Caching-Assisted Access for Vehicular Resources

Sherin Abdelhamid*, Hossam S. Hassanein*, Glen Takahara†, and Hesham Farahat‡
*School of Computing, Queen's University, Kingston, ON, Canada
{sherin, hossam}@cs.queensu.ca
†Department of Mathematics and Statistics, Queen's University, Kingston, ON, Canada
takahara@mast.queensu.ca
‡Department of Electrical and Computer Engineering, Queen's University, Kingston, ON, Canada
h.farahat@queensu.ca

***Abstract*—Smart vehicles are considered major providers of ubiquitous information services. In this paper, we propose a solution for expedited and cost-effective access to vehicular public sensing services. The proposed caching-assisted data delivery (CADD) scheme applies caching on the delivery path of the collected data. Cached information can be used for later interests without having to request similar data from vehicles. CADD relies on the deployment of a "light-weight" road caching spot (RCS) at intersections, and on vehicles for communication to/from RCSs. CADD involves a novel caching mechanism that utilizes real-time information for selecting the caching RCSs while considering popularity in cache replacement. A data chunk to be replaced may be forwarded to another less-loaded RCS. CADD considers vehicles' headings to direct communication towards the destination, which reduces access delay. Performance evaluation of CADD shows significant improvements in the access cost and delay compared to a scheme that does not deploy RCSs.**

***Keywords*—Vehicular resources and data delivery, Caching.**

I. Introduction

Smart vehicles with their abundant on-board resources such as the sensing, computing, data storage, and relaying resources are promising solutions for providing ubiquitous information services [1]. These services are not only confined to other vehicles on the road but also to remote third parties. Vehicle-based services have broad potential for enhancing the public sensing domain. A vehicle on the road can be considered a mobile sensor collecting data on the go, processing such data, and sharing it with others for supporting a wide range of information services [2].

Utilizing vehicular resources for providing sensing-based services faces two major concerns. The resource owners need to get rewarded each time their resources are accessed which brings an access cost challenge. Also, when data is needed from a certain area of interest (AoI), a sensing request needs to be forwarded towards that area which imposes an access delay challenge. In this paper, we present a solution that handles the aforementioned access concerns through applying caching on the delivery path of the service data (i.e., the collected sensing data needed for providing the service).

Our argument is that caching of collected data somewhere on the road helps in resolving later interests in similar data without having to access vehicular resources again. In addition, bringing the data of interest closer to the requesters through caching aids in reducing the access delay. In this paper we propose a caching-assisted data delivery (CADD) scheme that depends on utilizing caching spots deployed on the road for assisting in collecting vehicular data and providing vehicle-based information services. Examples of such services include reporting road and weather conditions to interested authorities, reporting how crowded it is near points of interest, monitoring medium/long-lasting events on the road (e.g. fires), and reporting pollution and noise levels. These supported services can benefit a wide scope of consumers that include municipalities, governmental authorities, news and weather centers, end users, and other vehicles.

Some data delivery schemes that utilize assistance from road-side entities are available in the literature. These schemes depend on utilizing powerful road-side units (RSUs) for forwarding assistance without caching consideration. These RSUs can be adjusted and utilized for supporting the proposed caching concept; however, the cost of ubiquitous deployment will be high since the off-the-shelf RSU models available are not inexpensive. As a solution, we propose a schematic for a simple, light-weight device that can complement RSUs when ubiquitous RSU deployments are not feasible. Having only the components needed for forwarding and caching assistance, our proposed road caching spot (RCS) is lower-priced than an RSU.

As a default caching mechanism in caching-assisted schemes, ubiquitous caching is used for caching every packet everywhere. Apparently, this mechanism is not efficient and other caching mechanisms are proposed to solve its inefficiency [3]. One approach gaining popularity is the centrality-based caching approach [4] which aims at finding a subset of caching nodes that are the most central nodes in the network and directing caching to these nodes as these are the spots that are more likely to get many interests passing by and hence, more cache hits. Unfortunately, most of the caching mechanisms utilizing this approach use static computations for the centrality values as they target networks with static topology such as the Internet backbone. As a vehicular network is dynamic in nature, such mechanisms cannot fit in our scheme. Therefore, we propose a dynamic centrality-based caching mechanism as part of the proposed CADD scheme. Another feature of the proposed mechanism is considering data popularity in cache replacement, favoring data types with more interests from end users. As well, in cache replacement, the proposed mechanism works on giving the to-be-replaced data chunk another caching opportunity instead of dropping it as commonly followed by other caching mechanisms.

In addition to being *caching-assisted*, the proposed data delivery scheme is *heading-aware* in the sense that the vehicles carrying packets to be forwarded check at intersections

978-1-4799-3780-6/14/$31.00 ©2014 IEEE

whether they are heading towards the packet destination or not. If a vehicle finds that it is going far from the destination, it seeks forwarding assistance from the neighboring RCS by offloading the carried packet to that RCS giving it a better delivery chance. The proposed CADD scheme inherits this heading-awareness feature from our previously-proposed infrastructure-assisted data delivery (IADD) scheme [5].

Our contributions are as follows:

- Proposing the caching-assisted data delivery (CADD) scheme aiming at reducing the access cost of vehicular resources and expediting the access time. To the best of our knowledge, CADD is the first data delivery scheme that considers caching-assistance from road-side entities to support location-based vehicular information services.
- Introducing the road caching spot (RCS) as a cost-effective assistant to the highly-priced RSUs.
- Proposing a decentralized, dynamic, centrality-based caching mechanism with data popularity in cache replacement for favoring highly-interested data.
- Improving the commonly used cache replacement concept through considering re-caching a replaced data chunk at another caching spot instead of dropping it.

We evaluate the performance of the proposed CADD scheme using the NS-3 simulator and compare it to the popular store-carry-forward (SCF) mechanism with no road-side assistance. Simulation results show that CADD achieves significant improvements in both the access cost and delay.

The remainder of this paper is organized as follows. In Section II, we discuss some related work on infrastructure-assisted data delivery for VANETs and data caching. We present the proposed CADD scheme in Section III along with the caching mechanism. In Section IV, we present the performance evaluation of the scheme and the simulation results. Finally, we conclude the paper and present our future work in Section V.

II. Related Work

In this section, we touch upon some related work in the areas of infrastructure-assisted data delivery and data caching.

A. Infrastructure-assisted Data Delivery for VANETs

Among the many VANET data delivery schemes available in the literature, some are proposed with infrastructure-assistance to enhance the forwarding performance. All these schemes utilize assistance from RSUs deployed at intersections. For forwarding, some of these schemes utilize multi-hop communication over vehicles with SCF mechanisms to handle intermittent connectivity [5] [6], and others assume that RSUs are inter-connected through the Internet and utilize that for delay-critical forwarding [7] [8]. An example of the former category is our IADD scheme [5] that seeks RSU forwarding assistance in handling cases when the data carrying vehicles are going away from the intended destination for the sake of improving both the data delivery ratio and delivery delay. Another example is the Static-Node Assisted Adaptive routing protocol (SADV) [6] that utilizes RSUs at intersections for reducing the delivery delay. In SADV, a data packet can be stored at an RSU till a forwarding vehicle is encountered on the best delivery path for expedited forwarding. An example of the latter category is the Infrastructure-Assisted Geo-Routing scheme [7] that utilizes inter-connectivity of RSUs for improved end-to-end performance through reducing the number of hops and, hence, the delivery delay. Another example is the Infrastructure-Assisted Routing scheme proposed in [8] that follows the same concept proposed in [7] with the focus being on the RSUs buffer allocation and management challenges.

Although the aforementioned schemes succeed in achieving performance improvements, they build on an assumption of having an RSU deployed at each intersection. As mentioned earlier, the RSU models available on the market are quite pricey and, therefore, do not readily support such ubiquitous deployments. In addition, the assistance sought from RSUs in these schemes is only for forwarding purposes without caching considerations. Our proposed CADD scheme tackles these limitations through the introduction of the lower-priced RCS and using it for both forwarding and caching assistance.

B. Data Caching

Generally speaking, different data caching mechanisms are proposed in the literature to handle, for instance, the web caching [9] and information-centric network (ICN) caching [3] components. Examples of such mechanisms include the ubiquitous, probabilistic, and centrality-based caching. Among the aforementioned mechanisms, centrality-based caching has proven to be highly efficient in terms of cache hits with reasonable storage requirements [4]. Unfortunately, the centrality-based caching mechanisms proposed in the literature are designed for use in networks with static topologies, mainly the Internet. For example, the betweenness centrality-based caching mechanism [4] computes a node centrality value as the the number of times that node lies in the set of shortest paths between all pairs of nodes in the network, with the argument that if a node lies in many delivery paths, it is more likely to experience a cache hit. Apparently, this approach cannot be applied directly to a dynamic network such as a VANET. Therefore, there is a need for a mechanism that utilizes this centrality-based concept with support for highly dynamic networks.

In the area of caching in VANETs, few schemes are proposed in the literature that utilize the caching concept but with considering the mobile vehicles themselves to be the caching entities. In [10], the authors extended the location-aware VITP protocol [11] to enable in-vehicle caching. VITP is proposed for the retrieval of vehicular information over VANET through directing queries to areas of interest and retrieving resolved replies with both query and reply dissemination being handled by intermediate vehicles through multi-hop communication. The caching-enabled VITP presented in [10] allows the intermediate nodes to cache the replies on their way to the requesting nodes. While propagating the queries to the areas of interest, the VITP-enabled nodes check their local cache for the possibility of cache hits. If a matching replica is not found locally, the query is forwarded to a neighboring vehicle. The authors use the Time-to-Live (TTL) value of the messages as the metric for cache replacement and management. A message is removed from the cache once its TTL reaches a pre-defined value. Another example is the CRoWN framework [12] that brings the content-centric networking (CCN) concept [13] to the vehicular environment

implemented on top of the IEEE 802.11p standard. In CRoWN, communications are driven by the contents instead of host addresses while exploiting in-network caching and replication to achieve fast content retrieval. Although these schemes improve on their counterparts that do not consider caching, with the very dynamic nature of VANETs, caching replicas on mobile nodes that can be reached opportunistically does not achieve a high level of cache hits. In addition, with the large-scale nature of VANETs, finding a vehicle with a replica of interest requires querying a huge number of vehicles. Thus, a solution that utilizes static nodes for on-road caching is much more desirable to increase the opportunities of cache hits and minimize the interest dissemination overhead.

III. Caching-Assisted Data Delivery (CADD)

One of the main goals of the proposed CADD scheme is to minimize the cost of accessing the vehicular resources through deploying road caching spots, one at each intersection, that help in caching previously asked-for data for resolving later interests. The second goal is to reduce the roundtrip delay of the interest-reply cycle through utilizing the caching concept for bringing data closer to requesters and introducing cache hits on the interest dissemination path. In order to achieve these goals, the CADD scheme depends on the caching and forwarding capabilities of the deployed RCSs and vehicles on the road that work as carriers for both interests and replies.

As the caching concept is a main part of our proposed scheme, we introduce the RCS that works as our caching and delivery-assistant to be deployed at each intersection. Compared to the different RSU models that are on the market (e.g. the LOCOMATE devices that are used in the US pilot and test-field programs [14]), the RCS is a more cost-effective solution for data delivery assistance as it only holds the components needed for the caching and multi-hop data delivery processes. The other RSU models include a variety of components that are not required for those processes such as the Ethernet, M2M, and GPS modules. Our proposed RCS is a simple, light-weight device that can be deployed on traffic lights or electric poles to complement RSUs for providing ubiquitous road-side assistance. It consists of an 802.11p radio for communication which comes with an embedded processor, a memory chip for caching, and a power port. Fig. 1 shows the basic architecture of an RCS. It is worth noting that the RCS is not proposed to replace the RSU; it will be used to complement and assist the RSUs when their ubiquitous deployment is not feasible.

Since our CADD scheme aims at providing vehicle-assisted information services to remote end-users, a connection needs to be established between these end-users and the vehicular network in order to be able to inject the service requests and get the data replies. As part of our system, we deploy gateway entities to work as the points of attachment (PoA) of end users to the vehicular network. A gateway can either have the same architecture as an RCS with an added capability for Internet-connectivity, or can be an RSU. A single gateway will be deployed at each area of a city/deployment region which will be divided into main non-overlapping areas. An end-user interested in a service communicates with a gateway through the Internet. Each end-user, in the subscription stage, selects which gateway will be his/her PoA based on the user's preference and the area he/she will be interested in getting services from more often.

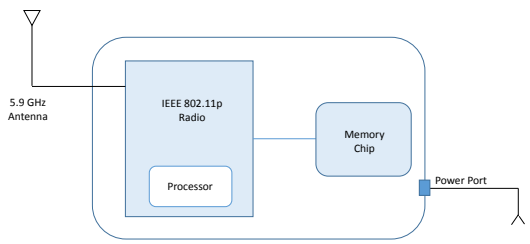

Fig. 1. The basic architecture of an RCS.

In addition to the *caching-assisted* feature of the proposed CADD scheme, another main feature is being *heading-aware*. CADD builds on the heading-awareness concept of our previously proposed IADD scheme [5] which considers vehicles' headings for the goals of improving the data delivery ratio and end-to-end delay.

In the next subsections, we discuss the general operation of CADD through an illustrating scenario, then we describe the detailed operations of the three main entities involved in the scheme; a vehicle on the road, an RCS at an intersection, and a gateway.

A. CADD General Operation

The service acquisition process starts with an end-user sending a request to the designated gateway which formulates a corresponding interest then injects it into the vehicular network. The interest is disseminated in the network through vehicles and RCSs towards the AoI defined in the interest packet. While passing by RCSs in its path, the interest is checked at each passed-by RCS for a cached match (a cache hit). Unless a cache hit happens, the interest is forwarded towards the AoI using heading-aware geographical forwarding as detailed later in this section.

Once the interest reaches a vehicle in the AoI, this vehicle generates a data reply packet resolving the interest parameters. On the way back to the requesting gateway, the reply is cached at the RCS with the maximum centrality among all the RCSs in the interest-followed path. Details about the proposed centrality-based caching mechanism are discussed in Section III-C. With caching the collected data on the reply delivery path, the scheme brings chances for cache hits to be encountered by subsequent interests in the same data as depicted in the scenario shown in Fig. 2.

While propagating the interest and reply packets, the data carrying vehicles keep checking their headings at any approached intersection. If it happens that a vehicle does not find a neighboring vehicle as a potential forwarder, it keeps carrying the packet if it finds that it is going towards the packet's destination, otherwise, it leaves the packet at the neighboring RCS to give it a chance for a better forwarding opportunity towards its destination which means a higher delivery probability for that packet.

All vehicles and RCSs exchange periodic beacon packets carrying their IDs, positions, and velocities (for vehicles) to announce their presence. Such beacons are used for neighbor discovery needed by the geographical forwarding procedure.

The scenario illustrated in Fig. 2 shows the benefit of caching-assistance in reducing access to the vehicular resources, implying reducing the service access cost, and mini-

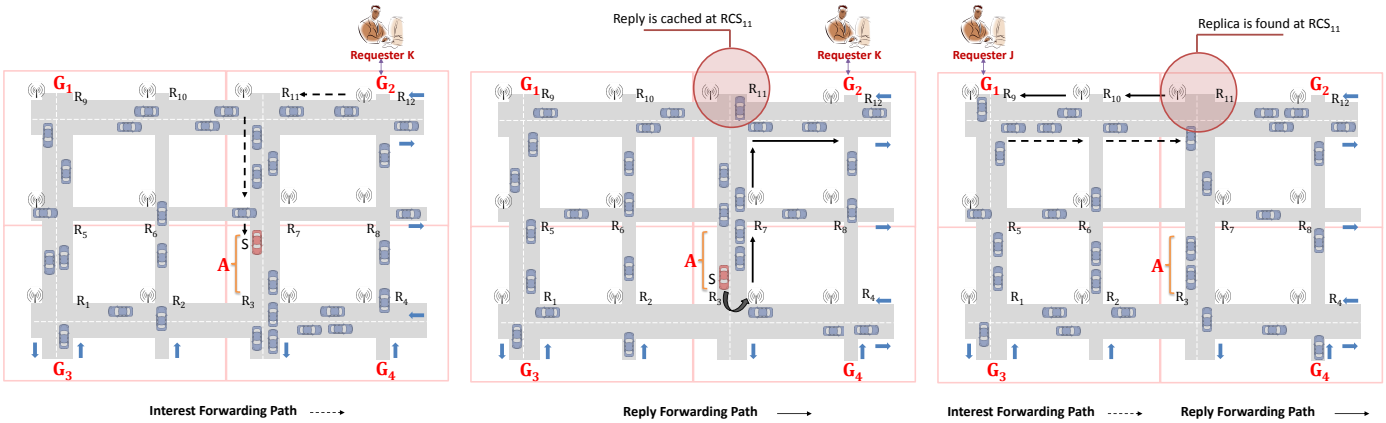

Fig. 2. An illustration of a scenario showing the benefit of caching on the service data delivery path.

mizing the data retrieval delay. In Fig. 2(a), requester K needs information about an event in the area of interest A. Since he has registered to gateway G_2, his request goes to it through the Internet. G_2 generates a corresponding interest, I_k, and sends it to the closest neighboring vehicle. The carrying vehicles follow the CADD heading-aware forwarding procedure to get the interest towards A resulting in the forwarding path shown in Fig. 2(a). During the forwarding process, the RCSs that are passed by the interest packet check the availability of a matching reply in their caches. In addition, they store information in the interest header about the maximum central and 2^{nd} maximum central RCSs to be used for the caching purposes. In the case shown, no match has been encountered so the interest has had to go all the way to A.

In Fig. 2(b), I_k is received by a vehicle S in A. S generates a reply packet, P_k, with corresponding data matching the parameters of I_k, and caching-related fields matching those carried in I_k. S follows the CADD forwarding procedure to send P_k back to G_2 resulting in the path shown in Fig. 2(b). At each intersection, the carrying vehicle of P_k checks if the neighboring RCS is the one with the maximum centrality stored in the packet header. In the illustrated scenario, it happened that RCS_{11} was the maximum central node on the interest path and, therefore, it is where a replica of P_k is cached on the way back. Note that an RCS at an intersection i is abbreviated as R_i in the figure for simplicity.

A moment later, a requester J, registered with gateway G_1, gets interested in information similar to that requested by requester K. G_1 generates a corresponding interest I_j and sends it towards A through the vehicular network. As I_j is checked at the passed-by RCSs for a matching reply, it happens that it hits the replica of P_k cached in RCS_{11}. Since P_k matches the parameters of I_j, a replica of it is forwarded to G_1. In this case, I_j is kept from going all the way towards A minimizing the roundtrip access delay. Since the reply sent to J was a replica of a previously-generated data packet, no access for vehicular sensing resources was required saving the cost that would have incurred if caching was not considered. The interest and reply paths of this case are shown in Fig. 2(c).

B. CADD at a Vehicle on the Road

In CADD, vehicles work as the main carriers for both interest and reply packets. We detail the forwarding logic followed by a vehicle in Algorithm 1 for both forwarding an interest (lines 10-25) and forwarding a reply (lines 28-52). On a road, a vehicle can be in one of two modes: a segment mode or an intersection mode. In the *segment mode*, a vehicle is moving on a road segment and not close to any intersections while in the *intersection mode*, a vehicle is approaching an intersection and hence, has the opportunity of getting RCS assistance. We discuss the detailed forwarding logic below.

1) Interest Forwarding:
When a vehicle gets an interest packet which is not expired yet, it checks if it is in the defined AoI. If so, it generates a reply packet, sets the corresponding caching values in the header which it retrieves from the interest header, and triggers the reply forwarding procedure (lines 12-14). Otherwise, the vehicle checks its current mode of operation to start forwarding the interest. If it is in *the segment mode* (lines 18-22), it anchors the packet towards the next RCS (RCS_{next}) through the use of greedy forwarding. It checks its neighboring vehicles to find the one closest to RCS_{next} and if this potential forwarder is closer to RCS_{next} than the vehicle itself, it forwards the packet to it, otherwise, it keeps holding the packet until a better forwarder is encountered or it approaches RCS_{next}.

When a vehicle gets a beacon from an approached RCS, it activates the *intersection mode* of forwarding (lines 16 & 17). The carrying vehicle sends the interest to RCS_{cur} which checks the possibility for a cache hit, if it has a matching reply in its cache, or continues the forwarding process towards the AoI, otherwise.

2) Reply Forwarding:
The reply forwarding procedure gets called when a vehicle generates a reply packet itself (if it gets an interest while it is in the corresponding AoI), or receives a generated one to be forwarded. If the vehicle is in the *segment mode*, it forwards the packet towards RCS_{next} the same way defined above in the interest forwarding (lines 47-51). If it is in the *intersection mode*, it goes through many check points as follows.

(a) **The vehicle checks if it has a candidate neighbor for forwarding the packet**. The beacon packet sent from RCS_{cur} carries the RCS's real-time assessment of the densities of its linked road segments. A segment density assessed by an RCS is defined as the number of beacons heard by that RCS from vehicles on that road segment

during an assessment period ($asmt\_prd$):

$$density(s_i) = \#received\_beacons_{s_i} \mid_t^{t+asmt\_prd} \quad (1)$$

The vehicle uses the received densities for prioritizing the neighboring segments through computing a weighted priority for each segment as follows

$$weighted\_priority(s_i) = \alpha \text{ x } density(s_i) + \beta \text{ x } direction\_priority(s_i) \quad (2)$$

where $direction\_priority(s_i)$ is the priority of road segment i in terms of its direction towards the packet destination, α and β are tunable weights. The vehicle uses the prioritized segment list for finding its next-hop forwarder. It checks the availability of neighboring vehicles on the segments in a prioritized order and if it finds any, it stops looping on the segments and selects the next forwarder on the segment with availability in a greedy fashion (lines 33-37).

(b) **If not (a), the vehicle checks if it is going towards the packet destination**. If it is, it keeps carrying the packet till it finds a better forwarder (38-40). Otherwise, it sets a designated forwarding flag in the packet to ON then sends it to RCS_{cur} to give it a better forwarding opportunity (lines 41 & 42).

(c) **The vehicle checks if RCS_{cur} is the RCS with maximum centrality (RCS_{max}) stored in the header for caching**. If so, the vehicle generates a replica of the reply packet, marks its caching flag as ON, and sends it to RCS_{cur} to be cached (lines 43 & 44).

(d) **If not (c), the vehicle checks if it is going far from RCS_{max} so that the packet may not pass by the caching RCS on its way**. If this is the case, the carrying vehicle generates a replica of the packet, sets its destination to RCS_{max}, and forwards it towards RCS_{max} using the same forwarding procedure (lines 45 & 46). The idea behind that is to maintain the caching opportunity for that packet regardless of the path followed by the original packet itself.

Algorithm 1 : CADD at a Vehicle

```
1: Input:
2: Forwarding vehicle V
3: Interest packet i
4: Reply packet p
5: Gateway Q
6: Set of neighboring road segments S sent from a neighboring RCS along
   with their densities
7: Neighborhood list N
8:
9: forward_interest(i)
10: Begin
11:   if i is not expired then
12:       if V is in the area of interest defined in i then
13:           generate a reply packet p and set its RCS_max and RCS_2max
              fields as carried in i
14:           forward_reply(p)
15:       else
16:           if Intersection_Mode = true then
17:               send i to RCS_cur
18:           else      //In the segment mode
19:               if N is empty then
20:                   keep holding i
21:               else
22:                   send i to the neighbor closest to RCS_next if it is closer
                      than V
23:   else
24:       drop i       // i is expired
25: End
26:
27: forward_reply(p)
28: Begin
29:   if V is a neighbor to Q then
30:       send p to Q
31:   else
32:       if Intersection_Mode = true then
33:           prioritize S according to the density and direction priority
34:           for all s_i ∈ the prioritized segment set do
35:               if list of neighboring vehicles on s_i is not empty then
36:                   send p to the farthest vehicle on s_i
37:                   break
38:           if p is not relayed then
39:               if V is heading towards Q then
40:                   keep holding p
41:               else
42:                   send p to RCS_cur with the forwarding flag ON
43:           if RCS_cur = RCS_max then
44:               send a replica of p to RCS_cur with the caching flag ON
45:           else if V is heading away from RCS_max then
46:               send a replica of p to RCS_max using the same procedure
47:       else      //In the segment mode
48:           if N is empty then
49:               keep holding p
50:           else
51:               send p to the neighbor closest to RCS_next if it is closer
                  than V
52: End
```

C. CADD at an Intersection RCS

RCSs are used mainly for caching replicas of the reply packets with the aim of resolving later relevant interests so that there will not be a need to access vehicular resources every time an interest is injected into the vehicular network. They are also used for forwarding assistance in cases when the packet-carrying vehicles are heading away from the destination for the sake of saving the packet from getting far from its destination which may lead to eventual dropping. The logic followed by an RCS for both caching and forwarding is detailed in Algorithm 2 and discussed in the following paragraphs.

1) Receiving an Interest:

When an RCS receives an interest packet from a vehicle, if this interest is not expired yet, it checks its cache for a matching replica. If it finds a match, it sends a reply back to the requesting gateway (lines 11 & 12), otherwise, it forwards the interest towards the AoI using the forwarding procedure discussed later in this section (line 18).

Before forwarding a received interest, an RCS checks its centrality value relevant to the interest type and area, as discussed later in this section, to determine if it is a candidate for the maximum central RCS (RCS_{max}) or the 2^{nd} maximum central RCS (RCS_{2max}) among the RCSs encountered on the interest-traversed path so far (lines 14-17). In the interest header, the ID and location of RCS_{max} and RCS_{2max} is recorded along the path and is updated by a passed-by RCS if it is a better candidate than any of the recorded ones.

2) Receiving a Reply:

After receiving a reply packet by an RCS, it starts checking the forwarding and caching flags carried in the reply header. If the caching flag is ON, the RCS calls the caching procedure for handling both the storage and cache replacement (lines 27 & 28). If the forwarding flag is ON, the RCS inserts the

packet into the forwarding queue to be forwarded using the forwarding procedure discussed below (lines 25 & 26).

3) Forwarding a Packet:
The packet forwarding procedure is called by an RCS when it has to forward either an interest or a reply packet. An RCS's forwarding logic has similarities with the one used by a vehicle for forwarding a reply. When a packet needs to be forwarded, the carrying RCS uses Eqn. 2 for computing a weighted priority for all of its linked segments based on the density and direction criteria as discussed earlier. Afterwards, the RCS searches for a neighboring vehicle on these road segments in the order of their weighted priorities (lines 33 & 34). If neighbors are found on a segment while searching, the RCS sends the packet to the farthest vehicle on that segment (lines 35 & 36). If no potential forwarder is encountered, the RCS keeps carrying the packet and marks it to be re-transmitted (lines 38 & 39).

4) Caching a Reply:
When an RCS receives a reply packet to be cached, it checks the availability of a vacant spot in its cache; if there is any, it caches the packet right away (lines 44 & 45). If the RCS's cache is full, it considers replacing a previously cached packet, (lines 46-52), as discussed below.

One of the main contributions of this paper is the proposed cache replacement mechanism which, compared to other caching mechanisms that drops the replaced packet, gives this packet another caching chance to stay longer in the network and increase the chances for a cache hit. In contrast to many caching mechanisms that consider picking the to-be-replaced packet using the Least Recently Used (LRU) policy while ignoring the popularity of the different packets, our replacement mechanism considers this popularity criterion in picking the replacement candidate. Considering popularity favors the packets with more interests from end-users which enhances the QoS.

When a replacement needs to be considered, the RCS with the full cache picks the least popular packet among those stored in its cache (line 47). Details about popularity computation are discussed later in this part. When it happens that an RCS finds many packets with the same lowest popularity value, it picks the LRU one among those of equal popularity. Then, the RCS compares the popularity of the candidate packet ($lowest\_p$) to the to-be-cached packet (p). The one with the higher popularity will be the caching winner and the other one will be given another caching chance. Each reply packet carries in its header information about the 2^{nd} maximum RCS (RCS_{2max}) encountered on the interest path and there is where the other caching chance will be targeted. The packet with the lower popularity between $lowest\_p$ and p will be offloaded to the RCS_{2max} recorded in its header using the same forwarding procedure (lines 48-52). A packet is kept in an RCS's cache till the packet expires or gets replaced.

5) Centrality Computation:
Another main contribution in the paper is our dynamic, decentralized mechanism for computing the centrality of the RCSs. These centrality values are used for caching purposes through directing a reply replica to the maximum central node encountered on the interest forwarding path. A replaced replica is directed to the 2^{nd} maximum central node encountered on its interest path, as discussed before.

The concept of centrality-based caching aims at directing caching to the nodes that are most central in the network as they are more likely to get many interests passing by and hence, more cache hits. Unlike the static centrality-based caching mechanisms that compute the centrality value of a node based on its position in the static network topology, our proposed mechanism handles the computation in a dynamic way based on the number of interests received by a node such that the more interests a node has received, the more central

Algorithm 2 : CADD at an RCS

```
1: Input:
2: Interest packet i
3: Reply packet p
4: Gateway Q
5: Set of neighboring road segments S
6: Neighborhood list N
7:
8: interest_received(i)
9: Begin
10:   if i is not expired then
11:       if there is p matching i in Cache then
12:           forward p to Q by calling forward_packet(p)
13:       else
14:           if centr_{i_(tp,a)} of the RCS > centr of RCS_max then
15:               update RCS_max, RCS_2max, and their stored locations
16:           else if centr_{i_(tp,a)} of the RCS > centr of RCS_2max then
17:               update RCS_2max and its stored location
18:           forward_packet(i)
19:   else
20:       drop i        // i is expired
21: End
22:
23: reply_received(p)
24: Begin
25:   if the packet's forwarding flag is set to ON then
26:       forward_packet(p)
27:   if the packet's caching flag is set to ON then
28:       cache_reply(p)
29: End
30:
31: forward_packet(k)        // k can be a reply packet or an interest packet
32: Begin
33:   prioritize S according to the density and direction priority
34:   for all s_i ∈ the prioritized segment set do
35:       if list of neighboring vehicles on s_i is not empty then
36:           send p to the farthest vehicle on s_i
37:           break
38:   if a forwarder is not found then
39:       keep holding k to be re-transmitted
40: End
41:
42: cache_reply(p)
43: Begin
44:   if Cache is not full then
45:       store p in Cache
46:   else      //Cache is full, consider replacement
47:       lowest_p ← the packet with the lowest popularity in Cache
48:       if p.pop < lowest_p.pop then
49:           forward p to p.RCS_2max
50:       else
51:           forward lowest_p to lowest_p.RCS_2max
52:           store p in Cache
53: End
54:
55: //The following two functions will be called periodically upon the firing
    of corresponding timers.
56: calculate_centrality()        //as in Eqn. 3
57: calculate_popularity()        //as in Eqn. 4
```

it is in the network and the more likely it will get upcoming interests.

In CADD, all the potential interests are classified into main types (e.g., traffic conditions, road conditions, and crowd). Each RCS maintains an Interest Type_Area table that lists these pre-defined interest types each associated with the different pre-defined main areas of the whole region in (type, area) 2-tuples. The RCS computes its *centrality* for each tuple and uses it when it receives an interest of that tuple for checking its candidacy to be the RCS_{max} or the RCS_{2max} on that interest forwarding path.

As shown in Eqn. 3, an RCS's *centrality* of a certain tuple is computed as the number of interests of that type towards that area received during a pre-defined centrality period ($centr\_prd$).

$$centrality_{i\_(type,area)} = \#interest_{i\_(type,area)} \,|_{t}^{t+centr\_prd} \quad (3)$$

6) Popularity Computation:
Using a similar dynamic mechanism to the one used for computing the centrality, an RCS computes a popularity value for all of the different packets it carries in its cache as the popularity of its corresponding interest tuple. A tuple *popularity* is computed as the number of interests of that type towards that area received during a popularity period pop_prd, as shown below

$$popularity_{i\_(type,area)} = \#interest_{i\_(type,area)} \,|_{t}^{t+pop\_prd} \quad (4)$$

The RCS uses the computed popularity values for picking the caching replacement candidate as discussed before.

D. CADD at a Gateway

In CADD, a gateway is used as the PoA of the requesters to the vehicular network. Their sole job in the scheme is injecting the service requests into the network and waiting for the replies, which is presented in Algorithm 3 through the *request_data* procedure. When a gateway receives a service request through the Internet, it generates an interest packet with the corresponding parameters defined in the request including the interest type, AoI, and the expiry times of the interest and its requested reply. Before injecting the interest into the network, the gateway initializes the caching-related fields of the interest header; the centrality, IDs and locations of RCS_{max} and RCS_{2max}. Then, it sends the generated interest to its closest neighbor of the vehicular network. A gateway keeps track of all the interests it sent till it gets matching replies or the interests expire. Once a gateway receives a reply matching with a stored interest, it sends this reply back to the requester through the Internet.

IV. Performance Evaluation

In this section, we evaluate the performance of the CADD scheme in comparison to the non-assisted SCF scheme using simulations. The performance is analyzed in terms of: 1) the ratio of received replies through accessing vehicular resources to the total received replies, which is an indicator of the incurred access cost, and 2) the average round-trip access delay since an interest is sent until its reply is received.

Algorithm 3 : CADD at a Gateway

```
1: Input:
2: Neighborhood list N
3:
4: request_data()
5: Begin
6:   generate interest i with the corresponding parameters
7:   initialize the centrality-related caching fields of the header
8:   send i to the nearest neighbor in N
9:   keep track of i till either getting a reply or it expires
10: End
```

A. Simulation Setup

Both CADD and SCF are implemented using the NS-3 network simulator [15]. Simulations are performed over different vehicle densities for a period of 2000 seconds each. We considered a grid simulation topography similar to the part of the city of Edmonton's downtown shown in Fig. 3 with one gateway deployed at each corner. The interest generation is uniformly distributed among the four gateways with the injection rate equals 20 seconds. In generating interests, 4 interest types and 4 targeted areas of interest are considered leading to 16 different interest tuples. The SUMO vehicular simulator [16] is used to generate realistic mobility traces with the maximum vehicular speed is 40 km/h. The IEEE 802.11p WAVE standard is used for communication in the vehicular network with the beacon interval set to 0.5 seconds and the transmission range set to 150 meters.

In CADD, an RCS is added at each intersection. The $centr\_prd$ is set to 250 seconds and the pop_prd is assigned the whole simulation time (2000 sec). The α and β weights for calculating segments' priorities are set to 0.2 and 0.8, respectively, giving a higher weight at each RCS to the segment whose direction brings a packet closer to the destination. We consider 2 different cache sizes (the maximum number of cached packets at each RCS) to show the effect of the popularity-based cache replacement. Cache sizes of 5 (CADD-5) and 20 (CADD-20) are considered to represent cases with stimulation for cache replacement and accommodation of all to-be-cached packets, respectively, according to the 2000 second simulation time. Thus, in CADD-5, only the 5 most popular tuples are kept at an RCS_{max} while less popular ones are offloaded to their corresponding RCS_{2max}. Larger values of cache sizes with higher data traffic volumes could be equally considered with longer simulation periods.

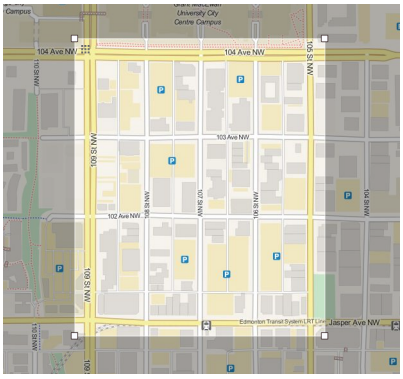

Fig. 3. The part of Edmonton city representing the simulation topography.

B. Simulation Results and Analysis

First, we compare CADD-5, CADD-20, and SCF in terms of the average round-trip delay over varying vehicular densities. As shown in Fig. 4, CADD has significantly improved the delay compared to SCF. This improvement is due to the caching assistance that brought the replies closer to the requesters saving the interests from having to go to the AoI. In addition, due to the heading-awareness nature of CADD, both the interests and replies have been saved from going farther from their destinations which reduces the delay. Comparing CADD-5 and CADD-20, we can notice a slight increase in the delay when reducing the cache size. Since our cache replacement mechanism gives a replaced packet another caching chance instead of dropping it, reducing the cache size has not greatly affected the delay as still there have been chances for cache hits for the replaced replicas.

Second, we perform the same comparison considering the 2^{nd} metric; the ratio of vehicular resource-accessed replies to the total received replies. The higher this ratio is, the higher the incurred access cost is. Since SCF does not involve any caching assistance, all the received replies are vehicle-generated so the ratio is always equal to 1, as shown in Fig. 5. With caching assistance, CADD has achieved significant reduction in this ratio. For the same reason discussed earlier, CADD-5 maintains almost the same access cost of CADD-20. To summarize, the simulation results indicate that CADD has achieved significant improvements in terms of both the access delay and cost compared to the commonly-used SCF.

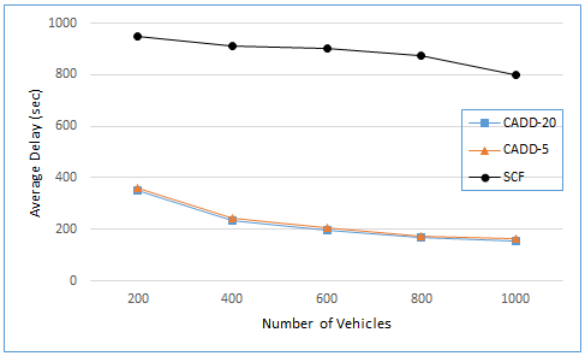

Fig. 4. Average delay of CADD and SCF with varying densities.

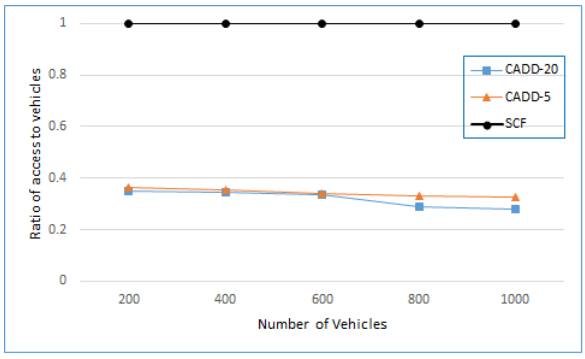

Fig. 5. Ratio of access to vehicular resources of CADD and SCF with varying densities.

V. Conclusions and Future Work

In this paper, we proposed the caching-assisted data delivery (CADD) scheme that enhances access to vehicular resources for vehicle-based public sensing services. Through applying caching on the data delivery path, CADD aims at reducing the cost and delay of accessing vehicular resources. CADD relies on the deployment of a light-weight road caching spot (RCS) at each intersection, and vehicles work as carriers of both interests and replies between the RCSs. As part of CADD, a novel centrality-based caching mechanism was proposed that handles the dynamic nature of vehicular networks and considers popularity in cache replacement. CADD considers vehicles' headings to direct interests and replies towards their destinations. Performance evaluation showed that CADD achieves significant improvements in both the access cost and delay compared to another scheme that does not involve caching-assistance. In our future work, we will consider having a targeted cache hit ratio and study the placement and distribution of RCSs to achieve this ratio.

Acknowledgment

This research is supported by a grant from the Natural Sciences and Engineering Research Council (NSERC) of Canada.

References

[1] S. Abdelhamid, H. S. Hassanein, and G. Takahara, "Vehicle as a resource (VaaR)," IEEE Network Magazine, 2014. [Online]. Available: http://arxiv.org/ftp/arxiv/papers/1403/1403.1890.pdf

[2] S. Abdelhamid, H. S. Hassanein, and G. Takahara, "Vehicle as a mobile sensor," School of Computing - Queen's University, Canada, Tech. Rep. 2014-618, 2014.

[3] G. Zhang, Y. Li, and T. Lin, "Caching in information centric networking: a survey," *Computer Networks*, vol. 57, no. 16, pp. 3128–3141, 2013.

[4] W. K. Chai, D. He, I. Psaras, and G. Pavlou, *Cache "less for more" in information-centric networks*, ser. NETWORKING 2012, Lecture Notes in Computer Science. Springer, 2012, pp. 27–40.

[5] S. Abdel Hamid, M. Abu-Elkheir, H. S. Hassanein, and G. Takahara, "Towards provisioning vehicle-based rural information services," in *IEEE 37th Conference on Local Computer Networks Workshops*, 2012, pp. 835–842.

[6] Y. Ding and L. Xiao, "SADV: Static-node-assisted adaptive data dissemination in vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 59, no. 5, pp. 2445–2455, 2010.

[7] D. Borsetti and J. Gozalvez, "Infrastructure-assisted geo-routing for cooperative vehicular networks," in *IEEE Vehicular Networking Conference (VNC)*, 2010, pp. 255–262.

[8] Y. Wu, Y. Zhu, and B. Li, "Infrastructure-assisted routing in vehicular networks," in *the 31st IEEE International Conference on Computer Communications (INFOCOM '12)*, 2012, pp. 1485–1493.

[9] W. Ali, S. M. Shamsuddin, and A. S. Ismail, "A survey of web caching and prefetching," *Int. J. Advance. Soft Comput. Appl.*, vol. 3, no. 1, pp. 18–44, 2011.

[10] N. Loulloudes, G. Pallis, and M. D. Dikaiakos, *Caching dynamic information in vehicular ad hoc networks*, ser. Euro-Par 2010-Parallel Processing. Springer, 2010, pp. 516–527.

[11] M. D. Dikaiakos, S. Iqbal, T. Nadeem, and L. Iftode, "VITP: an information transfer protocol for vehicular computing," in *the 2nd ACM international workshop on Vehicular ad hoc networks*, Germany, 2005, pp. 30–39.

[12] M. Amadeo, C. Campolo, and A. Molinaro, "CRoWN: Content-centric networking in vehicular ad hoc networks," *IEEE Communications Letters*, vol. 16, no. 9, pp. 1380–1383, 2012.

[13] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, 2009, pp. 1–12.

[14] "Arada systems." [Online]. Available: http://www.aradasystems.com/ (Accessed 2014, June 26)

[15] "The NS-3 network simulator." [Online]. Available: https://www.nsnam.org/ (Accessed 2014, June 26)

[16] "SUMO (simulation of urban mobility)." [Online]. Available: http://sumo-sim.org/ (Accessed 2014, June 26)