR} < C_{s-user}) \\ & \text{AND}\,(\text{P}_s \geq Th_{Ps}) \end{aligned} \tag{2}$$

If the user requires certain data, the RSU looks for STARs that offer DaaS. The first condition that the STAR should satisfy is its willingness to cache data types (DT_{STAR}) that are similar to those requested by the user (DT_{user}). To determine this, the RSUs will store a comprehensive list of data types and keywords, which could be related to each other. For this purpose, RSUs can make use of a mechanism that deals with the relationships between keywords (a similar system was presented in [6]). Next, the RSU looks for STARs that meet the size and cost criteria. Hence, the condition that a STAR must satisfy to fulfill a data request is:

$$\begin{aligned} Condition_{DaaS} = {} & (DT_{user} \subseteq DT_{STAR})\,\text{AND} \\ & (D_{c-STAR} > D_{c-user})\,\text{AND}\,(C_{d-STAR} < C_{d-user}) \end{aligned} \tag{3}$$

The RSU searches its records for STARs that satisfy one of the above conditions as follows: first, it examines whether the STAR offers the service it is looking for, and then checks if the relevant condition is met. The RSU adds the IDs of the STARs satisfying the above to a candidate list L_c. If the RSU does not find a candidate STAR, it searches again looking for STARs that offer the requested service but with attributes that are similar to those requested. Here, similarity could mean that the difference between what is offered and what is desired is less than a certain threshold (e.g., 20%). The RSU could increase this threshold until it finds at least one STAR.



Period	Part of Period in T_{STAR}	Period length	t_{pi}
Monday: 7 am to 8 am	7 am to 8 am	1 hour	1 hour
Monday: 4 pm to 6 pm	4 pm to 5 pm	2 hours	1 hour
Wednesday: 7 am to 8 am	7 am to 8 am	1 hour	1 hour
Wednesday: 4 pm to 6 pm	4 pm to 5 pm	2 hours	1 hour
Thursday: 7 am to 11 am	7 am to 11 am	4 hours	4 hours
Friday: 7 am to 8 am		1 hour	0 hours
Friday: 4 pm to 6 pm		2 hours	0 hours
	Totals	13 hours	8 hours
	P_s	8/13 = 61.5%	

Figure 3. An example of how the RSU calculates P_s of a user.

Equations (1), (2), and (3) above represent the conditions of NaaS, STaaS, DaaS, respectively. If the user requests multiple services, the RSU first searches for STARs that satisfy the conditions of all requested services. If the RSU does not find any such STARs, it separates the user's requests into single services. For example, if the user requested NaaS and DaaS, the RSU looks for STARs that offer both NaaS and DaaS and satisfy conditions (1) and (3). If the RSU does not find such STAR, it separates the user's requests into a NaaS request and a DaaS request and searches for each request separately. Hence, the RSU will create two candidate lists: $L_{c\text{-}NaaS}$ and $L_{c\text{-}DaaS}$. The user will have to consume the services from multiple STARs.

E. RSU Reply to a User

Whenever a user sends a request packet to the nearest RSU, he specifies what results he wants to receive. For this purpose, the body of the request packet will contain at its beginning a constant-size variable R_e. If the user sets R_e to "All", it indicates to the RSU that the user wants to receive information about all STARs in L_c. Else, if the user is requesting a STAR that offers NaaS, he can set R_e to one of four values: "Access Delay", "Access Cost", "Bandwidth", or "Nearest STAR". For example, if the user sets R_e to "Bandwidth", he indicates to the RSU that he wants to get information about the STAR that offers the highest bandwidth among all STARs in L_c. The same can be said about STaaS and DaaS services. If the user is requesting multiple services, he can specify one or more values of R_e according to his preferences. The RSU will choose the corresponding STAR(s) from L_c based on the value(s) of R_e. Next, the RSU will formulate a reply packet that contains the following for each STAR chosen by the RSU:

- ID of the STAR.
- Last location of the STAR's (from its last beacon).
- Last estimated area A_E of the STAR (center, radius).
- The resources offered by the STAR, and attributes of each resource.

When the requesting vehicle receives the reply packet, it plots on a digital map the locations, estimated areas, resources, and attributes of the STAR(s). In case the user requested all STARs in L_c, he can observe the map to get an idea about the estimated location and attributes of each STAR so he can manually choose one of them to inquire about its resources. If he requested multiple services and needed to select a STAR for each service, he can make his choice based on his own criteria after observing the map. This choice might be desired by many users who prefer to make their selection based on their own judgment. There are users however who would prefer to let the system choose the best STAR for them. In this case, the user could indicate his selection criteria via R_e, and receive the information about the selected STAR. Fig. 4 shows an example of a user requesting NaaS with R_e set to "ALL". The figure shows the RSU replying with all STARs matching the user's query (shown in the figure along with their A_E's and attributes), while other STARs are shown without information.

F. Consuming the STAR's resources

After a user chooses one or more STARs from the reply packet, he specifies the resources (and their attributes) that he requires from each STAR. He then formulates a service packet for each chosen STAR and sends it using the routing protocol. The service packet contains the user credentials and his request. In CROWN, we focus on finding a way to discover and choose the best STARs for consuming services. It should be emphasized that the connection between a user and a STAR should be tightly secured via a mobile security system that insures the privacy and integrity of users and the confidentiality and correctness of data. Examples of such systems can be found in [7-9], which can operate alongside our protocol to secure its various operations.

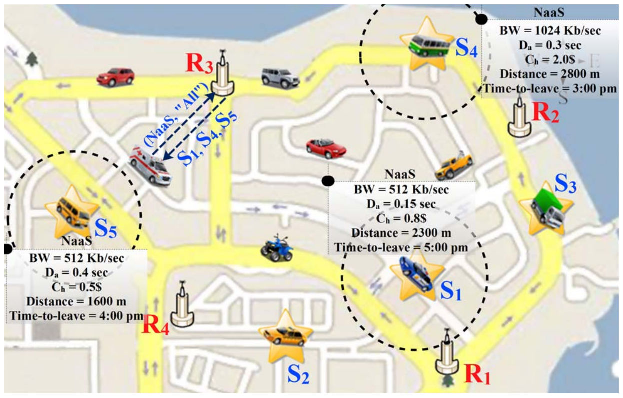


Figure 4. An example of the reply packet sent from the RSU to the user: candidate STARs with their estimated areas and attributes.

When a STAR receives the service packet from the user, it either replies with a negative acknowledgment if it is not able to satisfy the user's request, or with a service-reply packet otherwise, in which case the protocol may prompt the user for the method of payment. Upon replying with an identify packet that contains the necessary information about the payment and the order, the user and the STAR can exchange data packets corresponding to the resources that the user wants to consume.

An important factor in the proposed protocol is the routing strategy that should be used to connect STARs (and users) to nearest RSUs, and to connect users to STARs. For this, we adopt the "CAN DELIVER" routing protocol [5], which we specifically designed for vehicular environments and which exploits RSUs to route packets between different VANET entities. CAN DELIVER operates by using vehicles to carry and forward messages from a source vehicle to a nearby RSU and, if needed, route these messages through the RSU network, and finally send them from an RSU to the destination vehicle. The evaluation of CAN DELIVER in [5] confirmed its advantage when compared to three recent and known routing protocols that were proposed for VANETs. Finally, before moving to the performance evaluation, Fig. 5 summarizes the major operations of the CROWN protocol.

(1) Smart vehicle **S_1** becomes a STAR
(2) **S_1** registers its resources with its nearest RSU **R_1**
(3) **R_1** calculates the influence area **A_I** of **S_1**
(4) **R_1** multicasts the registration data (RD) of **S_1** to all RSUs **R_i** that are within **A_I**
(5) Each RSU **R_i** within **A_I** saves the RD of **S_1**
(6) **S_1** periodically (default 2 seconds) updates its location and resources at **R_1**
(7) A consumer **C_1** sends a request packet to one of the RSUs (**R_2**) within **A_I**
(8) **R_2** sends the ID, RD, and location of **S_1** to **C_1**
(9) **C_1** sends a resource request packet to **S_1**
(10) **S_1** exchanges service agreement information with **C_1**
(11) **S_1** sends reply data or receives storage data from **C_1**

Figure 5. A summary of the operations performed in CROWN between a STAR, its nearest RSU, another RSU within A_I of STAR, and a consumer.

IV. Performance Evaluation

This section presents the results of the simulations that we made to evaluate CROWN, which we implemented using the ns2 software (with the 802.11p amendment and the Nakagami model), and used SUMO [10] to generate the simulations node movement file. The map used to generate the movement file has a size of 9×9 Km^2. The default number of vehicles was set to 300, and their minimum and maximum speeds were set to 15 and 30 m/s. Each simulation scenario was repeated ten times for reliability. Ten RSUs were deployed uniformly across the map to balance their loads. The query model was chosen such that each vehicle sends a new request every 30s. The beacon interval was set to 2s. Other important parameters are shown in Table III.

We designed the simulation setup as follows: the simulation time was divided into two parts. In the first one that lasted for 20 seconds, each vehicle chosen to act as a STAR sends its registration data to its nearest RSU. We picked a certain number *m* out of the vehicles as STARs. One seventh of the STARs were chosen to offer NaaS, and similar proportions to offer STaaS, DaaS, both NaaS and STaaS, both NaaS and DaaS, both STaaS and DaaS, and all three services. The default value of *m* in the simulations was set to 35. However, we also tested some scenarios with another two values of *m*: 70 and 105. We chose random values for the attributes of each service. In the second simulation part, each of the other non-STAR vehicles will choose randomly one to three services (with random attributes and R_e) and sends a request packet to the nearest RSU every 30 seconds. In case R_e was set to "All", we made the requesting vehicle randomly choose one of the STARs from each candidate list. When a vehicle sends a service packet to a STAR, the later replies with a Negative ACK if its queue is full; else, the STAR starts serving the user if its queue is empty or adds the request to its queue if it is busy with other requests. As stated in Table III, the STAR's queue was set to a size of 10 to reduce the requests' serving delay.

TABLE III. Values of some of the simulation parameters

Parameter	Value
Simulation time	2000 sec
Wireless Bandwidth	6 Mbps
Radio transmission range	300 m
Request packet size	50 bytes
Data packet size	[0.1-10] MB
STAR's Queue size	10
Packet TTL in STAR's Queue	20 sec

In order to make the simulation scenarios as realistic as possible, we attached a custom ns2 traffic agent to each vehicle ns2 class. The purpose of this ns2 agent is to use the AODV routing protocol (which contains a special routing table for neighbor nodes) to transmit periodic traffic beacons that contain the location information of the vehicle to its neighbors. In case of an emergency (accident, slippery road, congestion, etc…), the beacon will contain data about the emergency. We set the traffic beacon interval to 1 second.

A. *Compared Protocol and Comparison Parameters*

To assess the performance of CROWN, we compared it with another cloud service discovery protocol in which the various operations of CROWN were replaced by broadcasting. We call this protocol Broadcast-CROWN (B-CROWN). In B-CROWN, a STAR registers its services by broadcasting to its neighbors a registration packet with a TTL equal to 10. Each neighbor caches the RD, decrease the TTL, and rebroadcast the packet to neighbors. When a vehicle *S* needs a certain service, it searches its cache to see if one of the STARs in its cache can fulfill its request. If it finds such STAR, it sends a service packet to it; else, *S* broadcasts a request packet with a unique *reqID* and a TTL equal to *h*. Each vehicle V_I that receives a request packet will search its cache for STARs that can fulfill the request. If V_I finds such STARs, it unicasts a reply packet to *S*, provided it has not forwarded a reply packet to *S* with the same *reqID* (i.e., it did not detect another vehicle replying before it). If V_I does not find such STARs, it decreases the TTL and rebroadcasts the request to its neighbors, if the TTL is greater than 0 and if it did not forward a reply to *S* from another vehicle. When *S* receives one or more reply packets, it chooses the best suitable STARs. In our simulations, we set *h* to a default value of 5. However, we tested some scenarios with another two values of *h*: 10 and 15.

The metrics used for evaluating the two protocols are the:

1. Service Discovery Delay (D_{SD}): time between sending a request and receiving the reply packet.
2. Service Consuming Delay (D_{SC}): time between sending a service packet and exchanging the first data packet.
3. Queuing Delay (D_Q): average time during which a user's request stays in the queue of the STAR.
4. Percentage of Hits (P_H): percentage of requests to which a reply is received.
5. Percentage of NACKs ($P_{N\text{-}ACK}$): percentage of requests received by a STAR to which it replies with a NACK.
6. Vehicle Traffic (T_V): average traffic generated, received, and forwarded by a vehicle.

The above metrics were computed while varying the total number of vehicles (N_v) between 50 and 500, and the user to STAR distance ($D_{STAR\text{-}user}$) between 300 and 3000 meters. The results are shown in Figures 6 and 7. We also tried three values of *m* and *h*, whose results are shown in Fig. 8.

B. *Results and Discussions*

In Figures 6-a and 6-b, we notice that both the discovery and consuming delays of CROWN decrease as N_v increases. This is because the average number of neighbors per vehicle increases as N_v increases, meaning that a vehicle in a route will have a higher probability of forwarding the packet to a neighbor. Hence, packets will be more forwarded and less stored-and-carried (which are the operations used in CAN DELIVER to route packets). Consequently, packets will take smaller times to reach their destinations. As for B-CROWN, although these same operations exist, the discovery and consuming delay of B-CROWN decrease as N_v increases up to 200 vehicles, after which the two delays increase to very high values. This behavior is due to the high congestion that appears in B-CROWN when N_v is high, due to the abundance of broadcasting, where each vehicle broadcasts a service packet to its neighbors, and this operation is repeated until a candidate STAR is found. It is expected that these operations will yield a huge number of packets in the network. In this case, a vehicle storing packets in its queue will not be able to forward all these packets to a neighbor that passes by, due to the small communication time between two neighbors. Hence, many packets will be delayed to reach their destinations. In general, we can deduce that broadcasting reduces the delay when the network is sparse, but leads to congestion when the network is dense.

The congestion in B-CROWN when N_v is higher than 200 can be noticed in Figures 6-c to 6-e. In Fig. 6-c, we notice that both systems have close D_Q's until N_v reaches 200, after which the D_Q of B-CROWN jumps due to congestion. Also, although the P_H of B-CROWN is higher than that of CROWN when N_v is less than 300, it decreases from 98% to 74% due to congestion. Also, $P_{N\text{-}ACK}$ of B-CROWN increases from 0.9% to 46% (Fig. 6-e). As for T_V (Fig. 6-f), each vehicle in B-CROWN will receive and/or

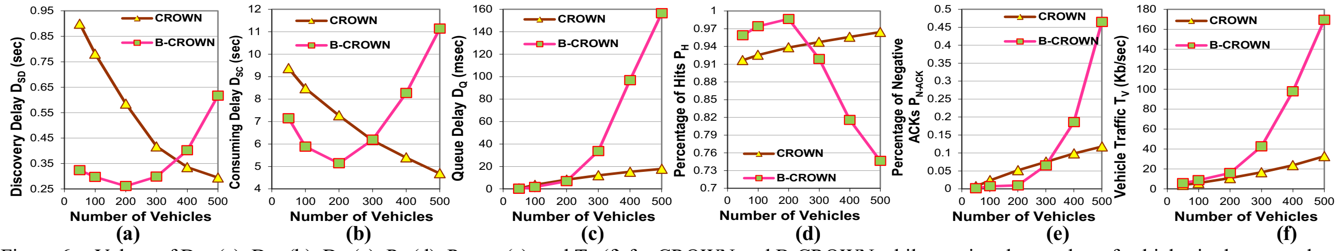


Figure 6. Values of D_{SD} (a), D_{SC} (b), D_Q (c), P_H (d), $P_{N\text{-}ACK}$ (e), and T_V (f) for CROWN and B-CROWN while varying the number of vehicles in the network.

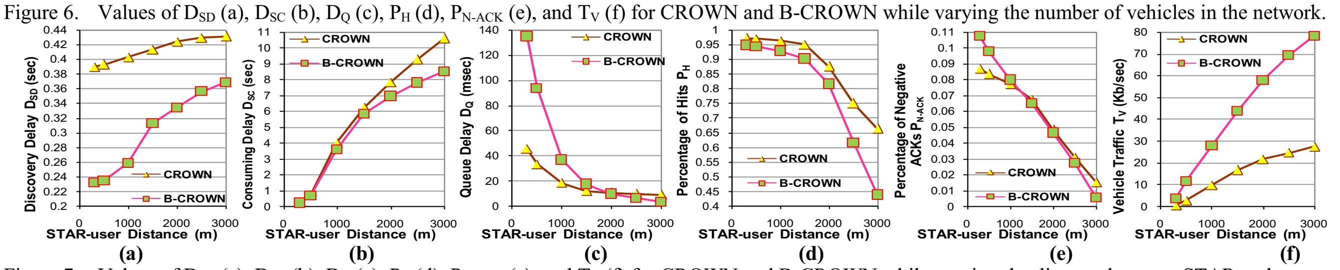


Figure 7. Values of D_{SD} (a), D_{SC} (b), D_Q (c), P_H (d), $P_{N\text{-}ACK}$ (e), and T_V (f) for CROWN and B-CROWN while varying the distance between STAR and user.

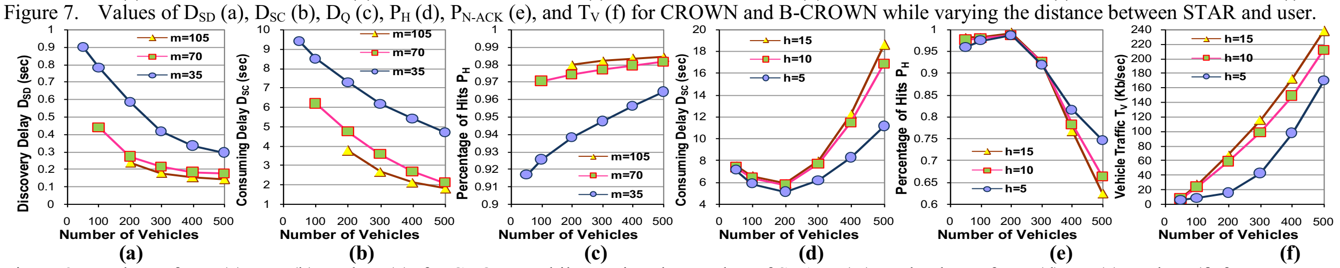


Figure 8. Values of D_{SD} (a), D_{SC} (b), and P_H (c), for CROWN while varying the number of STARs (*m*), and values of D_{SC} (d), P_H (e), and T_V (f) for B-CROWN while varying the number of broadcast hops (*h*).

forward 170 kbits per second when N_v is equal to 500, while the same vehicle in CROWN will receive and/or forward 33 kbits per second. Contrary to B-CROWN, we notice that CROWN experiences stable performance as N_v increases.

An important factor that influences the performances of the two systems is the distance between the user and the STAR that serves him. In order to study this factor, we made RSUs in CROWN send in reply packets information about STARs that are within a specific distance range from users. For example, if we set $D_{STAR\text{-}user}$ to 1km, then an RSU will include in a reply packet STARs that are 950 to 1050 meters away from the user. With respect to B-CROWN, we made users contact STARs that are $D_{STAR\text{-}user}$ away from them.

From Fig. 7-a, we observe that B-CROWN has a smaller discovery delay than CROWN, since in certain conditions broadcasting is much faster than unicasting. As for the consuming delay, the two systems have similar results for small $D_{STAR\text{-}user}$. However, D_{SC} of B-CROWN is smaller than that of CROWN for high $D_{STAR\text{-}user}$, also due to broadcasting which explores all possible options to send a packet in the fastest possible method. However, the small delays of B-CROWN are at the expense of other factors. For example, we see in Fig. 7-c that the queuing delay of B-CROWN is much higher than that of CROWN when $D_{STAR\text{-}user}$ is small. Also, the P_H of B-CROWN decreases to about 40% when $D_{STAR\text{-}user}$ is equal to 3 Km. This is due to the process of RSUs discovering STARs, yielding higher success than broadcasting since RSUs exchange information about STAR which leads to a larger STAR influence area. The two systems have similar $P_{N\text{-}ACK}$ (Fig. 7-e), and finally, the traffic of B-CROWN is about two times that of CROWN when $D_{STAR\text{-}user}$ is 3 Km. Note that T_V is the total traffic per vehicle. From Figures 7 and 8, we notice that T_V has an average value of 35 Kbits/sec. From our simulations, we found that about 17% (6 Kbits/sec) of this traffic is due to traffic beacons, which shows that traffic beacons do not significantly affect the operations of the CROWN protocol.

In Figures 8-a to 8-c, we study the effect of varying the number of STARs (m) on the D_{SD}, D_{SC}, and P_H of CROWN. We observe that as m increases, D_{SD} and D_{SC} decrease and P_H increases. The existence of more STARs means that a vehicle will have a higher probability of finding a STAR that satisfies its request at the nearest RSU. Hence, D_{SD} drops. More STARs means that a vehicle is likely to find a nearer STAR that satisfies its request, causing D_{SC} to decrease.

Finally in this section, we study in Figures 8-d to 8-f the effect of varying the broadcast range (h) on D_{SC}, P_H, and T_V of B-CROWN. We see in Fig. 8-d that D_{SC} increases as h increases, since farther STARs can be discovered when h is increased. However, more congestion will occur, as we can deduce from Fig. 8-f (since packets will be further broadcasted to far distances). This congestion decreases the performance of B-CROWN, as we notice from Fig. 8-e which shows that P_H decreases as h increases (when N_v is high). This reflects the fact that queues of vehicles will be packed with packets due to the high number of broadcasted packets. Consequently, packets will be delayed and/or dropped. In general, the results show that broadcasting is a double-edged sword when used in cloud-oriented VANETs.

V. Conclusion and Future Work

This paper presented CROWN, a novel cloud service discovery protocol for VANETs. In CROWN, STARs were introduced as mobile cloud servers that offer their services to other vehicles. We specified three main service types that can be offered in VANETs, and described how the network of RSUs can be exploited to ease the operation of discovering the best candidate STAR by the user. The results of CROWN were compared to a broadcasting-based protocol and results of some parameters were calculated and discussed. For future work, we will study the required security strategies for the system. Also, we plan on designing and testing a method to compute the value of A_I based on parameters such as the resources of the STAR (transmission range, speed, etc.), the number of RSUs, the road-traffic conditions around the STAR location, etc.

References

[1] S. Olariu, I. Khalil, M. Abuelela, “Taking VANET to the clouds”, *International Journal of Pervasive Computing and Communications*, Vol. 7, No. 1, 2011, pp. 7-21.

[2] S. Olariu, T. Hristov, and G. Yan, “The Next Paradigm Shift: From Vehicular Networks to Vehicular Clouds,” Mobile Ad Hoc Networking: Cutting Edge Directions, Second Edition, Wiley and Sons, New York (2012), pages 645–700.

[3] Z. Alazawi, S. Altowaijri, R. Mehmood, and M. B. Abdljabar, “Intelligent Disaster Management System based on Cloud-enabled Vehicular Networks”, *Proc. of the 11th International Conference on ITS Telecommunications (ITST 2011)*, St. Petersburg, Russia, Aug. 2011.

[4] I. Lequerica, M. Garcia, and P. M. Ruiz, “Drive and Share: Efficient Provisioning of Social Networks in Vehicular Scenarios”, *IEEE Communications Magazine*, Nov. 2010, pp. 90-97.

[5] K. Mershad, H. Artail, and M. Gerla, “We Can Deliver Messages to Far Vehicles”, *IEEE Tran. on Intell. Transp. Systems*, Vol. 13, No. 3, Sep. 2012, pp. 1099-1115.

[6] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, “Privacy-Preserving Multi-keyword Ranked Search over Encrypted Cloud Data”, *Proc. of IEEE INFOCOM 2011*, Shanghai, China, Apr. 2011, pp. 829-837.

[7] W. Jia, H. Zhu, Z. Cao, L. Wei, and X. Lin, “SDSM: A Secure Data Service Mechanism in Mobile Cloud Computing”, *Proc. of the 1st International Workshop on Security in Computers, Networking, and Communications (SCNC)*, Shanghia, China, Apr. 2011, pp. 1060-65.

[8] K. Mershad and H. Artail, “A Framework for Secure and Efficient Data Acquisition in Vehicular Ad Hoc Networks,” *IEEE Tran. on Vehicular Technology*, Vol. 62, No. 2 (2013), pages 536 - 551.

[9] G. Yan, D. Wen, S. Olariu, and M. C. Weigle, “Security Challenges in Vehicular Cloud Computing,” *IEEE Tran. on Intell. Transp. Systems*, Vol. 14, No. 1, Mar. 2013, pp. 284-294.

[10] Centre for Applied Informatics (ZAIK), Institute of Transport Research, German Aerospace Centre, “Sumo - simulation of urban mobility”, http://sumo.sourceforge.net/