

Temporal data dissemination in vehicular cyber-physical systems

Liu, Kai; Lee, Victor Chung Sing; NG, Joseph K Y; Chen, Jun; Son, Sang Hyuk

Published in:
IEEE Transactions on Intelligent Transportation Systems

DOI:
[10.1109/TITS.2014.2316006](https://doi.org/10.1109/TITS.2014.2316006)

Published: 01/12/2014

[Link to publication](#)

Citation for published version (APA):
Liu, K., Lee, V. C. S., NG, J. K. Y., Chen, J., & Son, S. H. (2014). Temporal data dissemination in vehicular cyber-physical systems. *IEEE Transactions on Intelligent Transportation Systems*, 15(6), 2419-2431. [A6]. <https://doi.org/10.1109/TITS.2014.2316006>

General rights

Copyright and intellectual property rights for the publications made accessible in HKBU Scholars are retained by the authors and/or other copyright owners. In addition to the restrictions prescribed by the Copyright Ordinance of Hong Kong, all users and readers must also observe the following terms of use:

- Users may download and print one copy of any publication from HKBU Scholars for the purpose of private study or research
- Users cannot further distribute the material or use it for any profit-making activity or commercial gain
- To share publications in HKBU Scholars with others, users are welcome to freely distribute the permanent publication URLs

Temporal Data Dissemination in Vehicular Cyber-Physical Systems

Kai Liu, *Member, IEEE*, Victor C.S. Lee, *Member, IEEE*, Joseph K. Y. Ng, *Senior Member, IEEE*
Jun Chen, and Sang H. Son, *Fellow, IEEE*,

Abstract—Efficient data dissemination is one of the fundamental requirements to enable emerging applications in vehicular cyber-physical systems. In this work, we present the first study on real-time data services via roadside-to-vehicle communication by considering both the time-constraint of data dissemination and the freshness of data items. Passing vehicles can submit their requests to the server and the server disseminates data items accordingly to serve the vehicles within its coverage. Data items maintained in the database are updated periodically to keep the information up-to-date. We present the system model and analyze challenges on data dissemination by considering both application requirements and communication characteristics. On this basis, we formulate the temporal data dissemination (TDD) problem by introducing the snapshot consistency requirement on serving real-time requests with temporal data items. We prove that TDD is NP-hard by constructing a polynomial time reduction from the Clique problem. Based on the analysis of the time bound on serving requests, we propose a heuristic scheduling algorithm, which considers the request characteristics of productivity, status and urgency in scheduling. An extensive performance evaluation demonstrates that the proposed algorithm is able to effectively exploit the broadcast effect, improve the bandwidth efficiency and enhance the request service chance.

Index Terms—Data dissemination, temporal consistency, real-time scheduling, vehicular cyber-physical system

I. INTRODUCTION

VEHICULAR cyber-physical systems (VCPS), which embrace the latest advances in communications, computing, electronics, sensing and control, etc., are envisioned as a promising approach to achieving breakthroughs in transportation safety, efficiency and sustainability [1]. Efficient data dissemination is critical to enable emerging applications in VCPS, such as collision avoidance [2], roadway reservation [3] and autonomous intersection management [4], to name but a few. This paper investigate the scenario where the roadside unit is installed along the road and provides real-time data services to passing vehicles. Data items maintained in the

database are updated periodically to keep the information up-to-date. Vehicles within the service range of the roadside unit can submit requests for particular services or information, such as routing advisories, road conditions and parking slots, etc. In response, the roadside unit disseminates corresponding data items to serve the vehicles. Obviously, to enable such services, it is expected to guarantee the freshness of information and the timeliness of data dissemination.

The vehicular communication system plays an important role in interconnecting the driver, the vehicle and the cyber information in VCPS. Different parties including automotive manufacturers, governments and universities are actively engaged into the research of vehicular communications. In automotive manufacturers, *MyFord Touch* is an embedded communication system developed by the Ford Motor Company and Microsoft. It enables drivers to interact with vehicles via smart phones. *Toyota Entune* is an integrated multimedia navigation system developed by the Toyota Motor Company. It provides data services such as stocks and traffic information. *Mbrace2* developed by the Mercedes-Benz Company provides drivers with both safety-critical and value-added information via vehicular communications. The U.S. Department of Transportation is also collaborating with a number of universities on the development of a variety of vehicular communication systems. Examples include the *Connect Vehicle* research program, the *Vehicle Infrastructure Integration* project, the *MITCarTel* project and the Berkeley *PATH* project, etc. Clearly, there is great significance of the research on providing real-time data services via vehicular communications.

Although there has been extensive research on data scheduling in conventional mobile computing environments ([5], [6], [7]), none of them addressed unique challenges arising in VCPS for providing real-time data services. In the community of vehicular ad-hoc networks (VANET), great efforts have been put into the roadside-to-vehicle and vehicle-to-vehicle communications ([8], [9], [10]). Nevertheless, these studies mainly focused on the issues resided on MAC or PHY layers, such as improving wireless communication quality or reliability in VANET. To the best of our knowledge, this is the first work on data dissemination in VCPS which considers both the time-constraint of services and the freshness of data items at the application layer. In particular, the following issues are investigated. First, there is a strict time constraint on serving requests. On the one hand, vehicles cannot retrieve data items after leaving the service region of the roadside unit. On the other hand, there are a variety of location-dependent data services such as route queries, which have to be satisfied

This work was supported in part by the HKBU Research Centre for Ubiquitous Computing (RCUC), the Institute of Computational and Theoretical Studies (ICTS), and the HKBU Strategic Development Fund under the grant no. HKBU SDF 10-0526-P08, and in part by MSIP (CPS Global Center) and NRF GRL Program (2013K1A1A2A02078326).

Kai Liu is with the College of Computer Science, Chongqing University, Chongqing, China (email: liukai0807@gmail.com)

Victor C.S. Lee is with the Computer Science Department, City University of Hong Kong, Hong Kong (email: csvlee@cityu.edu.hk)

Joseph K.Y. Ng is with the Computer Science Department, Hong Kong Baptist University, Hong Kong (email: jng@comp.hkbu.edu.hk)

Jun Chen is with the School of Information Management, Wuhan University, China (corresponding author, email:christina_cj@whu.edu.cn)

Sang H. Son is with the Information and Communication Engineering at DGIST, Korea (email:son@dgist.ac.kr)

within a certain time bound (e.g. before reaching the road intersection). Second, a request may ask for multiple dependent data items, and the query cannot be fully processed until all the requested data items are retrieved. For example, in order to compute the optimal route to a destination, the navigation system has to request the road conditions of all possible routes. Since different routing information corresponds to different data items, the optimal route can be computed only when all the corresponding data items are retrieved. Third, due to the temporality of data items, the multiple retrieved data items should be consistent in versions. Inconsistency readings of dependent data items may result in fatal failure of services.

The main contributions of this paper are outlined as follows. First, we present the roadside-to-vehicle communication system in VCPS and investigate the newly arising challenges of data dissemination. Second, we formulate the temporal data dissemination problem called TDD by investigating the snapshot consistency requirement on serving real-time requests with temporal data items, and we prove that TDD is NP-hard by constructing a polynomial-time reduction from the Clique problem. Third, by analyzing time bound of serving real-time requests with temporal data items, we propose a heuristic scheduling algorithm, which aims at enhancing overall system performance by improving the broadcast effect, the bandwidth efficiency and the request service chance. Finally, we build the simulation model for performance evaluation. The comprehensive simulation results validate that the proposed solution is effective in providing real-time data services under different traffic scenarios and application requirements.

The rest of this paper is organized as follows. Section II presents the system model. In Section III, we formulate the temporal data dissemination problem and prove that it is NP-hard. In Section IV, we analyze the service time bound and propose a heuristic scheduling algorithm. In Section V, we build the simulation model and evaluate the algorithm performance. Section VI reviews the related work. Section VII concludes this work and discusses future research directions.

II. DATA DISSEMINATION MODEL

The data dissemination model is shown in Figure 1, where the road-side unit (RSU) is installed along the road and provides data services. In general, there are both location-independent and location-dependent services [11]. For the location-independent service such as multimedia downloads, it may tolerate longer delay to be completed. However, for the location-dependent service such as navigation queries, it may impose both temporal and spacial constraints on successful completion. In this model, we focus on the data dissemination for location-dependent services, where stringent time constraints are required. Besides, it is commonly assumed that RSUs are sparsely deployed along the road ([8], [10]). Accordingly, we consider the data service has to be completed within the coverage of one RSU.

When vehicles driving into the service range of the RSU, they may submit requests for particular services, such as road condition queries or routing advisories. Outstanding requests are pended in the service queue. According to a certain

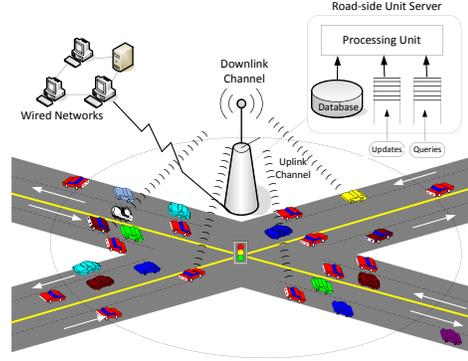


Fig. 1. Data dissemination model in vehicular cyber-physical systems

scheduling algorithm (e.g. Earliest Deadline First (EDF) [12]), the RSU retrieves data items from the local database and broadcasts them to passing vehicles. Note that the broadcast nature of wireless communication is commonly exploited for data dissemination in vehicular networks ([8], [13], [14]). Therefore, each broadcasted data item can be retrieved by all vehicles within the service region. Typically, there are two types of broadcast approaches: push-based and pull-based [15]. Safety-critical services are usually provided via push-based broadcast, where the messages are disseminated periodically. On the other hand, many value-added services are provided via the pull-based broadcast, where the messages are only disseminated based on explicit requests. This data dissemination model is dedicated to the latter application scenario.

The RSU is connected to a backbone network, via which it can access the internet and other specific sensor networks. Due to the highly dynamic nature of traffic information, the data items stored in the local database are updated periodically by the sensors and information providers from the backbone network. Once an update is installed for a data item, a new version is created and the previous version becomes outdated. Only the latest version of each data item is maintained in the database. Each request is associated with a deadline, which may be either imposed by specific application requirements or bounded by the dwell time of vehicles in the service region [15]. For requests asking for multiple dependent data items, they have to retrieve all the data items before their respective deadlines. In addition, due to the temporality of data items, the multiple dependent data items in a request have to be consistent in versions. Detailed requirements on serving time-critical multi-item requests with temporal data items are formulated as follows.

III. PROBLEM FORMULATION

The set of data items in the database is denoted by $D = \{d_1, d_2, \dots, d_{|D|}\}$, where $|D|$ is the total number of data items. Each data item d_i ($1 \leq i \leq |D|$) is characterized by a 3-tuple: $\langle V(d_i|t), U(d_i|t), E(d_i|t) \rangle$, where $V(d_i|t)$ is the value of d_i at time t . $U(d_i|t)$ and $E(d_i|t)$ represent the update time and the expiration time of d_i in its version at time t . The update interval of d_i is denoted by $l(d_i)$. Accordingly, we have

TABLE I
SUMMARY OF NOTATIONS

Notations	Descriptions	Notes
D	set of data items	$D = \{d_1, d_2, \dots, d_{ D }\}$
$V(d_i t)$	value of d_i at t	
$U(d_i t)$	update time of d_i at t	$U(d_i t) \leq t$
$E(d_i t)$	expiration time of d_i at t	$E(d_i t) > t$
$l(d_i)$	update interval of d_i	$U(d_i t) + l(d_i) = E(d_i t)$
$Q(t)$	set of pending requests at t	
Q_m	the m th request	$Q_m \in Q(t)$
q_m^n	the n th data item requested by Q_m	$q_m^n \in D$
$R(Q_m)$	requested data set of Q_m	$R(Q_m) = \{q_m^1, q_m^2, \dots, q_m^{ R(Q_m) }\}$
$S(Q_m)$	submission time of Q_m	
$L(Q_m)$	deadline of Q_m	
τ	data transmission time	
$T_{Q_m}^f(q_m^n)$	time when q_m^n is disseminated	
$T_{Q_m}^f$	time when the first data item is disseminated for Q_m	$T_{Q_m}^f = \min(T_{Q_m}^f(q_m^n))$
$T_{Q_m}^l$	time when the last data item is disseminated for Q_m	$T_{Q_m}^l = \max(T_{Q_m}^f(q_m^n))$
$E_{Q_m}^e(t)$	earliest expiration time of the requested data items	$E_{Q_m}^e(t) = \min(E(q_m^n t))$
$R_{Q_m}^u(t)$	unserved set of Q_m	$R_{Q_m}^u(t) \subseteq R(Q_m)$
$\chi_{Q_m}(t)$	tentative time bound	$\chi_{Q_m}(t) = \min(L(Q_m), E_{Q_m}^e(t))$
X_{Q_m}	determined time bound	$X_{Q_m} = \min(L(Q_m), E_{Q_m}^e(T_{Q_m}^f))$
$Q_s(t)$	set of schedulable requests	$Q_s(t) \subseteq Q(t)$
$P_{d_i}(t)$	effective data productivity of d_i	
$\Lambda_{Q_m}(t)$	effective request productivity	
$\Psi_{Q_m}(t)$	remaining ratio	
$\omega_{Q_m}(t)$	feasible scheduling segment	
$\Omega_{Q_m}(t)$	feasible scheduling period	the length is $ \Omega_{Q_m}(t) $

$E(d_i|t) = U(d_i|t) + l(d_i)$. The set of pending requests in the service queue is denoted by $Q(t)$. The request Q_m ($Q_m \in Q(t)$) is characterized by a 3-tuple: $\langle R(Q_m), S(Q_m), L(Q_m) \rangle$. $R(Q_m)$ is the set of requested data items, and it is represented by $R(Q_m) = \{q_m^1, q_m^2, \dots, q_m^{|R(Q_m)|}\}$, where $|R(Q_m)|$ is the number of requested data items. q_m^n ($1 \leq n \leq |R(Q_m)|$) represents the n th data item requested by Q_m and $q_m^n \in D$. $S(Q_m)$ and $L(Q_m)$ represent the submission time and the deadline of Q_m , respectively. The time taken to broadcast a data item is denoted by τ , which is referred to as the *transmission time*. The primary notations are summarized in Table I.

A. Snapshot consistency requirement

Due to the temporality of data items, the versions of the retrieved dependent data items in a request are expected to be correlated in time. Otherwise, the query result could be meaningless. For example, when a vehicle requests the traffic conditions of two routes, it is expected that the time stamps of these two pieces of information (i.e. the update time of the two data items) are close enough for comparison. In order to guarantee certain correlation among the retrieved data items in a request, we define the snapshot consistency requirement as follows. At time t , the database maintains the latest version for each data item, which is regarded as the *snapshot* of the current database. The versions of the multiple retrieved data items in a request are demanded to be in the same snapshot with respect to a particular time instance, while the time instance is determined by the time when the first data item for this request is disseminated. In other words, when a request retrieves its first data item, the versions of its remaining requested data items need to be in the same snapshot regarding to the first one. The detailed conditions are stated as follows.

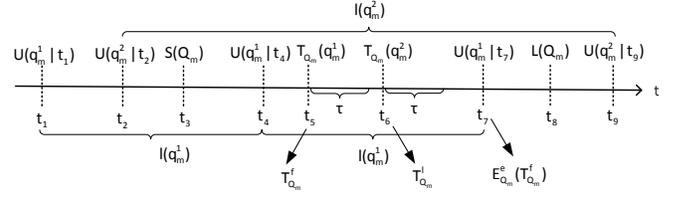


Fig. 2. Relationship of timings on satisfying a request

- The time to disseminate each q_m^n for Q_m (i.e. $T_{Q_m}^f(q_m^n)$) has to be later than the submission time of Q_m (i.e. $S(Q_m)$). This is because vehicles only retrieve data items after submitting their requests. Denote $T_{Q_m}^f$ as the time when the first data item is disseminated for Q_m . The first condition is represented by:

$$T_{Q_m}^f > S(Q_m) \quad (1)$$

where $T_{Q_m}^f = \min(T_{Q_m}^f(q_m^n))$, $\forall q_m^n \in R(Q_m)$.

- Each q_m^n has to be retrieved before the request deadline $L(Q_m)$. The retrieval time of q_m^n is $T_{Q_m}^f(q_m^n) + \tau$, where τ is the data transmission time. Denote $T_{Q_m}^l$ as the time when the last data item is disseminated for Q_m . The second condition is represented by:

$$T_{Q_m}^l + \tau \leq L(Q_m) \quad (2)$$

where $T_{Q_m}^l = \max(T_{Q_m}^f(q_m^n))$, $\forall q_m^n \in R(Q_m)$.

- The version of each retrieved data item has to be in the same snapshot, which is determined when the first data item is disseminated for Q_m (i.e. $T_{Q_m}^f$). Each q_m^n is associated with an expiration time at $T_{Q_m}^f$, which is $E(q_m^n|T_{Q_m}^f)$. Denote the earliest expiration time of these data items as $E_{Q_m}^e(T_{Q_m}^f)$. The third condition is represented by:

$$T_{Q_m}^l + \tau \leq E_{Q_m}^e(T_{Q_m}^f) \quad (3)$$

where $E_{Q_m}^e(T_{Q_m}^f) = \min(E(q_m^n|T_{Q_m}^f))$, $\forall q_m^n \in R(Q_m)$.

Figure 2 illustrates the relationship of timings for serving a request. Suppose Q_m is submitted at t_3 , and the requested data set $R(Q_m) = \{q_m^1, q_m^2\}$. The deadline of Q_m is t_8 . q_m^1 is updated at t_1, t_4 and t_7 with the update interval of $l(q_m^1)$, while q_m^2 is updated at t_2 and t_9 with the update interval of $l(q_m^2)$. To serve Q_m , according to the first condition, the broadcast time of q_m^1 and q_m^2 should be later than $S(Q_m)$ (i.e. t_3). Since q_m^1 is the first data item broadcasted for Q_m , we have $T_{Q_m}^f = t_5$. This satisfies the first condition as $t_5 > t_3$. According to the second condition, both q_m^1 and q_m^2 have to be retrieved before $L(Q_m)$ (i.e. t_8). Since q_m^2 is the last data item broadcasted for Q_m , we have $T_{Q_m}^l = t_6$. This satisfies the second condition as $t_6 + \tau \leq t_8$. According to the third condition, the versions of q_m^1 and q_m^2 have to be consistent regarding to the snapshot at time $T_{Q_m}^f$, which is t_5 . Since $E(q_m^1|t_5) = t_7$ and $E(q_m^2|t_5) = t_9$, the earliest expiration time $E_{Q_m}^e(T_{Q_m}^f) = t_7$. This satisfies the third condition as $t_6 + \tau \leq t_7$. With the satisfaction of the three conditions, Q_m can be satisfied by such a schedule.

Suppose the duration between t_3 and t_4 is less than the time of transmitting two data items (i.e. $[t_3, t_4] < 2 \cdot \tau$), then

Q_m could not be satisfied if either q_m^1 or q_m^2 is disseminated during $[t_3, t_4]$. This is because if $t_3 < T_{Q_m}^f < t_4$, the expiration time of q_m^1 at this snapshot (i.e. $E(q_m^1 | T_{Q_m}^f)$) would be t_4 . So, the earliest expiration time (i.e. $E_{Q_m}^e(T_{Q_m}^f)$) is computed by $\min(t_4, t_9) = t_4$. However, it cannot satisfy $T_{Q_m}^l + \tau \leq t_4$ due to $[t_3, t_4] < 2 \cdot \tau$. Therefore, it violates the third condition and Q_m cannot be served by such a schedule. This example indicates that even for serving a single request, it is not always wise to start the service as early as possible.

B. NP-hardness

The *Temporal Data Dissemination* (TDD) problem is specified as follows.

Instance: There are M requests Q_1, Q_2, \dots, Q_M and $R(Q_m)$ is the set of data items requested by Q_m ($1 \leq m \leq M$). Each data item d_i ($d_i \in D$) has an update interval of $l(d_i)$. The submission time and the deadline of Q_m are $S(Q_m)$ and $L(Q_m)$, respectively. The data transmission time is τ .

Question: Is there a schedule which can satisfy at least C ($C \in \mathbb{Z}^+$) requests?

We prove TDD is NP-hard by constructing a polynomial time reduction from a well known NP-hard problem, namely, the Clique problem [16], to a special instance of TDD.

Proof. Given an instance of Clique $\langle G, k \rangle$ for determining whether there is a clique of size k in a graph G , we consider a special instance of TDD as follows to complete the reduction. For any Q_m , let $|R(Q_m)| = 2$, $S(Q_m) = 0$ and $L(Q_m) = k$. For any d_i , let $l(d_i) = k$ and $U(d_i|0) = 0$. Let the data transmission time $\tau = 1$ and the target $C = \frac{k \cdot (k-1)}{2}$. With the above settings, for a graph $G = \langle V, E \rangle$, we construct a one-to-one mapping from each vertex v_i ($v_i \in V$) to each requested data item d_i ($d_i \in D$). In addition, for any Q_m with the requested data items d_i and d_j , we construct an edge e_m between the two corresponding vertices v_i and v_j , which gives a one-to-one mapping between a request Q_m and an edge e_m . Consider different requests asking for different sets of data items, namely, $R(Q_i) \neq R(Q_j)$ ($1 \leq i, j \leq M$ and $i \neq j$). According to the mapping, there is at most one edge between two vertices. An example of this reduction is illustrated in Figure 3. Suppose $k = 4$, for the Clique problem, the question is whether there is a clique with at least the size of 4. For the TDD problem, since $C = \frac{k \cdot (k-1)}{2} = 6$, the question is whether there is a schedule that can satisfy at least 6 requests. In this example, the answers to both the questions are Yes. Specifically, for the Clique problem, there is $V' = \{v_1, v_3, v_4, v_5\} \subset V$, which forms a clique with the size of 4. For the TDD problem, there is a schedule of disseminating d_1, d_3, d_4, d_5 , which can satisfy 6 requests (i.e. Q_3, Q_4, Q_5, Q_6, Q_7 and Q_8).

In the following, we prove that with the above polynomial time reduction, the graph G has a clique with at least size k if and only if there is a schedule that can satisfy at least C requests, where $C = \frac{k \cdot (k-1)}{2}$.

Suppose G has a clique $V' \subseteq V$ and $|V'| = k$. According to the mapping, there are k corresponding data items, which are requested by $\frac{k \cdot (k-1)}{2}$ requests. Each request asks for two out of these k data items. Since $\tau = 1$ and $L(Q_m) = k$ for any

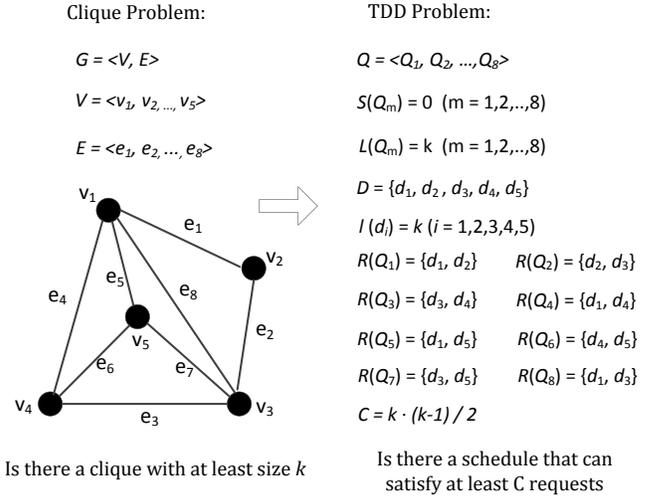


Fig. 3. An example of reduction from the Clique problem to the TDD problem

Q_m , when k data items are disseminated, none of the request will miss its deadline. Besides, the expiration time of each d_i is computed by $E(d_i|0) = U(d_i|0) + l(d_i)$, which equals k according to the default setting. So, the earliest expiration time $E_{Q_m}^e(0) = k$. Therefore, disseminating k data items will not violate the snapshot consistency requirement. To sum up, all the $\frac{k \cdot (k-1)}{2}$ requests can be satisfied. So, given a clique of size k , there is a schedule that can satisfy at least $\frac{k \cdot (k-1)}{2}$ requests.

Conversely, suppose there is a schedule that can satisfy at least $\frac{k \cdot (k-1)}{2}$ requests. According to the mapping, there is a sub-graph $G' = \langle V', E' \rangle$, which consists of $\frac{k \cdot (k-1)}{2}$ edges. Since there is at most one edge between two vertices, given k vertices, the number of edges is bounded by $C_k^2 = \frac{k \cdot (k-1)}{2}$. Therefore, at least k vertices are required, which implies $|V'| \geq k$. In addition, since $L(Q_m) = k$ and $l(d_i) = k$, there are at most k time units (i.e. k data items can be disseminated) to serve these requests. Therefore, G' contains at most k vertices, which implies $|V'| \leq k$. To sum up, we have $|V'| = k$ and G' is a complete sub-graph of G . So, given a schedule that can satisfy at least $\frac{k \cdot (k-1)}{2}$ requests, there is a clique of size k .

The above proves the NP-hardness of a special case of the TDD problem. Accordingly, TDD is NP-hard \square

IV. ALGORITHM DESIGN

A. Time bound analysis

Due to the broadcast effect, some requests may be partially served before being scheduled. Specifically, these requests may have retrieved part of their data items, which are broadcasted to serve other scheduled requests. In view of this, we define the *unserved set of a request* as follows.

Definition 1. *Unserved set of a request.* At time t , the set of unserved data items of a request Q_m is represented by $R_{Q_m}^u(t) = \{q_m^1, q_m^2, \dots, q_m^{|R_{Q_m}^u(t)|}\}$, where $|R_{Q_m}^u(t)|$ is the number of unserved data items ($0 < |R_{Q_m}^u(t)| \leq |R(Q_m)|$) and $R_{Q_m}^u(t) \subseteq R(Q_m)$.

Given Q_m , if none of its requested data items have been retrieved (i.e., $R_{Q_m}^u(t) = R(Q_m)$), then Q_m is called the *unserved request*. In contrast, if part of the requested data items have been retrieved (i.e., $R_{Q_m}^u(t) \subset R(Q_m)$ and $R_{Q_m}^u(t) \neq \emptyset$), then Q_m is called the *partially-served request*. Recall that each request is associated with a deadline. Meanwhile, each data item is associated with an expiration time. Therefore, there is a practical time bound to serve a request when considering both the request deadline and the data expiration time. Indeed, for unserved requests and partially-served requests, they have different attributes in terms of the time bound. We analyze this issue by introducing the *tentative time bound* and the *determined time bound* for unserved requests and partially-served requests, respectively.

Definition 2. *Tentative time bound.* At time t , if Q_m is an unserved request, then the tentative time bound for Q_m ($\chi_{Q_m}(t)$) is either its request deadline ($L(Q_m)$), or the earliest expiration time of its requested data items at time t ($E_{Q_m}^e(t)$), whichever is earlier. That is:

$$\chi_{Q_m}(t) = \min(L(Q_m), E_{Q_m}^e(t)) \quad (4)$$

Note that $\chi_{Q_m}(t)$ is not the finalized time bound for Q_m and the value of $\chi_{Q_m}(t)$ may change with time. This is because although the request deadline $L(Q_m)$ has been determined since the submission of Q_m , the value of $E_{Q_m}^e(t)$ may vary with time. Therefore, at different scheduling points, the dynamic value of $E_{Q_m}^e(t)$ may result in different values of $\chi_{Q_m}(t)$. Consider the example shown in Figure 2 and suppose $t_3 < t < t_4$. We have $E_{Q_m}^e(t) = \min(E(q_m^1|t), E(q_m^2|t)) = t_4$. Accordingly, $\chi_{Q_m}(t) = \min(L(Q_m), t_4)$, which equals t_4 . In contrast, if $t_4 < t < t_7$, then we have $E_{Q_m}^e(t) = \min(E(q_m^1|t), E(q_m^2|t)) = t_7$, and hence the value of $\chi_{Q_m}(t)$ changes to $\min(L(Q_m), t_7) = t_7$.

Note that the tentative time bound is only the attribute of unserved requests. As soon as the service starts for a request, due to the snapshot consistency requirement, the versions of all its data items are determined. This gives a *determined time bound* for the partially-served request.

Definition 3. *Determined time bound.* If Q_m is a partially-served request, then the determined time bound for Q_m (X_{Q_m}) is either its request deadline $L(Q_m)$, or the earliest expiration time of its requested data items at time $T_{Q_m}^f$ (i.e., $E_{Q_m}^e(T_{Q_m}^f)$), whichever is earlier. That is:

$$X_{Q_m} = \min(L(Q_m), E_{Q_m}^e(T_{Q_m}^f)) \quad (5)$$

where $T_{Q_m}^f$ is the time when the first data item is disseminated for Q_m .

Since both $L(Q_m)$ and $E^e(q_m^n|T_{Q_m}^f)$ have been fixed at time $T_{Q_m}^f$, X_{Q_m} is a static value. With the above time bound analysis, we can find those requests which have chance to be satisfied at a specific time t , which are called the *schedulable request*. At time t , denote $Q_s(t)$ as the set of schedulable requests. Based on the service status of requests, $Q_s(t)$ can be divided into two subsets (i.e., $Q_{s1}(t)$ and $Q_{s2}(t)$), which are obtained as follows.

- a) For an unserved Q_m , it is schedulable if Q_m can retrieve all of its data items before the tentative time

bound $\chi_{Q_m}(t)$. The subset is represented by:

$$Q_{s1}(t) = \{Q_m | R_{Q_m}^u(t) = R(Q_m) \wedge t + |R(Q_m)| \cdot \tau \leq \chi_{Q_m}(t)\} \quad (6)$$

- b) For a partially-served Q_m , it is schedulable if Q_m can retrieve its remaining data items before the determined time bound X_{Q_m} . The subset is represented by:

$$Q_{s2}(t) = \{Q_m | R_{Q_m}^u(t) \subset R(Q_m) \wedge R_{Q_m}^u(t) \neq \emptyset \wedge t + |R_{Q_m}^u(t)| \cdot \tau \leq X_{Q_m}\} \quad (7)$$

To sum up, the set of schedulable requests is obtained by:

$$Q_s(t) = Q_{s1}(t) \cup Q_{s2}(t) \quad (8)$$

B. PSU Algorithm

As the broadcast is an intrinsic nature of wireless communication in VCPS, the scheduling is expected to exploit the broadcast effect for data dissemination and enhance the system scalability. Besides, as a request may correspond to multiple dependent data items for completing the service, it is critical to consider the bandwidth efficiency in terms of satisfying multi-item requests. Last, due to the time-constraint of services and the temporality of data items, it is expected to serve as many vehicles as possible before their respective service time bounds. With the above motivation, we propose a heuristic algorithm called PSU, which considers request characteristics of *Productivity*, *Status* and *Urgency* in scheduling.

1) *Objectives:* The primary objectives of PSU include exploiting the broadcast effect, improving the bandwidth efficiency and enhancing the request service.

- Exploiting the broadcast effect: In roadside-to-vehicle communication, a data item broadcasted from the RSU can be retrieved by all the vehicles within its coverage, which is called the *broadcast effect*. Clearly, it is expected to exploit such a benefit to enhance the system scalability. Conventionally, scheduling a data item with the largest number of pending requests would be the best option to maximize the broadcast benefit. However, as analyzed in Section IV-A, not all the requests are schedulable at a scheduling point due to the temporality of data items. Therefore, a more sophisticated approach is expected to effectively exploit the broadcast effect.
- Improving the bandwidth efficiency: Due to the application requirement in VCPS, a request may ask for multiple dependent data items for query processing. In this case, it is a waste of bandwidth if a request retrieved part of its data items but could not be completely served before its time bound. Previous studies ([17], [18]) have revealed that there is a trade-off between maximizing the broadcast effect and improving the bandwidth efficiency for serving multi-item requests. Specifically, it will cause the starvation problem if an algorithm simply schedules data items with large number of pending requests but ignores the fact that a multi-item request has to retrieve all of its data items before the time bound. To address the starvation problem and avoid the waste of bandwidth,

the algorithm should consider the service status of each multi-item request in scheduling.

- Enhancing the request service: due to the time-constraint of services and the temporality of data items, there is a stringent time bound for serving requests. In order to enhance the service chance of requests, conventionally, the algorithms consider either the deadline (e.g. EDF [12]) or the slack time (e.g. SIN [6]) in scheduling to capture the attribute of request urgency. Nevertheless, these solutions cannot be directly applied into this new setting, because as analyzed in Def.2, the time bound cannot be determined for unserved requests (i.e. the tentative time bound). Therefore, it is desirable to design new metrics to reflect the request urgency and enhance the request service.

2) *Metrics*: For exploiting the broadcast effect, the *effective data productivity* and the *effective request productivity* are defined as follows.

Definition 4. *Effective data productivity.* At time t , denote $Q_{d_i}(t)$ as the set of requests which satisfies the following two conditions: a) d_i is in the unserved set of Q_m , namely, $d_i \in R_{Q_m}^u(t)$. b) Q_m is schedulable at time t , namely, $Q_m \in Q_s(t)$. The effective data productivity of d_i ($1 \leq i \leq |D|$), denoted by $P_{d_i}(t)$, is defined as the number of requests in $Q_{d_i}(t)$. It is represented by:

$$P_{d_i}(t) = |Q_{d_i}(t)| \quad (9)$$

It has been demonstrated that it is effective in serving multi-item requests by broadcasting all the unserved data items of a request successively [18], namely, scheduling at the request level. In this regard, the *effective request productivity* is defined as follows.

Definition 5. *Effective request productivity.* At time t , the effective request productivity of Q_m , denoted by $\Lambda_{Q_m}(t)$, is the average of the effective data productivity of its unserved data items, which is computed by:

$$\Lambda_{Q_m}(t) = \sum_{d_i \in R_{Q_m}^u(t)} P_{d_i}(t) / |R_{Q_m}^u(t)| \quad (10)$$

Selecting requests with higher values of $\Lambda_{Q_m}(t)$ can better exploit the broadcast effect.

For improving the bandwidth efficiency, the *remaining ratio* is defined as follows to capture the service status of requests.

Definition 6. *Remaining ratio.* At time t , the remaining ratio of Q_m , denoted by $\Psi_{Q_m}(t)$, is the number of its unserved data items over the total number of its requested data items, which is computed by:

$$\Psi_{Q_m}(t) = \frac{|R_{Q_m}^u(t)|}{|R(Q_m)|} \quad (11)$$

A small value of the remaining ratio implies that a large percentage of the requested data items have been retrieved and the request is close to be satisfied. Giving higher priority to a request with a smaller value of remaining ratio will help to improve the bandwidth efficiency. We justify this claim by the following two cases. Case 1: Consider two

requests Q_m and Q_n , which ask for the same number of data items ($|R(Q_m)| = |R(Q_n)|$). Suppose $\Psi_{Q_m}(t) < \Psi_{Q_n}(t)$, which implies that Q_m has fewer unserved data items than Q_n (i.e. $|R_{Q_m}^u(t)| < |R_{Q_n}^u(t)|$). So, it requires less bandwidth to complete the service of Q_m . Case 2: Consider two requests Q_m and Q_n , which have the same number of unserved data items ($|R_{Q_m}^u(t)| = |R_{Q_n}^u(t)|$). Suppose $\Psi_{Q_m}(t) < \Psi_{Q_n}(t)$, which implies that Q_m asks for more data items than Q_n . Accordingly, more bandwidth has been consumed to serve Q_m (i.e. $|R(Q_m)| - |R_{Q_m}^u(t)| > |R(Q_n)| - |R_{Q_n}^u(t)|$). So, completing the service of Q_m will make the previously consumed bandwidth for Q_m count, which means more cost effective.

For enhancing the request service, we define the metric called *feasible scheduling period* to capture the request urgency. First, we introduce the concept of *feasible and non-feasible scheduling segments*. At time t , depending on the service status of Q_m , there are two cases for feasible/non-feasible scheduling segments.

- Q_m is an unserved request: if Q_m is schedulable (i.e., $t + |R(Q_m)| \cdot \tau \leq \chi_{Q_m}(t)$), then $[t, \chi_{Q_m}(t)]$ is a *feasible scheduling segment*. Otherwise, $[t, \chi_{Q_m}(t)]$ is a *non-feasible scheduling segment*.
- Q_m is a partially-served request: if Q_m is schedulable (i.e., $t + |R_{Q_m}^u(t)| \cdot \tau \leq X_{Q_m}$), then $[t, X_{Q_m}]$ is a *feasible scheduling segment*. Otherwise, $[t, X_{Q_m}]$ is a *non-feasible scheduling segment*.

Note that in Case b), $[t, X_{Q_m}]$ is the only possible feasible/non-feasible scheduling segment. However, in Case a), it may have multiple feasible/non-feasible scheduling segments. Recall the example shown in Figure 2. The tentative time bound $\chi_{Q_m}(t) = t_4$ if $t_3 < t < t_4$. At this time, $[t, \chi_{Q_m}(t)]$ is a non-feasible scheduling segment because $[t, t_4] < 2 \cdot \tau$. Since $t_4 < L(Q_m)$, the algorithm proceed to examine other segments by setting t' to t_4 . At t' , the new tentative time bound $\chi_{Q_m}(t') = t_7$. Clearly, $[t', \chi_{Q_m}(t')]$ is a feasible scheduling segment. Finally, when setting t'' to t_7 , the current tentative time bound $\chi_{Q_m}(t'') = t_8$, and $[t'', \chi_{Q_m}(t'')]$ is a non-feasible scheduling segment (suppose $[t_7, t_8] < 2 \cdot \tau$). As t_8 equals the request deadline $L(Q_m)$, which is the latest time for Q_m to be served, it cannot move forward to check other segments. With the above knowledge, we define the *feasible scheduling period* of Q_m as follows.

Definition 7. *Feasible scheduling period.* At time t , the feasible scheduling period for Q_m , denoted by $\Omega_{Q_m}(t)$, is the union of the feasible scheduling segments ($\omega_{Q_m}(t)$) in $[t, L(Q_m)]$. It is represented by:

$$\Omega_{Q_m}(t) = \cup \omega_{Q_m}(t) \text{ in } [t, L(Q_m)] \quad (12)$$

For a partially-served Q_m , the only possible feasible scheduling segment is $[t, X_{Q_m}]$. Therefore, $\Omega_{Q_m}(t)$ is either $[t, X_{Q_m}]$ or \emptyset , depending on whether $[t, X_{Q_m}] \subseteq \omega_{Q_m}(t)$. For an unserved Q_m , $\Omega_{Q_m}(t)$ is obtained by combining all the feasible scheduling segments of Q_m at t . Note that these feasible scheduling segments may not be consecutive. The operation to compute $\Omega_{Q_m}(t)$ for unserved requests is described as follows.

- Set the feasible scheduling period to the null set ($\Omega_{Q_m}(t) \leftarrow \emptyset$), and set t' to the current time t ($t' \leftarrow t$).

- Check whether $[t', \chi_{Q_m}(t')]$ is a feasible scheduling segment. If it is true, add $[t', \chi_{Q_m}(t')]$ to $\Omega_{Q_m}(t)$ (i.e., $\Omega_{Q_m}(t) \leftarrow \Omega_{Q_m}(t) \cup [t', \chi_{Q_m}(t')]$).
- Check whether $\chi_{Q_m}(t')$ equals the request deadline $L(Q_m)$. If it is true, the operation is terminated. Otherwise (i.e., $\chi_{Q_m}(t') < L(Q_m)$), set t' to the current tentative time bound (i.e., $t' \leftarrow \chi_{Q_m}(t')$).
- Repeat the update of t' until the tentative time bound equals the request deadline (i.e., $\chi_{Q_m}(t') = L(Q_m)$).

The length of feasible scheduling period $|\Omega_{Q_m}(t)|$ reflects the actual duration in which a request can be served. In particular, for a schedulable request, the shorter of $|\Omega_{Q_m}(t)|$ is, the more urgent of the request is.

3) *Scheduling policy*: With the above metric design and analysis, we have the following three observations.

- To exploit the broadcast effect, the algorithm should give a higher priority to the request with a larger value of the effective request productivity.
- To improve the bandwidth efficiency, the algorithm should give a higher priority to the request with a smaller value of the remaining ratio.
- To enhance the request service, the algorithm should give a higher priority to the request with a shorter length of the feasible scheduling period.

Accordingly, the request priority is a compound effect of the effective request productivity ($\Lambda_{Q_m}(t)$), the remaining ratio ($\Psi_{Q_m}(t)$) and the length of feasible scheduling period ($|\Omega_{Q_m}(t)|$), which is defined as follows.

Definition 8. *Request priority.* At time t , the priority of Q_m is computed by:

$$Priority_{Q_m}(t) = \frac{\Lambda_{Q_m}(t)}{\Psi_{Q_m}(t) \cdot |\Omega_{Q_m}(t)|} \quad (13)$$

PSU consists of three steps. Step 1, it constructs the set of schedulable requests $Q_s(t)$ by analyzing the time bound. Step 2, it computes the priority of requests in $Q_s(t)$ and schedules the one with the highest priority. Step 3, it broadcasts each of the unserved data item for the scheduled request and updates the status of pending requests in the service queue. The pseudo-code of PSU is illustrated in Algorithm 1.

4) *Computation complexity*: It is commonly to evaluate the computation complexity of the algorithm by measuring the number of data items to be examined in each scheduling point ([5], [6]). Suppose there are n requests in the service queue and each request asks for s data items. The computation complexity of PSU is analyzed as follows. First, in order to construct the set of schedulable requests $Q_s(t)$, PSU traverses the service queue and computes the time bound of each request. For this operation, at most s data items will be examined for a request. Accordingly, the total number of data items to be examined is bounded by $s \cdot n$. Second, PSU computes the priority of each request in $Q_s(t)$ and selects the one with the highest priority. As $|Q_s(t)| \leq n$, likewise, there are at most $s \cdot n$ data items need to be examined. Once the request is selected, its unserved data items are scheduled to broadcast successively, where at most s data items are examined, and the maintenance of the service queue examines at most n

Algorithm 1 PSU

```

Step 1: Construct the set of schedulable requests  $Q_s(t)$ 
1:  $Q_s(t) \leftarrow \emptyset$ ;
2: for each unserved request  $Q_m \in Q(t)$  do
3:   if  $t + |R(Q_m)| \cdot \tau \leq \chi_{Q_m}(t)$  then
4:      $Q_s(t) \leftarrow Q_s(t) \cup \{Q_m\}$ ;
5:   end if
6: end for
7: for each partially-served request  $Q_m \in Q(t)$  do
8:   if  $t + |R_{Q_m}^u(t)| \cdot \tau \leq X_{Q_m}$  then
9:      $Q_s(t) \leftarrow Q_s(t) \cup \{Q_m\}$ ;
10:  end if
11: end for
Step 2: Schedules the request with the highest priority
12:  $maxPriority \leftarrow 0$ ;
13: for each  $Q_m \in Q_s(t)$  do
14:    $Priority_{Q_m}(t) = \frac{\Lambda_{Q_m}(t)}{\Psi_{Q_m}(t) \cdot |\Omega_{Q_m}(t)|}$ ;
15:   if  $maxPriority < Priority_{Q_m}(t)$  then
16:      $maxPriority \leftarrow Priority_{Q_m}(t)$ ;
17:      $Q_{selected} \leftarrow Q_m$ ;
18:   end if
19: end for
Step 3: Broadcast data items for the selected request and update
the service queue
20: broadcast each  $d_i \in R_{Q_{selected}}^u(t)$ ;
21: for each  $Q_n \in Q_s(t)$  do
22:   if  $d_i \in R_{Q_n}^u$  then
23:      $R_{Q_n}^u(t) \leftarrow R_{Q_n}^u(t) - \{d_i\}$ ;
24:     if  $d_i$  is the first retrieved data item of  $Q_n$  then
25:       Compute the determined time bound  $X_{Q_n}$ ;
26:     end if
27:     if  $R_{Q_n}^u(t) == \emptyset$  then
28:        $Q(t) \leftarrow Q(t) - \{Q_n\}$ ;
29:     end if
30:   end if
31: end for
32: for each  $Q_n \in Q(t)$  do
33:   if  $t > L(Q_n)$  then
34:      $Q(t) \leftarrow Q(t) - \{Q_n\}$ ;
35:   end if
36: end for

```

requests for updating. To sum up, the computation complexity of PSU is $O(s \cdot n)$. Note that existing representative scheduling algorithms have the computation complexity of $O(n)$ [19]. In practice, given a specific application, s is normally a fixed number (or within a fixed range), and it is bounded by the size of the database ($|D|$), whereas n is a variable which increases with the system scale. Therefore, the variable n is the most critical factor to be considered for system scalability. Moreover, unlike previous algorithms ([6], [12]), which have to make a scheduling decision in every broadcast tick, PSU schedules at the request level (i.e., makes a scheduling decision in every several broadcast ticks, depending on the number of unserved data items in the selected request). Therefore, the scheduling frequency of PSU is much lower. Overall, the scheduling overhead of PSU is reasonable and it will not be a hurdle of the system scalability.

TABLE II
DEFAULT SETTING

Parameter	Default	Description
$ D $	100	number of temporal data items
$\frac{1}{\lambda}$	0.6	mean inter-arrival time of requests
s	3	request size
L_{min}	50	minimum tolerated latency
L_{max}	70	maximum tolerated latency
T_{min}	200	minimum update period
T_{max}	300	maximum update period
θ	0.6	Zipf distribution parameter

V. PERFORMANCE EVALUATION

A. Preliminaries

The simulation model is implemented by CSIM19 [20], which captures the data dissemination characteristics as described in Section II. The inter-arrival time of requests follows the Exponential distribution with mean value of $\frac{1}{\lambda}$. Each request may ask for multiple dependent data items and the request size (s) is the number of required data items. Each request is associated with a deadline, which is obtained by $t_{arrival} + t_{relative}$, where $t_{arrival}$ is the submission time of the request, and $t_{relative}$ is the relative deadline of the request. For general purposes, the value of $t_{relative}$ is uniformly selected from the range (L_{min}, L_{max}) , where L_{min} and L_{max} represent the minimum and the maximum tolerated latency, respectively, for serving requests. The database D consists of $|D|$ data items. Each d_i ($d_i \in D$) has an update interval of $l(d_i)$, which is uniformly generated from the range (T_{min}, T_{max}) , where T_{min} and T_{max} represent the minimum and the maximum update period of data items, respectively. The data access pattern follows the commonly used Zipf distribution [21] with a skewness parameter θ . The time unit (i.e., a broadcast tick) refers to the data transmission time. The main parameters and the corresponding descriptions are summarized in Table II. Unless stated otherwise, the simulations are conducted under the default setting.

For performance comparison, we implement two well-known real-time scheduling algorithms. One is EDF (Earliest Deadline First) [12] and the other is SIN (Slack time Inverse Number of pending requests) [6]. The statistics including the total number of submitted requests (N_{sub}), the number of satisfied requests (N_{sat}) and the number of failed requests (N_{fail}) are captured for performance analysis, where $N_{sat} + N_{fail} = N_{sub}$. Moreover, we classify the failed requests into two parts. One part is caused by missing deadlines (the number is denoted by N_{miss}), and the other part is caused by data expiration (the number is denoted by N_{exp}). Accordingly, we have $N_{miss} + N_{exp} = N_{fail}$. With the collected statistics, the following criteria are adopted for performance evaluation.

- Service ratio: It is the ratio of the number of satisfied requests to the total number of submitted requests, which is computed by $\frac{N_{sat}}{N_{sub}}$. The primary objective of a scheduling algorithm is to maximize the service ratio.
- Bandwidth efficiency ratio: In order to measure the bandwidth efficiency quantitatively, the bandwidth efficiency ratio is computed by $\frac{N_{rev} - N_{bst}}{N_{req}}$, where N_{rev} is the total

number of data items received by satisfied requests. N_{bst} is the number of broadcasted data items, and N_{req} is the total number of data items required by all the requests. We give an example to explain this criterion. Assume there are four pending requests $Q_1 = \{a, b\}$, $Q_2 = \{a, b\}$, $Q_3 = \{a, c\}$ and $Q_4 = \{c, d\}$, and all of them have to be served in 2 broadcast ticks. Case a): if a and b are scheduled to broadcast, then Q_1 and Q_2 would be satisfied, while Q_3 and Q_4 would be failed. In this case, $N_{bst} = 2$, $N_{req} = 8$ and $N_{rev} = 4$. Therefore, the bandwidth efficiency ratio is $(4 - 2)/8 = 1/4$. A positive value means that this schedule has positive contribution to the bandwidth efficiency, because each broadcast tick contributes to multiple satisfied requests. Case b): if a and c are scheduled to broadcast, then only Q_3 could be satisfied. In this case, the values of N_{bst} and N_{req} remain the same, but the value of N_{rev} is reduced to 2. Therefore, the bandwidth efficiency ratio is $(2 - 2)/8 = 0$. A zero value means that this schedule has no contribution to the bandwidth efficiency, because each broadcast tick contributes to exactly one satisfied request. Case c): if a and d are scheduled to broadcast, then none of the requests could be satisfied. In this case, the value of N_{rev} becomes 0, and the bandwidth efficiency ratio is $(0 - 2)/8 = -1/4$. A negative value means that this schedule has negative contribution to the bandwidth efficiency, because each broadcast tick contributes to less than one satisfied request. To sum up, this criterion reflects the efficiency of allocating bandwidth for satisfying requests.

- Proportion of missing deadline to data expiration: Since $N_{fail} = N_{miss} + N_{exp}$, the percentage of failed requests due to missing deadline (P_{miss}) is computed by $P_{miss} = \frac{N_{miss}}{N_{fail}}$, while the percentage of failed requests due to data expiration (P_{exp}) is computed by $P_{exp} = \frac{N_{exp}}{N_{fail}}$, where $P_{miss} + P_{exp} = 1$. This criterion examines the proportion between P_{miss} and P_{exp} , which indicates the influence of these two reasons to the overall performance.
- Data expiration ratio: It is the ratio of the number of failed requests due to data expiration (N_{exp}) to the total number of submitted requests (N_{sub}), which is computed by $\frac{N_{exp}}{N_{sub}}$. Different from the proportion of missing deadline to data expiration, which focuses on examining the weights of the two reasons causing service failure, this criterion concerns the impact of data expiration on overall scheduling performance. It evaluates the algorithm performance in terms of satisfying the snapshot consistency requirement.

B. Experimental results and analysis

To emphasize the general applicability of the performance analysis, we do not specify the absolute values of the data size and the bandwidth, but rather, use the broadcast tick as unit for comparing relative performance of different algorithms.

1) *Effect of relative deadline*: Figure 4 shows the service ratio of algorithms under different relative deadlines. PSU achieves the highest service ratio across the whole range. Also, note that even though the relative deadline is getting looser, none of the algorithm can achieve a 100% service ratio. The

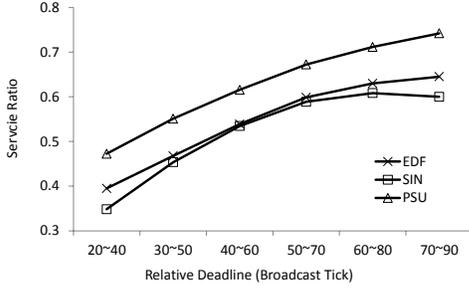


Fig. 4. Service ratio under different relative deadlines

statistics shown in Figure 5 explains the reason. When the relative deadline is getting looser, although the number of failed requests due to missing deadline is decreased, more requests are failed because of data expiration. It demonstrates that the snapshot consistency requirement has higher impact to the algorithm performance in a looser relative deadline environment. The data expiration ratio of algorithms under different relative deadlines is shown in Figure 6. PSU performs the best in terms of satisfying the snapshot consistency requirement, especially in a looser relative deadline environment. Figure 7 examines the bandwidth efficiency of algorithms under different relative deadlines, which shows that PSU significantly outperforms other algorithms in terms of improving the bandwidth efficiency.

2) *Effect of data update*: Figure 8 shows the service ratio of algorithms under different data update periods. The short update period implies a harsh condition on satisfying the snapshot consistency requirement. PSU always outperforms EDF and SIN, especially in a short update period. This is because PSU prevents from selecting those requests which may retrieve inconsistent versions of data items by identifying schedulable requests. The proportion of missing deadline to data expiration is shown in Figure 9. As expected, the data expiration dominates the performance of all the algorithms when the data update period is short. In particular, for EDF and SIN, when the update period ranges between 50 and 150 broadcast ticks, almost 100% of request failure is attributed to data expiration. Figure 10 shows the data expiration ratio of algorithms under different update periods. It further confirms that PSU has the best performance in terms of satisfying the snapshot consistency requirement, especially in a highly dynamic vehicular environment where the data items are updated very frequently. Figure 11 shows the bandwidth efficiency ratio of algorithms under different data update periods. It demonstrates that no matter whether the missing deadline or the data expiration is the dominate factor, PSU achieves the highest bandwidth efficiency.

3) *Effect of request size*: For comparison purpose, we maintain a constant system workload across different request sizes by adjusting the request arrival rate. For example, when the request size is 1, the mean request arrival rate (λ) is adjusted to 5 (requests/tick) to maintain the same system workload with the default setting. Figure 12 shows the service ratio of algorithms under different request sizes. When the request size equals 1, SIN has higher service ratio than EDF

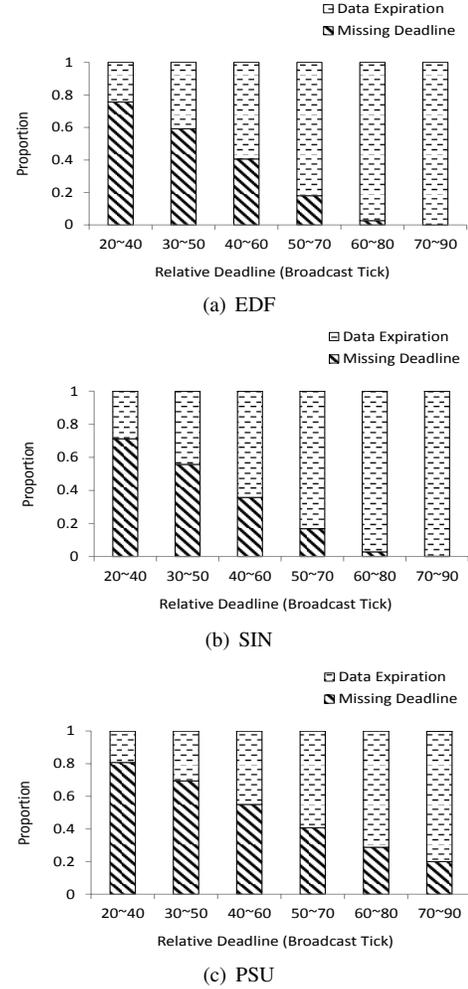


Fig. 5. Proportion of missing deadline to data expiration under different relative deadlines

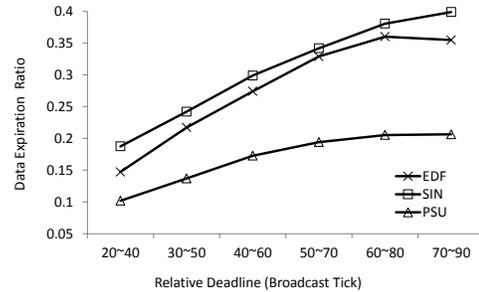


Fig. 6. Data expiration ratio under different relative deadlines

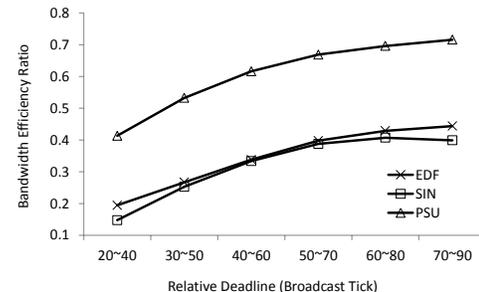


Fig. 7. Bandwidth efficiency ratio under different relative deadlines

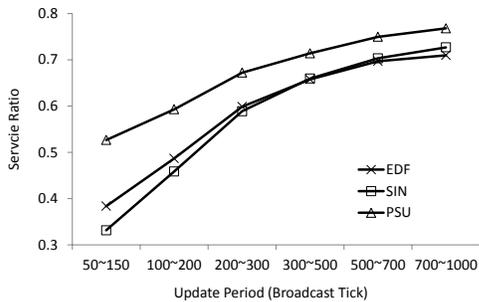


Fig. 8. Service ratio under different data update periods

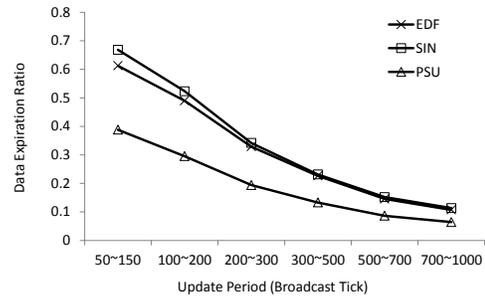
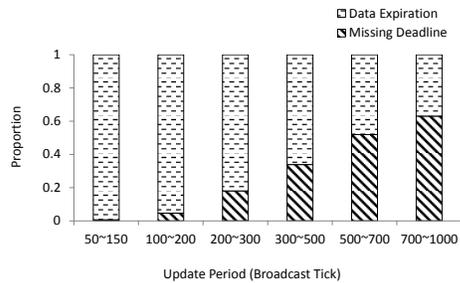
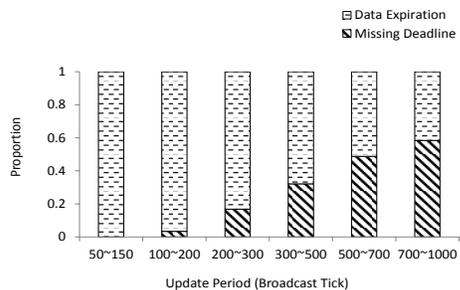


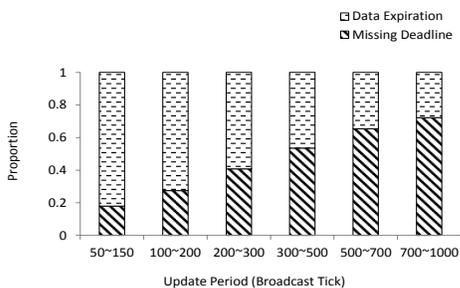
Fig. 10. Data expiration ratio under different data update periods



(a) EDF



(b) SIN



(c) PSU

Fig. 9. Proportion of missing deadline to data expiration under different data update periods

and it achieves similar performance with PSU. However, when the request size increases, the performance of SIN drops drastically. These results are consistent with the observations in previous studies ([6], [18]). Note that even though the system workload remains, the performance of all the algorithms drops to different extent when the request size increases. This is because the more data items are in a request, the more difficult to satisfy the snapshot consistency requirement. In other words, it is likely that more requests cannot be served due to data expiration. The result shown in Figure 13 verifies

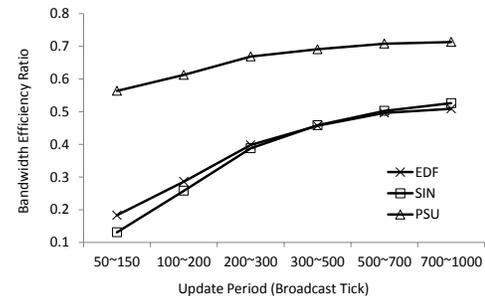


Fig. 11. Bandwidth efficiency ratio under different data update periods

this claim. We note that when the request size increases to 6, for EDF and SIN, around 60% and 70% of requests are failed, respectively, due to data expiration. Nevertheless, PSU maintains a decent data expiration ratio around 40%.

C. Performance validation

In order to validate the effectiveness of PSU in vehicular environments, we examine the algorithm performance by modeling the following application scenario. The RSU is installed along the highway and provides location-dependent services such as traffic conditions to passing vehicles. The data service has to be completed before vehicles leave the coverage of the RSU.

Both the traffic and the communication characteristics are considered to simulate a proper vehicular environment. In particular, the traffic characteristic is simulated according to the Greenshield's model [22], which is widely adopted in macroscopic traffic models [23]. Specifically, the relationship between the vehicle speed (v) and the density (k) is represented

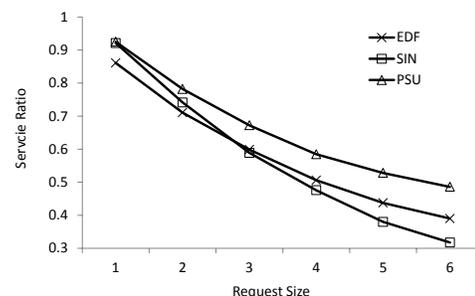


Fig. 12. Service ratio under different request sizes

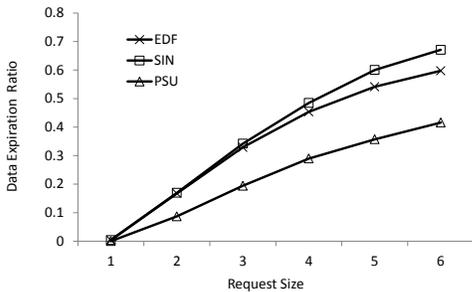


Fig. 13. Data expiration ratio under different request sizes

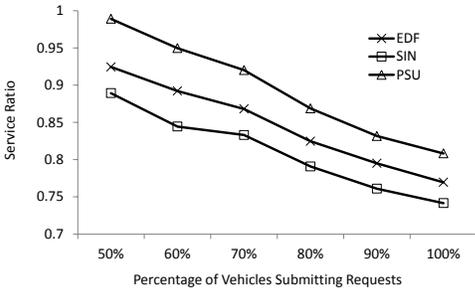


Fig. 14. Service ratio under different percentages of vehicles submitting requests

by $v = v^f - \frac{v^f}{k^j} \cdot k$, where v^f is the free flow speed (i.e. the maximum driving speed), and k^j is the jam density (i.e. the density which causes traffic jam). In this model, we set $v^f = 100 \text{ km/h}$ and $k^j = 100 \text{ vehicles/km}$, which are reasonable values in realistic highway environments. We simulate a four-lane highway, and the arrival of vehicles in each lane follows the Poisson process with the mean arrival rates (λ) of 1, 1/2, 1/3 and 1/4 (vehicles/s), respectively. The communication characteristic is simulated based on DSRC [24]. In particular, the data transmission rate is set to 6 Mbps and the radius of the RSU coverage is set to 300 m. It has been shown that the above settings can provide reliable data dissemination via DSRC [25]. The data size is set to 1 Kb, which is sufficient for normal location-dependent information. The data update period is uniformly distributed from 200 s to 300 s. The request size is 3, and the request deadline is bounded by the dwell time of vehicles in the RSU's coverage.

Due to the factors such as the market penetration of on-board units mounted on vehicles and the options of drivers, it is not necessarily that all the vehicles will submit requests. Figure 14 shows the algorithm performance with different percentages of vehicles submitting requests. As shown, PSU achieves nearly 100% service ratio when half of the vehicles submit requests. Moreover, even in a saturated scenario where all the vehicles submit requests, over 80% of requests can be satisfied by PSU. This result demonstrates the scalability of PSU in realistic vehicular environments.

VI. RELATED WORK

In vehicular networks, current studies on data dissemination largely focused on the design of protocols to improve communication quality and reliability. Maeshima et al. [26]

designed a MAC protocol in supporting emergency message delivery. Whenever an emergency notification occurred, the transmission of general information would be suspended to ensure timely delivery of emergency messages. Jhang and Liao [27] proposed a proxy-based communication protocol for data uploading from vehicles to the RSU. It relieves the uplink channel contention and improves the system throughput by electing proxy vehicles. Non-proxy vehicles which attempt to communicate with the RSU must forward their data items to a proxy vehicle. Mak et al. [28] proposed a coordinated MAC mode. It improves the performance for both safety and non-safety applications by adopting a multi-channel coordination mechanism, which minimizes the collision between vehicle-to-vehicle and roadside-to-vehicle communications. In order to support high reliability and low delay for data dissemination in vehicular networks, Farnoud and Valaee [29] proposed a topology-transparent broadcast protocol, which enhances the system performance on providing safety-critical services. These studies focused on vehicular communication issues at MAC and PHY layers, while none of them considered the time-constraint of services and the freshness of data items at the application level.

In the real-time database community, extensive research has been devoted to maintaining and processing time-variant information, where the values of data items are valid only for a certain time interval [30]. Many approaches have been proposed for managing temporal data items to track the dynamics of the real world ([31], [32], [33], [34]). These studies focused on striking a balance between the quality of service (QoS) and the quality of data (QoD) to improve overall system performance. However, none of them are designed to address the scheduling problem of serving real-time requests with temporal data items.

Scheduling algorithms have been extensively examined in the network community. In non-real-time systems, there are a number of classical scheduling algorithms. MRF (Most Requested First) [35] broadcasts the data item which has the largest number of pending requests to account for the productivity of broadcast. RxW (Number of pending Requests Multiply Waiting time) [5] calculates the number of pending requests for a data item multiplied by the amount of time that the oldest outstanding request for that data item has been waiting, and it schedules the request with the maximum RxW value. In real-time systems, EDF (Earliest Deadline First) [12] is one of the foremost classical scheduling algorithms, which broadcasts the data item with the shortest remaining lifetime to cater for the urgency of requests. SIN (Slack time Inverse Number of pending requests) [6] is another representative real-time scheduling algorithm, which combines the advantages of both EDF and MRF. It has been demonstrated that SIN outperforms existing algorithms in serving time-critical requests in on-demand broadcast environments. Our recent study has investigated the temporal data dissemination problem in vehicular networks [36]. This paper significantly extends our preliminary study on both theoretical analysis and simulation results with respected to the investigated problem.

VII. CONCLUSION AND FUTURE WORK

Efficient data dissemination is critical to enable innovative applications in VCPS. In this paper, we introduce the roadside-to-vehicle communication system and investigate the unique characteristics and challenges of data dissemination in such an environment. According to an intensive analysis of the requirement on serving real-time requests with temporal data items, we present the snapshot consistency requirement and formulate the TDD problem. We prove that TDD is NP-hard by constructing a polynomial-time reduction from the Clique problem to TDD. Based on the analysis of the time bound of request services, we propose an on-line scheduling algorithm PSU, which aims at exploiting the broadcast effect, improving the bandwidth efficiency and enhancing the request service. We build the simulation model and design a number of metrics for performance evaluation. A comprehensive simulation study is presented by comparing the performance of PSU with alternative real-time scheduling algorithms including EDF and SIN. The simulation results validate that PSU is effective in providing real-time data services under different traffic scenarios and application requirements.

As an early stage of exploring the temporal data dissemination in VCPS, this paper concentrates on a proof of concept of providing real-time data services via roadside-to-vehicle communication. In future, more realistic approaches (e.g. considering communication influences at MAC and Physical layers such as packet drops and interferences) are expected to be developed for real-world applications. Besides, it is desirable to further enhance the system performance by incorporating inter-vehicle communication for data sharing among neighboring vehicles.

REFERENCES

- [1] A. Miloslavov and M. Veeraraghavan, "Sensor data fusion algorithms for vehicular cyber-physical systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 9, pp. 1762–1774, 2012.
- [2] S. K. Gehrig and F. J. Stein, "Collision avoidance for vehicle-following systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 8, no. 2, pp. 233–244, 2007.
- [3] K. Liu, E. Chan, V. Lee, K. Kapitanova, and S. H. Son, "Design and evaluation of token-based reservation for a roadway system," *Transportation Research Part C: Emerging Technologies*, vol. 26, pp. 184–202, 2013.
- [4] J. Lee and B. Park, "Development and evaluation of a cooperative vehicle intersection control algorithm under the connected vehicles environment," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 1, pp. 81–90, 2012.
- [5] D. Aksoy and M. Franklin, "R×w: a scheduling approach for large-scale on-demand data broadcast," *IEEE/ACM Transactions on Networking*, vol. 7, no. 6, pp. 846–860, 1999.
- [6] J. Xu, X. Tang, and W.-C. Lee, "Time-critical on-demand data broadcast: Algorithms, analysis, and performance evaluation," *IEEE Transactions on Parallel and Distributed Systems*, vol. 17, no. 1, pp. 3–14, 2006.
- [7] K. Liu and V. Lee, "Simulation studies on scheduling requests for multiple data items in on-demand broadcast environments," *Performance Evaluation*, vol. 66, no. 7, pp. 368–379, 2009.
- [8] J. Zhang, Q. Zhang, and W. Jia, "Vc-mac: A cooperative mac protocol in vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 58, no. 3, pp. 1561–1571, 2009.
- [9] X. Ma, J. Zhang, X. Yin, and K. S. Trivedi, "Design and analysis of a robust broadcast scheme for vanet safety-related services," *IEEE Transactions on Vehicular Technology*, vol. 61, no. 1, pp. 46–61, 2012.
- [10] Q. Wang, P. Fan, and K. B. Letaief, "On the joint v2i and v2v scheduling for cooperative vanets with network coding," *IEEE Transactions on Vehicular Technology*, vol. 61, no. 1, pp. 62–73, 2012.
- [11] C.-J. Chang, R.-G. Cheng, H.-T. Shih, and Y.-S. Chen, "Maximum freedom last scheduling algorithm for downlinks of dsrc networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 8, no. 2, pp. 223–232, 2007.
- [12] P. Xuan, S. Sen, O. Gonzalez, J. Fernandez, and K. Ramamritham, "Broadcast on demand: Efficient and timely dissemination of data in mobile environments," in *Proceedings of the 3rd IEEE Real-Time Technology and Applications Symposium (RTAS'97)*, 1997, pp. 38–48.
- [13] Y. Zhang, J. Zhao, and G. Cao, "On scheduling vehicle-roadside data access," in *Proceedings of the fourth ACM international workshop on Vehicular ad hoc networks (VANET '07)*, 2007, pp. 9–18.
- [14] F. J. Ros, P. M. Ruiz, and I. Stojmenovic, "Acknowledgment-based broadcast protocol for reliable and efficient data dissemination in vehicular ad hoc networks," *IEEE Transactions on Mobile Computing*, vol. 11, no. 1, pp. 33–46, 2012.
- [15] K. Liu and V. Lee, "Adaptive data dissemination for time-constrained messages in dynamic vehicular networks," *Transportation research part C: emerging technologies*, vol. 21, no. 1, pp. 214–229, 2012.
- [16] Wikipedia, "Clique problem." [Online]: http://en.wikipedia.org/wiki/Clique_problem, 2013.
- [17] K. Liu and V. Lee, "On-demand broadcast for multiple-item requests in a multiple-channel environment," *Information Sciences*, vol. 180, no. 22, pp. 4336–4352, 2010.
- [18] V. Lee and K. Liu, "Scheduling time-critical requests for multiple data objects in on-demand broadcast," *Concurrency and Computation: Practice and Experience*, vol. 22, no. 15, pp. 2124–2143, 2010.
- [19] J.-Y. Ng, V. Chung-Sing, and C. Y. Hui, "Client-side caching strategies and on-demand broadcast algorithms for real-time information dispatch systems," *IEEE Transactions on Broadcasting*, vol. 54, no. 1, pp. 24–35, 2008.
- [20] H. Schwetman, "Csim19: a powerful tool for building system models," in *Proceedings of the 33rd Conference on Winter Simulation (WSC'01)*, 2001, pp. 250–255.
- [21] G. K. Zipf, "Human behavior and the principle of least effort." 1949.
- [22] C. F. Daganzo, *Fundamentals of transportation and traffic operations*, 1997.
- [23] P. Edara and D. Teodorović, "Model of an advance-booking system for highway trips," *Transportation Research Part C: Emerging Technologies*, vol. 16, no. 1, pp. 36–53, 2008.
- [24] FCC, "FCC report and Order 06-110," Amendment of the commission's rules regarding dedicated short-range communication services in the 5.850-5.925GHz band, 20-07-2006.
- [25] F. Bai and H. Krishnan, "Reliability analysis of dsrc wireless communication for vehicle safety applications," in *Proceedings of the 9th International IEEE Conference on Intelligent Transportation Systems (ITSC'06)*, 2006, pp. 355–362.
- [26] O. Maeshima, S. Cai, T. Honda, and H. Urayama, "A roadside-to-vehicle communication system for vehicle safety using dual frequency channels," in *Proceedings of the 10th International IEEE Conference on Intelligent Transportation Systems (ITSC'07)*, 2007, pp. 349–354.
- [27] M.-F. Jhang and W. Liao, "Cooperative and opportunistic channel access for vehicle to roadside (v2r) communications," *Mobile Networks and Applications*, vol. 15, no. 1, pp. 13–19, 2010.
- [28] T. K. Mak, K. P. Laberteaux, and R. Sengupta, "A multi-channel vanet providing concurrent safety and commercial services," in *Proceedings of the 2nd ACM International Workshop on Vehicular Ad Hoc Networks (VANET '05)*, 2005, pp. 1–9.
- [29] F. Farnoud and S. Valaee, "Reliable broadcast of safety messages in vehicular ad hoc networks," in *Proceedings of the 28th IEEE International Conference on Computer Communications (INFOCOM'09)*, 2009, pp. 226–234.
- [30] J. A. Stankovic, S. H. Son, and J. Hansson, "Misconceptions about real-time databases," *Computer*, vol. 32, no. 6, pp. 29–36, 1999.
- [31] M. Amirijoo, J. Hansson, and S. H. Son, "Specification and management of qos in real-time databases supporting imprecise computations," *IEEE Transactions on Computers*, vol. 55, no. 3, pp. 304–319, 2006.
- [32] W. Kang, K. Kapitanova, and S. H. Son, "Rdds: A real-time data distribution service for cyber-physical systems," *IEEE Transactions on Industrial Informatics*, vol. 8, no. 2, pp. 393–405, 2012.
- [33] M. Xiong, S. Han, K.-Y. Lam, and D. Chen, "Deferrable scheduling for maintaining real-time data freshness: Algorithms, analysis, and results," *IEEE Transactions on Computers*, vol. 57, no. 7, pp. 952–964, 2008.
- [34] J. Wang, S. Han, K.-Y. Lam, and A. K. Mok, "Maintaining data temporal consistency in distributed real-time systems," *Real-Time Systems*, vol. 48, no. 4, pp. 387–429, 2012.
- [35] J. W. Wong, "Broadcast delivery," *Proceedings of the IEEE*, vol. 76, no. 12, pp. 1566–1577, 1988.

- [36] K. Liu, V. Lee, J. Ng, and S. Son, "Scheduling temporal data for real-time requests in roadside-to-vehicle communication," in *Proceedings of the 19th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'13)*, 2013, pp. 297–305.



Kai Liu (S'07-M'12) received his Ph.D. degree in Computer Science from City University of Hong Kong in 2011. He is currently an Assistant Professor in the College of Computer Science, Chongqing University, Chongqing, China. He has been a visiting scholar in Computer Science Department at the University of Virginia, USA, from Dec. 2010 to May 2011. He has been a Postdoctoral Fellow at Singapore Nanyang Technological University, City University of Hong Kong and Hong Kong Baptist University from 2011 to 2014. His research interests

include real-time scheduling, mobile computing, intelligent transportation systems and vehicular cyber-physical systems.



Victor C.S. Lee (M'97) received the PhD degree in computer science from the City University of Hong Kong in 1997. He is currently an Assistant Professor in the Department of Computer Science, City University of Hong Kong. His research interests include data dissemination in vehicular networks, real-time databases and performance evaluation. He is a member of the ACM, IEEE and IEEE Computer Society. He has been the chairman of the IEEE, Hong Kong Section, Computer Chapter in 2006-2007.



Joseph Kee-Yin NG (M'91-SM'00) received his B.Sc., M.Sc., and Ph.D. degree in Computer Science all from the University of Illinois at Urbana-Champaign. Prof. Ng joined Hong Kong Baptist University in 1993, and is a Professor in the Department of Computer Science. He is the programme coordinator of the Computer Science degree programme and introduced Health Information Technology and Health Informatics into the undergraduate as well as the graduate programmes in HKBU. Prof. Ng's current research interests include Real-Time &

Embedded Systems, Multimedia Communications, and Ubiquitous/Pervasive Computing. Prof. Ng has obtained 2 Patents and published over 135 technical papers in journals, and conferences. Prof. Ng also served as Steering Chairs, Program Chairs, and General Chairs for numerous International Conferences as well as Associate Editors and members of the editorial board of International Journals.

Prof. Ng had also served as the Region 10 Coordinator for the Chapter Activities Board of the IEEE Computer Society, and was the Coordinator of the IEEE Computer Society Distinguished Visitors Program (Asia/Pacific). He is a senior member of the IEEE and has been a member of the IEEE Computer Society since 1991. Prof. Ng has been an Exco-member, General Secretary, Vice-Chair, Chair and now a past Chair and Exco-member for the IEEE, Hong Kong Section, Computer Chapter. Prof. Ng received numerous Awards and Certificate of Appreciation from IEEE, IEEE Region 10, IEEE Computer Society, and from IEEE Hong Kong Section, for his Leadership and Services to the ICT Industry. He is also a member of the IEEE Communication Society, ACM, Hong Kong Computer Society, and the Founding Member and Exco-Member & Treasurer for the Internet Society (ISOC)-Hong Kong Chapter. Prof. Ng is also a Director of the Hong Kong Internet Registration Corporation Limited (HKIRC).



Jun Chen received her Ph.D degree in Computer Science from the Wuhan University of China in 2008. She was a senior research associate in the Department of Computer Science, City University of Hong Kong from 2008 to 2009. She was a visiting fellow in the Department of Information System, City University of Hong Kong in 2012. She is currently an Associate Professor in the School of Information Management of the Wuhan University in China. Her research interests include mobile computing and data management in wireless networks.



Sang Hyuk Son (M'85-SM'98-F'13) is IEEE Fellow and Department Chair of Information and Communication Engineering at DGIST. He has been a Professor of Computer Science Department at the University of Virginia, and WCU Chair Professor at Sogang University. He received the B.S. degree in electronics engineering from Seoul National University, M.S. degree from KAIST, and the Ph.D. in computer science from University of Maryland, College Park. He has been a Visiting Professor at KAIST, City University of Hong Kong, Ecole

Centrale de Lille in France, and Linkoping University and University of Skovde in Sweden.

Prof. Son has served as the chair of the IEEE Technical Committee on Real-Time Systems during 2007-2008. He is serving as an Associate Editor for Real-Time Systems Journal and Journal on Self Computing, and has served on the editorial board of IEEE Transactions on Computers and IEEE Transactions on Parallel and Distributed Systems. He is also serving as a member of the steering committee for RTCSA, Cyber physical Systems Week, and SEUS. He received the Outstanding Contribution Award from ACM/IEEE Cyber Physical Systems Week in 2012. His research interests include cyber physical systems, real-time and embedded systems, database and data services, and wireless sensor networks. He has written or co-authored over 290 papers and edited/authored four books in these areas. His research has been funded by National Science Foundation, DARPA, Office of Naval Research, Department of Energy, National Security Agency, and IBM.