Efficient Data Dissemination in Cooperative Multi-RSU Vehicular Ad Hoc Networks (VANETs)

Nawaz Ali, G.G.Md.; Chong, Peter Han Joo; Samantha, Syeda Khairunnesa; Chan, Edward

2016

Nawaz Ali, G. G. M., Chong, P. H. J., Samantha, S. K., & Chan, E. (2016). Efficient data dissemination in cooperative multi-RSU Vehicular Ad Hoc Networks (VANETs). *Journal of Systems and Software*, 117, 508-527.

<https://hdl.handle.net/10356/80588>

<https://doi.org/10.1016/j.jss.2016.04.005>

© 2016 Elsevier. This is the author created version of a work that has been peer reviewed and accepted for publication by *Journal of Systems and Software*, Elsevier. It incorporates referee's comments but changes resulting from the publishing process, such as copyediting, structural formatting, may not be reflected in this document. The published version is available at: [<http://dx.doi.org/10.1016/j.jss.2016.04.005>].

Downloaded on 26 Apr 2023 19:40:06 SGT

Efficient Data Dissemination in Cooperative Multi-RSU Vehicular Ad Hoc Networks (VANETs)

G. G. Md. Nawaz Ali^{a,*}, Peter Han Joo Chong^a, Syeda Khairunnesa Samantha^b, Edward Chan^c

^a*School of Electrical & Electronic Engineering, Nanyang Technological University, Singapore*

^b*Department of Computer Science, Iowa State University, Ames, Iowa, USA*

^c*Department of Computer Science, City University of Hong Kong, Kowloon, Hong Kong*

Abstract

Many safety and non-safety related applications have been envisioned in VANETs. However, efficient data dissemination considering the mobility of vehicle is must for the success of these applications. Although the Road Side Unit (RSU) is a stationary unit, both RSU and vehicle have limited transmission range that restricts to shorter connection time. This endures a higher request drop rate specially at the overloaded RSUs. A cooperative load balancing (CLB) among the RSUs to use their residual bandwidth can be an effective solution to reduce the request drop rate. In this paper, we investigate that considering the remaining delay tolerance of submitted requests and the knowledge of fixed road layout, the performance of the cooperative load balancing system can be further improved significantly. We show that this performance gain comes from serving the requests based on the urgency and the efficient load balancing among the junction-RSUs and edge-RSUs. Based on the observations, we propose an Enhanced CLB (ECLB) approach in this paper. To demonstrate the efficiency of the ECLB approach a number of well-known scheduling algorithms are integrated and an extensive simulation experiments are conducted in the vehicular communication environment that supports the superiority of ECLB over the existing approaches.

Keywords: Vehicular Ad Hoc Networks (VANETs); Road Side Units (RSUs); cooperative load balancing; scheduling; on-demand broadcasting.

1. Introduction

Vehicular Ad Hoc Networks (VANETs) is a popular and emerging technology in the Intelligent Transportation System (ITS). A number of applications have been envisioned in ITS such as road safety, driving assistance, internet access from on-board vehicles etc. [1, 2]. The major goals of an ITS system are to reduce fatalities and financial losses due to road accidents, along with providing driving comfort [3, 4]. Due to the rapid growth of the emerging applications including safety critical information, other information also attracts passengers onboard. These are known as non-safety critical applications such as value added advertisement, infotainment, audio/video download etc. [5]. Nevertheless, efficient data dissemination considering vehicle

*Corresponding author

Email addresses: nawaz.ali@ntu.edu.sg (G. G. Md. Nawaz Ali), ehjchong@ntu.edu.sg (Peter Han Joo Chong), sammy@iastate.edu (Syeda Khairunnesa Samantha), csedchan@cityu.edu.hk (Edward Chan)

Preprint submitted to Elsevier

May 24, 2016

mobility, transmission range constrain and strict timing is necessary for making those VANETs applications successful.

Generally, there are two communication models in ITS [6], vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) (also called vehicle-to-RSU) communications. Our focus in this work is the V2I communication model. Note that vehicles may also exchange information with other vehicles outside the service region of the RSU, which are the issues addressed in the V2V communication model. Dedicated Short-Range Communications (DSRC) is exclusively developed and widely accepted for emerging ITS applications in VANETs [7, 8]. DSRC is a technology, which supports both V2V and V2I communications. In general, DSRC refers to a family of standards of Wireless Access in Vehicular Environments (WAVE). The standards include IEEE 802.11p, IEEE 1609.1, 1609.2, 1609.3, 1609.4 and SAE J2735 message set dictionary [8].

In DSRC, V2V communication is advised mainly for the collision avoidance applications such as forward collision avoidance, electronic brake lights, blind spot warning, intersection movement assist etc. On the other hand, V2I communication can be used for many other applications such as to assist navigation, disseminate traffic update, finding a traffic lot, make electronic payment, infotainment, to name but a few. In contrast of collision avoidance/safety critical applications, other applications in V2I may tolerate some servicing delay. For instance traffic update of a particular road, as soon as the vehicle get this update before reaching the intersection point, the driver can make it useful for routing decision. Hence it is an interesting but challenging problem to serve maximum number of vehicles with their requested data items within the prescribed delay tolerance. The objective can be achieved by the efficient V2I communication and the maximum usage of the available broadcast bandwidth of RSU by cooperation. Based on the tolerable delay of the applications, the cooperative usage of the residual bandwidth of multiple RSUs and data sharing among the vehicles during V2I communication can exploit the bandwidth usage and improve the data dissemination performance. It is reasonable to assume that a vehicle with DSRC technology is equipped with communication device (OBU) and Global Positioning System (GPS) navigator [9, 10]. With the OBU, a vehicle periodically broadcasts basic safety message (BSM), which includes the location, velocity and driving direction of the vehicle (defined in SAE J2735). Upon receiving the BSMs from vehicles an RSU knows the next heading direction of the passing vehicles. From the beacon message of RSU, a vehicle can know the transmission range of an RSU, and hence with the knowledge of driving speed the vehicle can estimate the dwelling duration inside the RSU range. Using the digital map, OBU and GPS navigator a vehicle can find the initial route, or reroute in the middle towards the destination. As road layout is fixed and RSUs are stationary, if an RSU knows the route of a corresponding vehicle, the RSU also knows which other RSUs the vehicle is going to pass towards the destination. Using this information the RSU can perform the cooperative request serving with the residual bandwidth of the other RSUs in that route. Note that although a GPS navigator/tablet device is not necessary for cooperative load balancing procedure, through such device a driver can be warned or assisted audio-visually for safety and improved driving. Hence, the driver can plan his route more wisely. The major contributions of this paper are stated below.

- We propose an Enhanced Cooperative Load Balancing (ECLB) system in V2I communication. In ECLB, the required data items corresponding to non-safety critical applications are scheduled to serve within the prescribed delay tolerance value such as disseminating traffic update, finding a traffic lot, infotainment etc. In addition, ECLB considers the vehicle dynamics and road layout. Note that safety-critical applications that require emergency service such as collision avoidance, are discussed in V2V communication, is out of the

scope and space of this paper. A preliminary work is shown in [11].

- ECLB prioritizes requests based on the remaining delay tolerance. This prompts the system to serve more urgent requests (requests with the lower remaining delay tolerance) at the current RSU, and to serve the requests with loose delay tolerance cooperatively with the residual bandwidth of the other RSUs.
- A vehicle may leave the RSU range before receiving the required data item. This might happen due to RSU being busy serving other urgent requests. The proposed ECLB approach transfers the unserved request to an RSU situated in the route the vehicle is heading and the RSU will have enough residual bandwidth to serve the transferred request when the corresponding vehicle arrives inside the servicing range.
- In the existing CLB approach [12], the current RSU even may have to transfer the overload to the junction-RSU (note that a junction-RSU refers an RSU located at the intersection of roads, on the other hand, an edge-RSU refers an RSU located at the non-intersection place of roads), which is usually a busy RSU, may result in a higher overall request drop rate. In the ECLB approach, we can overcome this limitation by extending the load balancing horizon, namely by increasing the number of participating RSUs to share the overload.
- We show that the ECLB approach can readjust the overloaded workload if a vehicle changes its route in the middle of its routing towards the destination. In addition, we ensure that the performance of ECLB is better or at least as good as the performance of CLB under any circumstances.
- The ECLB approach reduces the deadline conflicted requests (the requests which are going to miss their deadline before completing the transmission of the next broadcast data item) at the transferred RSU by carefully considering the load balancing, which is the key to satisfy more requests and improve the overall system performance.
- As VANETs needs to serve vehicles with different interests, we apply on-demand broadcasting at the RSUs for data dissemination [13, 14]. In the on-demand broadcasting the RSU server has the way to know the clients' interests [15]. According to the underlying scheduling algorithm the server takes the decision which item to broadcast next. Hence, scheduling algorithm plays a vital role for the efficient data dissemination [7, 13, 16]. We integrate the proposed ECLB approach with a number of on-demand scheduling algorithms and evaluate their performances against the same set of on-demand scheduling algorithms integrated with the CLB and the standalone approaches. Note that a standalone approach is an approach where RSUs do not communicate for load balancing with each other, namely there is no cooperation involved to use the residual bandwidth of other RSUs.
- We build a simulation model based on the vehicular communication characteristics and evaluate the performance of ELCB, CLB and standalone approaches under different circumstances. The simulation results demonstrate the superiority of proposed ECLB over other approaches under a variety of circumstances.

The rest of the paper is organized as follows. Section 2 presents the related work. Section 3 describes the motivation, system model, preliminaries and definitions. The proposed ECLB model is discussed in Section 4. Section 5 demonstrates the performance evaluation, and finally Section 6 concludes the paper.

2. Related Work

The two main components of the proposed work in this paper are the data dissemination challenges in VANETs, and efficient on-demand data scheduling for maximizing the RSU channel bandwidth. In the following subsections, first we review the existing approaches of data dissemination in VANETs, specifically, data dissemination approaches through V2I and V2V communication models and then we review the existing on-demand scheduling approaches that might be integrated with our proposed ECLB approach.

2.1. V2I communication

Substantial efforts have been put forth on V2I communication for efficient data dissemination. Some of the works consider single RSU environment [14, 17]. Zhang et al. [14] propose to install stationary RSUs along roadsides, which can act as buffer points and communicate with moving vehicles. They model a standalone RSU-based VANETs environment which maintains both upload and download queues and try to get service balanced among them. Liu and Lee [17] analyze the dynamic traffic characteristics in standalone RSU-based VANETs. They propose to use different channels to disseminate different types of data and apply push and pull data dissemination techniques based on the volume of requests at the RSU server. Some of the works consider multi-RSU environment [18, 19, 20, 21]. Lochert et al. [18] study the comparative effectiveness of VANETs communication in terms of with and without RSU-based communication and find inter-connected multiple RSUs can significantly improve communication performance. Shahverdy et al. [19] propose a multi-RSU architecture where RSUs are deployed at the intersections of the road with high speed backbone connection. They argued if a downloading file size is large, it can be served as multiple segments from multiple RSUs. Wang et al. [20] study the data dissemination problem in a joint V2V and V2I communication system. They propose to disseminate information using V2I to the vehicles present inside the RSU range and then these vehicles will relay the received information towards other vehicles outside the RSU's transmission range using V2V. To reduce the handoff latency in VANETs, Wu et al. [21] propose Access Point Controller (APC) based scheme *QualityScan* that periodically collects the status of multiple RSUs (APs) to predict the network traffic for the next moment. However, being different from the existing works, our work considers load balancing among multiple RSUs with cooperation to reduce the overall system workload.

2.2. V2V communication

TrafficView proposed by Nadeem et al. [22, 23] uses the data push communication model for exchanging information among vehicles. The system is developed for traffic data dissemination and visualization to assist a driver in route planning and driving in the adversarial weather condition. MDDV proposed by Wu et al. [24] is designed to exploit the vehicle mobility for data dissemination and it combines the idea of opportunistic forwarding, trajectory based forwarding and geographical forwarding. Here vehicles perform local operations based on their own knowledge while they collectively achieve a global behavior. Chen et al. [25] also propose a similar kind of message relayed technique for data forwarding. For delay tolerant and some localized applications messages can be relayed and stored temporarily at moving vehicles while waiting for the opportunities to be forwarded further. Yu and Heijenk [26] use the geocast approach to disseminate the warning and safety messages to a group of relevant vehicles whereas opposite vehicles are used just for relaying to reduce the broadcast overhead and fragmentation problems.

Moreover, it introduces the dynamic wait time for the relying vehicles whilst keeping the warning message in the desired affected area. To disseminate data at the road intersections, Zhao et al. [27] present the Data Pouring with the Intersection Buffering (DP-IB) scheme. Here DP is used to broadcast data periodically to the vehicles on the road and the poured data are buffered by the running vehicles and rebroadcast at the intersection regions. In VADD, Zhao and Cao [28] to minimize the data delivery delay, propose several vehicle-assisted data delivery techniques which can take the advantages of fixed road layout and predictable traffic patterns. All the above works consider push based periodic message broadcasting, and ignore the dynamic access patterns of vehicles. In contrast to these works, our work considers on-demand delay tolerant data broadcasting which can reflect the dynamic needs of vehicles and maximize the usage of system bandwidth as well.

2.3. On-demand scheduling

An efficient scheduling algorithm can play a significant role for the effective and time efficient data dissemination by utilizing the channel bandwidth [16, 29]. On-demand broadcast is more scalable for dynamic and large-scale data dissemination [13, 30]. On-demand scheduling can be applied both in non-real-time [13, 31, 32] and real-time systems [16, 33], however different type system has different system objective to achieve. In non-real-time system, *responsiveness*, which is defined as the required time period after submitting the request to get the corresponding response, is the main performance measure for the overall system performance. So the main system objective is to achieve the lower request-response time. Wong and Ammar [31] investigate the behavior of response time in the videotex system. Through a queuing model, the videotex system can satisfy the outstanding clients for the same data item by broadcasting. Queuing model is designed according to FCFS (First Come First Served) discipline. MRF (Most Request First) and LWF (Longest Wait First) are proposed by Wong [32]. The data item which has the most number of pending requests is broadcast first in MRF, on the other hand, LWF broadcasts the data item for which the aggregated waiting time of the pending requests is the maximum. To reduce the exhaustive searching space in LWF, Aksoy and Franklin [13] propose $R \times W$ (Number of pending requests multiple Waiting time) for large-scale on-demand broadcast, where R means the popularity of the requested item and W means the waiting time of the oldest request for that item. The data item with the maximum $R \times W$ value will be broadcast first. In real-time scheduling, request has deadline constraint and a request only be satisfied if it is served within the prescribed deadline. So the main objective of this type of system is to satisfy as many requests as possible within their deadlines, namely to minimize the *deadline miss ratio* which is the ratio of the number of deadline missed requests to the total requests. Xuan et al. [33] study the different scheduling policies on the proposed on-demand framework BoD (Broadcast on Demand) and found EDF performs the best. By integrating the properties of MRF and EDF, for time critical on-demand broadcast Xu et al. [16] propose an algorithm called SIN- α , where $SIN-\alpha = \frac{slack}{num^\alpha}$, *slack* is the duration from the current time to the deadline of the most urgent pending request and *num* is the pending requests number of the item. α is a tuning parameter used to shape the relative weight of productivity over urgency, SIN- α serves the item having minimum SIN- α value. The simulation results show that SIN-1 (also called SIN) outperforms EDF, MRF and $R \times W$ in terms of deadline miss ratio. In our proposed ECLB approach, an RSU uses an on-demand scheduling algorithm to determine the next data item to broadcast. We have incorporated a number of on-demand scheduling algorithms with ECLB to measure and analyze their performances in the vehicular network environment.

Figure 1: System Architecture.

3. Motivation, Preliminaries and System Model

3.1. Motivation

If a generated request is not served in an RSU by the dwelling duration of the corresponding vehicle, the request may need to be served by other RSU in the route of the vehicle within the prescribed delay tolerance. However, serving a request by other RSU requires the cooperative load balancing among the RSUs so that the transferred overloaded request(s) of the transferee RSU does not overwhelm the serving capacity of the transferred RSU. The two important characteristics of VANETs are (1) the mobility of vehicle [14], (2) the fixed road layout [27]. A vehicle can change its driving direction at any intersection point. However, as the road layout does not change and RSUs are the stationary units, the number of RSUs and their positions in a particular road that a vehicle passes can also be known. The above two observations form the basis of cooperative load balancing. For the cooperative load balancing among the RSUs, the coordination is must among the RSUs that a vehicle passes [21]. However, as a vehicle may change its driving direction at an intersection point, the previous assumed RSUs that a vehicle could pass might no longer be valid for load balancing. Hence the dynamic driving direction change is a disadvantage for the load balancing. On the contrary, as the positions of RSUs of a road do not change, it is certain that a vehicle on that road passes those RSUs before approaching the intersection ahead. Hence, coordination is possible among those RSUs for load balancing. So, fixed road layout is an advantage for load balancing. Hence, the efficient load balancing system should be carefully designed that can cooperate both the characteristics of VANETs. Note that a vehicle can inform the RSU about the route it is following in addition to submitting request for the required data item from the RSU. The detailed system architecture is stated below.

3.2. System architecture

In accordance with the IEEE 1609.4, we consider that one control channel (CCH) and two service channels (SCH) are operating [34]. Through the CCH, an RSU periodically broadcasts control messages, safety messages, service advertisement etc. One of SCH is used for V2I communication and the other is for V2V communication. We consider due to the deployment and economic concerns only single radio OBUs are adopted in VANETs [7]. Hence a vehicle at a time only can operate either V2I or V2V communication mode. Moreover each RSU can support two channels, one control channel (CCH) and one service channel (SCH) by periodic channel switching [35]. Through the CCH, vehicles can upload their requests for non-safety critical applications piggyback with the BSM. Later the uploaded requests are inserted into the RSU's service queue for scheduling. Along with the safety critical applications the RSU periodically broadcasts the index information of the stored data items from the database through the CCH. The CCH also broadcasts the index information of the next broadcast items so that the vehicle which needs the corresponding data item can tune to SCH. Before making the decision on which request to serve, the RSU invokes the underlying scheduling algorithm to select the corresponding data item for broadcasting through the SCH. In addition, according to the proposal in [18], we assume that multiple RSUs are interconnected through a backbone network which enable them to exchange their workload information and the unserved overloaded requests internally. A transferred request will be considered for scheduling only when the corresponding vehicle arrives at the transferred RSU service range. The overall system architecture is shown in Fig. 1.

3.3. Notations, assumptions and definitions

In this section we describe the notations and the assumptions used in our system model. Table 1 lists the primary notations. Assume that each RSU contains m data items, where the set of data items $D = \{d_1, d_2, \dots, d_m\}$. $S(d_j)$ denotes the size of data item d_j , and B denotes the service channel bandwidth, hence the transmission time of data item d_i is $T_i^{trans} = \frac{S(d_i)}{B}$.

Request. Assume when a vehicle Vh_i submits a request R_i to an RSU, it submits the following tuples:

$$R_i = (d_i, T_i^{DTolerance}, HD(Vh_i), route_i).$$

Here,

d_i : The requested data item;

$T_i^{DTolerance}$: The maximum delay tolerance of a request. Beyond this time request R_i will neither be served nor transferred to another RSU. When a request R_i is generated, it ensures that $T_i^{DTolerance}$ must be higher or equal to the dwelling duration of the vehicle in the current RSU. If $T_i^{DTolerance}$ equals the dwelling duration then it indicates that the request must be served before the vehicle leaves the RSU range. If $T_i^{DTolerance}$ is higher than the dwelling duration, it means that the request tolerates some delay and this request might be served in other RSU in its route, if not served in the current RSU. As time progresses from the request generation, $T_i^{DTolerance}$ decreases. After generation of R_i , at time t , the remaining delay tolerance denoted by $\bar{T}_i^{DTolerance}(t)$ is $\bar{T}_i^{DTolerance}(t) = \{T_i^{DTolerance} - t\}$.

$HD(Vh_i)$: The next heading direction of Vh_i .

$route_i$: The driving route of Vh_i .

Table 1: Summary of notations.

NOTATION	DESCRIPTION
R_i	A request
d_i	The requested data item of R_i
$S(d)$	Size of d_i
R^t	Set of requests at time t
Vh_i	A vehicle
RSU_i	An RSU
R	Transmission range of an RSU
V	Average speed of a vehicle
T_i^{trans}	Transmission time of R_i
T_i^r	Generation time of R_i
$T_i^{deadline}$	Deadline of R_i
$T^{Maxdwell}$	Maximum dwelling duration of a vehicle inside an RSU
T_i^{dwell}	Dwelling duration of Vh_i inside an RSU
$T_i^{DTolerance}$	Delay tolerance value of R_i
$\bar{T}_i^{DTolerance}(t)$	Remaining delay tolerance of R_i at time t
$HD(Vh_i)$	Heading direction of Vh_i
d_{avg}	Average blind zone distance
T_{hop}^{travel}	Per hop travel time
$SA(R_i, R^t)$	Sharable requests set of R_i at t

Figure 2: Different Timings in an RSU.

Let T_i^{in} denotes the time Vh_i enters into the service range of the RSU. When a vehicle starts receiving beacon messages from an RSU, that is the entering point of the vehicle in the corresponding RSU range. This entering point is also the arrival time of the vehicle in that RSU. T_i^{out} denotes the time Vh_i leaves the communication range of the RSU. Note that a vehicle equipped with OBU can estimate location information. In addition, through the beacon messages the vehicle will also get the RSU transmission range of the RSU. Together with the knowledge of driving speed, a vehicle can calculate the exiting time from the RSU range. T_i^r denotes the time the request is generated. Fig. 2 shows these timings in an RSU.

Blind Zone Distance. The inter-RSU distance a vehicle travels outside the RSU transmission range is called the *blind zone distance* [20]. Assume d_{hop} denotes the per hop blind zone distance and V denotes the average speed of a vehicle. Hence, per hop blind zone travel time is, $T_{hop}^{travel} = \frac{d_{hop}}{V}$.

Deadline of a request.

- **Deadline of a generated request:** The deadline of a generated request in an RSU depends on the request generation time and the dwelling duration of the vehicle inside the RSU range. Assume that the radius of the transmission range of an RSU is R meters and the average speed of a vehicle within the transmission range of an RSU is V m/s. The maximum dwelling duration of a vehicle at an RSU is $T^{Maxdwell} = \frac{2R}{V}$. If R_i is generated by Vh_i inside RSU_i at time T_i^r , the dwelling duration of Vh_i is $T_i^{dwell}(RSU_i) = \frac{2R}{V} - T_i^r$. As RSU_i gets $T_i^{dwell}(RSU_i)$ duration to serve R_i before Vh_i leaves the service range of RSU_i , $T_i^{dwell}(RSU_i)$ is the deadline of R_i at RSU_i , denoted by $T_i^{deadline}(RSU_i)$. Note that as delay tolerance is the upper bound of dwelling duration, it does not have impact on the deadline of a generated request, however, it does have impact to the transferred request, which is shown below.
- **Deadline of a transferred request:** Delay tolerance value of a request is the maximum delay bound assigned by a request within which the request must be satisfied. Beyond this time, the corresponding request will be removed from the system. When a request is transferred to another RSU, the request gets new deadline from its remaining delay tolerance value. The value of new deadline of the transferred request depends on the driving speed of the corresponding vehicle, blind zone distance, and the remaining delay tolerance value. If R_i is transferred to another RSU, e.g., RSU_j , at time t R_i will get the new deadline at RSU_j is,

$$T_i^{deadline}(RSU_j) = \min\{T^{Maxdwell}, \bar{T}_i^{DTolerance}(t)\} \quad (1)$$

A vehicle can generate request irrespective of receiving response of the previous submitted request which resembles the open system model [4, 30]. However, a vehicle Vh_i can generate a new request at RSU_i maintaining the following condition, $T_i^{in} \leq T_i^r \leq (T_i^{out} - T_i^{trans})$, that is a request should be generated by making sure that the RSU server gets the least enough time to transmit the requested data item before the corresponding vehicle leaves the transmission range of that RSU. To facilitate the formulation of the proposed ECLB approach, we first define some concepts as follows.

As time passes, the remaining delay tolerance value of a request reduces. The request must be served before the remaining delay tolerance becomes lower than the required serving time, in which case the request will be infeasible for serving, which is defined as follows.

Definition 1. *Infeasible Request for serving:* A request R_i becomes infeasible for serving when the remaining delay tolerance is smaller than the required transmission time of the requested data item, namely, $\bar{T}_i^{DTolerance}(t) < T_i^{trans}$ and it will then be discarded from the system and counted as deadline missed request.

Some requests may need to be served with higher urgency than other requests. The urgent request is defined as follows.

Definition 2. *Urgent Request:* If a request does not have enough remaining delay tolerance value to transfer and only be satisfied if it is served during the dwelling at the current RSU. That is, R_i is called an urgent request at t if, $\{\bar{T}_i^{DTolerance}(t)\} \geq T_i^{trans}$ but $\{\bar{T}_i^{DTolerance}(t)\} < \{T_{hop}^{travel} + T_i^{trans}\}$.

If a request is not an urgent request, it can be served with the help of the residual bandwidth of the suitable neighbor RSU. However it should be feasible for transferring. The definition of a feasible request for transfer is stated below.

Definition 3. *Feasible Request for Transfer:* If a generated request is not served in the current RSU, it needs to be transferred. However, the feasibility of request transfer to a target transferred RSU depends on the remaining delay tolerance value of the request. Again, remaining delay tolerance of a request is the function of distance from the current RSU to the target transferred RSU and the driving speed of the vehicle. A request R_i will be feasible for transfer if it has enough remaining delay tolerance to be served at the transferred RSU. R_i will be transferable to RSU_j from RSU_i at t if the following condition holds,

$$\{\bar{T}_i^{DTolerance}(t)\} \geq \{n_{hops} \times T_{hop}^{travel} + (n_{hops} - 1) \times T_i^{Maxdwell} + T_i^{trans}\} \quad (2)$$

Here $n_{hops} = 1, 2, 3, \dots$, where n_{hops} equals 1 means that the two RSUs are one hop apart and so on.

If a request does not have as much as remaining delay tolerance with which the request could be transferred to the suitable neighbor RSU in its route, but the request has at least as remaining delay tolerance with which it can be served by the immediate next neighbor RSU, then the request is called a potential request for serving by transfer. This type of request is defined as follows.

Definition 4. *Potential Request of Serving by Transfer:* Consider a request R_i . R_i is considered as a potential request of serving by transfer if it is not served at the current RSU but has enough remaining delay tolerance value to be served from the at least one hop apart neighbor RSU. Consider RSU_j is one hop apart from RSU_i . After t time from the generation of R_i , R_i is still be considered as a potential request of serving at RSU_j if the following condition holds,

$$\{\bar{T}_i^{DTolerance}(t)\} \geq \{T_{hop}^{travel} + T_i^{trans}\}$$

Upon receiving the requests in the service queue an RSU schedules requests before serving for improving data dissemination performance. Research shows that scheduling can improve the utilization of the service channel bandwidth, which impacts on the data dissemination performance [7, 16, 13]. As a request needs to be served before the corresponding vehicle moves out of the RSU range, the RSU should make sure that the request has been scheduled before the remaining dwelling duration of the vehicle becomes lower than the required transmission time of the requested data item.

Table 2: A running example of deadline conflicted request(s).

Request	Requested Data Item	Slack Time
R_1	d_1	1.5
R_2	d_3	1.0
R_3	d_5	3.0
R_4	d_1	1.7
Broadcast Bandwidth = 1.0		

Definition 5. *Schedule:* Assume that at time t a set of requests R^t resides in the RSU_i service queue to be scheduled. If a request $R_i \in R^t$ is scheduled to be served at t , we call R_i is satisfiable if

$$\{T_i^{dwell}(RSU_i) - t\} \geq T_i^{trans}$$

Let $(R_i, R^t)_*$ denotes the satisfiability of R_i . If this condition is violated, we call this is unsatisfiable at RSU_i .

Based on the remaining delay tolerance, an unsatisfiable request may be considered as infeasible for serving (Definition 1) and removed from the system, or be transferred to other RSU for serving (Definition 3).

By an efficient scheduling more than one request can be satisfied by a single broadcast, which can be defined as the sharable requests set as follows.

Definition 6. *Shareable Requests Set:* Due to the inherent nature of broadcast in wireless communication, when a data item is broadcast, the set of pending requests for the same data item can be satisfied at the same time. We call such a set of requests as shareable requests set. For instance, if $R_i \in R^t$ is scheduled at time t , its shareable requests set denoted by $SA(R_i, R^t)$ can be defined as,

$$SA(R_i, R^t) = \{R_j | \forall j \neq i, R_j \in R^t, d_j = d_i \ \& \ (R_j, R^t)_* = \text{satisfiable}\}$$

As by serving a request all other requests in the sharable set are served simultaneously, a scheduling algorithm should schedule the larger shareable requests set to maximize the broadcast channel bandwidth.

Definition 7. *Slack Time of a Request:* The remaining deadline of a request is called the slack time of a request.

Note that each request corresponds a deadline. Due to scheduling of one request ahead of other requests, some later scheduled requests with lower slack time may miss their deadlines. These deadline conflicted requests are defined as follows.

Definition 8. *Deadline Conflicted Request(s):* If due to the schedule of a request in the current broadcast other request(s) will miss their deadlines, namely if some requests do not have enough slack time to be served in the next broadcast, those prospective deadline missed requests are called the deadline conflicted request(s). If the conflicted request(s) would get the chance to be scheduled in the current broadcast they would not have missed their deadlines.

Consider Table 2. The attributes of the pending requests are given. All data items have equal size with unit 1.0, and RSU broadcast bandwidth also equals 1.0. It means that one data item

needs one time unit for transmitting. We assume that EDF scheduling is operating. Accordingly as R_2 has the most urgency (slack time equals 1.0), d_3 will be broadcast. For this, both R_1 and R_4 will miss their deadlines, because they have smaller slack time than time required for transmitting d_1 in the next broadcast. On the contrary, if d_1 would have been transmitted in the first broadcast, both R_1 and R_4 would be satisfied. So here both R_1 and R_4 are the deadline conflicted requests because of R_2 . The key to serve more requests is to minimize the number of deadline conflicted requests.

An unsatisfied request in the current RSU needs to be transferred to another RSU. Based on the mounting location, an RSU can be a junction-RSU or an Edge-RSU. The proposed ECLB approach efficiently balances the load among the junction-RSUs and edge-RSUs. The definitions of junction-RSU and edge-RSU are stated below.

Definition 9. *Junction-RSU and Edge-RSU: A junction-RSU refers to an RSU located at the intersection of roads. An intersection is a connecting point of different roads, so a junction-RSU usually receives more requests from more vehicles, as a result a junction-RSU can be overloaded easily. On the contrary, an edge-RSU refers to an RSU located at the non-intersection place of roads. An edge-RSU usually a less busy RSU than a junction-RSU.*

Refer to Fig. 3, for instance, RSU_1 , RSU_3 , RSU_6 , RSU_{10} , RSU_{12} , RSU_{15} are the junction-RSUs and RSU_2 , RSU_4 , RSU_5 , RSU_7 , RSU_8 etc. are the edge-RSUs.

4. A New Cooperative Approach

Before discussing the proposed ECLB approach, first we reiterate the existing CLB approach [12] and the limitations for performance maximization. Then, we outline the considerations that should be followed in the ECLB to overcome the limitations of CLB and further improve the data dissemination performance.

4.1. Existing approach: CLB

In the existing cooperative load balancing (CLB) approach while serving requests locally an RSU also transfers the overloaded requests to the neighbor RSUs [12]. The overloaded requests means those requests which are not satisfiable at the local RSU within the dwelling duration of the corresponding vehicles. However, these requests could be served by the neighbor RSUs if the requests have enough delay tolerance. The neighbor RSUs are those RSUs which the vehicle will traverse on its way towards the destination. The performance of the CLB approach heavily depends on the number of available neighbor RSUs for sharing the overload. We have identified that with the availability of more neighbor RSUs, system performance can be further improved. The performance maximization of the CLB approach is restricted due to the limited number of available neighbor RSUs. In the CLB approach the number of neighbor RSUs for load transferring of a particular vehicle depends on the next heading direction of the vehicle and the number of RSUs the vehicle will traverse until reaching the next intersection. For example, in Fig. 4, a vehicle is currently at RSU_1 and its destination RSU is RSU_{15} . From RSU_1 the vehicle can next head either towards the east direction or towards the north direction. For the east direction, the vehicle will pass through two RSUs (RSU_2 and RSU_3) upto reaching the next intersection namely, RSU_3 . So, these two RSUs are the neighbor RSUs (available for load transferring) for that vehicle towards the east direction, when the corresponding generated request(s) at RSU_1 becomes unsatisfiable. Moreover, among these two RSUs one is edge-RSU (RSU_2) and other one

is junction-RSU (RSU_3). As usually the intersection is the meeting points of different routes, a junction-RSU receives more requests than an edge-RSU. So, clearly RSU_1 has very limited number (only two) of available RSUs to transfer its overload. On the other hand, if the vehicle heads towards the north direction, the vehicle also passes through only two RSUs (one edge-RSU (RSU_9) and one junction-RSU (RSU_{10})) upto reaching the junction-RSU RSU_{10} . After reaching at the next intersection point, a vehicle will choose next which direction to head towards its destination, hence again a new set of neighbor RSUs will be available for load transferring from the RSU at the intersection. This is the approach for finding the available neighbor RSUs for load transferring in CLB. So, in this example from RSU_1 for either heading direction (east or north) for the vehicle there is only two available neighbor RSUs to serve the unserved requests. RSU_1 has to select one target transferee RSU among the available neighbor RSUs using the cooperative load transferring procedure. Finding a suitable target RSU among the available neighbor RSUs depends on the amount of pending workload of the RSUs. The objective is to find a suitable RSU so that the transferred requests does not become burden or becomes minimal burden to the transferred RSU. However, when number of available RSUs is smaller it is not easy to find such a suitable RSU.

4.2. Proposed ECLB approach

From the above observations, to further improve the system performance we have the following intuitive considerations:

- For a given destination, a driver does know which route it is following using the on-board GPS navigator. Through OBU a vehicle can inform the RSU about the selected route. Using the route information the RSU can calculate which next RSUs the vehicle is going to dwell within the remaining delay tolerance of the submitted request.
- When an RSU schedules a request, in addition of the metrics considered by a scheduling algorithm, the RSU also considers the remaining delay tolerance of the request and the available reachable RSUs in its route within the remaining delay tolerance of the pending request.
- The RSU gives the higher priority to the request with the lower delay tolerance and not possible to be served from the other RSU (Definition 2).
- The RSU gives the lower priority to the request which has higher remaining delay tolerance, which means the request also can be served from the other RSU if not served in the current RSU (Definition 3).
- If the request is not served at the current RSU, transfer the request to a suitable RSU in the route the vehicle is heading.
- When request transferring is taking place, the system also considers the dynamic direction change of a vehicle at the approaching intersection point to further load balance (if required) so that a rerouting vehicle can get its required data item.

The proposed ECLB approach finds the more available neighbor RSUs for load balancing. Intuitively, using the widely used onboard GPS navigation with loaded digital map a driver finds its route towards the destination [9]. As RSUs are the fixed unit mounted alongside road, it is also reasonable to assume that the system can identify beforehand how many RSUs a vehicle is

going to traverse in a certain route towards the destination. Using the GPS facility a vehicle can generate all the alternative routes towards the destination. After that the driver will choose which route to follow. The route selection may depend on various circumstances such as current traffic condition on the route. Upon selecting a route the current RSU gets more number of available neighbor RSUs, which provides a huge flexibility to transfer the overloaded requests. For example, from RSU_1 towards RSU_{15} there are ten alternative routes in Fig. 4, these are:

$route_1 = \{RSU_1 \sim RSU_7, RSU_{15}\};$
 $route_2 = \{RSU_1 \sim RSU_3, RSU_8, RSU_{12}, RSU_{11}, RSU_{10}, RSU_{18} \sim RSU_{24}, RSU_{16}, RSU_{15}\};$
 $route_3 = \{RSU_1 \sim RSU_3, RSU_8, RSU_{12} \sim RSU_{15}\};$
 $\vdots$
 $route_7 = \{RSU_1, RSU_9 \sim RSU_{12}, RSU_{17}, RSU_{21} \sim RSU_{24}, RSU_{16}, RSU_{15}\};$
 $\vdots$
 $route_{10} = \{RSU_1, RSU_9, RSU_{10}, RSU_{18} \sim RSU_{24}, RSU_{16}, RSU_{15}\};$

Assume that the vehicle selects $route_1$ for the destination (RSU_{15}). It will pass through seven RSUs ($RSU_2 \sim RSU_7, RSU_{15}$), among which four are edge RSUs ($RSU_2, RSU_4, RSU_5, RSU_7$) and three are junction RSUs (RSU_3, RSU_6, RSU_{15}). Hence in the ECLB approach for the vehicle, RSU_1 can choose the target transferee RSU among seven RSUs whereas for the CLB approach this was limited to among only two RSUs. This means that the ECLB has more flexibility to find the appropriate RSU for load transfer than the CLB approach. It is also not hard to understand that with the more available neighbor RSUs, in the ECLB approach an overloaded RSU can easily and more often finds the suitable target RSU to transfer its overload. So the ECLB approach can exploit more benefit of using cooperative load balancing than the CLB approach. The detailed cooperative load transferring procedure for finding the target transferred RSU among the available neighbor RSUs is stated in Appendix A.

It may be possible that in the ECLB approach a vehicle may deviate its route from the initial selected route, for instance, for the initial selected $route_1$ after leaving RSU_1 when the vehicle reaches at RSU_3 it may find that $route_1$ becomes congested. In that case from RSU_3 the vehicle again finds the suitable route (preferably less congested) from the available alternative routes towards the destination RSU_{15} . Now if the vehicle chooses a new route other than $route_1$, namely if the vehicle deviates its route from the initial selected route but its unserved request(s) has already been transferred to an RSU say RSU_4 which will not be traversed by the vehicle due to the route changed, the transferred request need to be dropped by RSU_4 . The dropping time of transferred request is determined by a timer which value is set usually slightly bigger than the traveling time of the vehicle (determined by the vehicle speed and the travel distance) of the vehicle from the current RSU to the target transferred RSU. A transferred request will be considered for scheduling in the transferred RSU only if a vehicle reaches at the transferred RSU before the timer expires. Upon receiving the BSMs from the vehicle the RSU realizes the presence of the vehicle in its service range. In addition, to readjust the load balancing (due to the route change possibility) in the ECLB approach, when a request is transferred to the target RSU, one copy is also sent to the next junction-RSU in the selected route. For example, if RSU_7 is the selected target transferred RSU among the available neighbor RSUs, the overloaded request will be transferred to both RSU_7 (as transferred RSU) and RSU_3 (as next junction-RSU). Note that here RSU_3 is not responsible for serving the request rather temporarily holds the request and only forwards the request to the next selected transferred RSU if the vehicle deviates from the initial selected route. The primary notations used in this section are given in Table 3.

Table 3: Summary of notations.

NOTATION	DESCRIPTION
RSU_i	A transferee RSU
RSU_j	A transferred RSU
RSU_{jun}	A junction RSU
T_i^{tm}	Setup timer for request R_i
$route_i$	A route from source to destination
$route(RSU_i(Vh_i))$	Set of routes for Vh_i from RSU_i to destination
$Neighbor(RSU_i(route_i))$	Set of RSUs in $route_i$ from RSU_i to destination

The summary of the ECLB steps is, an RSU can play three types of roles, *type 1 (current and transferee RSU)*: when it receives requests from the vehicles, the RSU tries to serve the request within the dwelling duration of the vehicle (Steps 1 and 2), and if the RSU cannot serve the request, it tries to transfer the request to the suitable RSU based on the heading direction of vehicle and the workload of the transferred RSU (Step 3). *Type 2 (transferred RSU)*: when the RSU receives the transferred requests, it waits for the corresponding vehicle to arrive before scheduling. Discard the request if the vehicle does not arrive within the setup time (Step 4). *Type 3 (junction RSU)*: when the RSU is a junction RSU and receives the request to hold temporarily. This RSU also may need to transfer the request if the corresponding vehicle reroutes from this RSU point (Step 5). The detailed steps of the proposed ECLB approach are stated as follows. The pseudo code is shown in Algorithm 1.

- **Step 1:** Schedule the urgent request (R_i) first among the pending requests (R') in the service queue. Here urgent request means the request which does not have enough remaining delay tolerance to transfer (Definition 2). If there is no more urgent request, start scheduling others.
- **Step 2:** Broadcast the requested data item (d_i) of the scheduled request. Update the status of the pending requests for d_i . Remove the already served requests from the service queue.
- **Step 3:** If a request R_i is not served at the current RSU_i and becomes unsatisfiable at current RSU_i (Definition 5), however, has enough remaining delay tolerance value to transfer (Definition 3), start the following load balancing process:
 - **Step 3.1** For the vehicle Vh_i , the current RSU_i , and the destination RSU_d among all the alternative routes towards RSU_d denoted by $route(RSU_i(Vh_i))$ select a route, $route_i \in route(RSU_i(Vh_i))$.
 - **Step 3.2** List all the RSUs of $route_i$ towards RSU_d as neighbor RSUs of RSU_i denoted by $Neighbor(RSU_i(route_i))$.
 - **Step 3.3** Using the cooperative load balancing procedure (Appendix A), subject to the remaining delay tolerance value of R_i which is enough to reach to RSU_j (according to Eq. (2)), find the target transferee RSU_j from the neighbor RSU set which is capable to serve the overloaded request(s) of RSU_i . Transfer the request(s) to RSU_j and to the next junction-RSU in the $route_i$, RSU_{jun} . Set up the timer T_i^{tm} .

- **Step 3.4** Otherwise, find an RSU_k from the neighbor RSU set which will have the minimum estimated workload (minimum $N_{usr}(RSU_k)$) upon Vh_i arrives there (Appendix A), transfer the overloaded request(s) to RSU_k and to the next junction-RSU in $route_i$, RSU_{jun} . Set up the timer T_i^{tm} .
- **Step 4:** Upon arriving Vh_i at transferred RSU_j or RSU_k , within the setup timer T_i^{tm} , consider transferred R_i for scheduling. If Vh_i does not reach transferred RSU_j or RSU_k within setup timer value T_i^{tm} , discard transferred request(s) from the transferred RSU (RSU_j or RSU_k).
- **Step 5:** If Vh_i deviates its direction at RSU_{jun} from the initial $route_i$, repeat steps 3.1 ~ 3.4 and step 4.
- **Step 6:** If at any time t remaining delay tolerance value becomes lower than the required transmission time of R_i , namely becomes infeasible for serving (Definition 1), remove R_i from the system.

For finding the complexity of ECLB we need to consider three cases, *case 1*: finding the route from the source RSU to the destination RSU; *Case 2*: finding the suitable request by scheduling; *Case 3*: finding the suitable target transferred RSU from the available neighbor RSUs. Assume that N_{RSU} and K denote the number of RSUs in the system and the number of edges (connected road between two RSUs) connecting the RSUs, respectively. The maximum number of pending requests at t is $|R'|$ (Definition 5). Case 1 is the DFS (depth-first search) search whose complexity is $O(N_{RSU} + K)$. Case 2 requires to find the suitable request R_i from R' whose complexity is $O(|R'|)$. Case 3 requires to find the suitable target RSU among the maximum possible neighbor RSUs whose complexity is $O(N_{RSU} - 1)$. So the overall complexity of ECLB is $O(N_{RSU} + K) + O(|R'|) + O(N_{RSU} - 1)$, which is linear and practical to implement.

4.3. Important features of ECLB

The proposed ECLB approach has the following important features for reducing the deadline missed requests and improving the system overall performance.

First: Load transferring using the ECLB approach has the reduced chance of deadline misses of other requests in the transferred RSU than the CLB approach.

For the same route and same amount of overload transferring, CLB may cause other requests to miss their deadlines at the transferred RSU, on the contrary, ECLB can serve the overloaded requests without causing other requests to miss their deadlines. For example, consider Fig. 3. A vehicle Vh_i does not get the service for request R_i during its dwelling duration at RSU_i . Assume that R_i has enough $\bar{T}_i^{DTolerance}(t)$ value towards its destination RSU_{15} . Using the ECLB approach, assume that Vh_i chooses $route_i$ towards the destination which comprises the set of RSUs, $Neighbor(RSU_i(route_i)) = \{RSU_2, \sim RSU_7, RSU_{15}\}$. When RSU_i calculates the suitable RSU for transferring R_i , it can choose the suitable transferred RSU from any of the RSUs from $Neighbor(RSU_i(route_i))$ (Appendix A). Assume that ECLB finds that RSU_7 will have enough residual bandwidth to serve R_i without hampering its own workload, when Vh_i will be there (Step 3.3 Algorithm 1). In this case, because of load transferring to RSU_7 there will be no hindrance for serving the own workload of transferred RSU.

Now if we consider CLB approach for load transferring, unlike ECLB it does not have more available RSUs in the neighbor RSU set. As we discussed earlier, it needs to select the possible

Algorithm 1: Enhanced Cooperative Load Balancing (ECLB) algorithm.

```
1  /*  $R^t(RSU_i)$ ,  $RSU_i$ , and  $RSU_j$  denote requests in the service queue of  $RSU_i$ , transferee  
   RSU and transferred RSU, respectively.  $Vh_i$  denotes the corresponding vehicle of  
    $R_i$ . */  
2  begin Scheduling_TransfereeRSU{ $R^t$ }  
3  /* Call by  $RSU_i$  at time  $t$  */  
4  while  $R^t(RSU_i) \neq \emptyset$  do  
5      /* Scheduling a set of requests  $R^t$ . /// Steps 1 and 2. */  
6      According to the scheduling algorithm schedule the most rewarded request  $R_i$  (say  $R_i$  is pending for  $d_i$ );  
7      Schedule from the urgent requests and continue until all urgent requests are served or become  
      unsatisfiable;  
8      Then schedule from other requests;  
9      Broadcast  $d_i$ ;  
10     Update  $R^t \leftarrow \{R^t - \{R_i + SA(R_i, R^t)\}\}$ ;  
11     if  $T_i^{deadline} < T_i^{trans}$  then  
12         /* deadline is expired at the current  $RSU_i$ . */  
13         if  $R_i$  has enough delay tolerance value to transfer, according to Eq. (2) then  
14             /*  $R_i$  is unsatisfiable at  $RSU_i$  but can be transferred for serving. */  
15             /// Step 3.  
16              $RSU_i \leftarrow \text{LoadBalancing}(Vh_i, R_i)$ ;  
17             Transfer  $R_i$  to  $RSU_i$ ;  
18             Setup timer  $T_i^{tm}$ ;  
19         else  
20             /*  $R_i$  doesn't have enough delay tolerance to be transferred, becomes  
             infeasible for serving. /// Step 6. */  
21             Remove  $R_i$  from the system.  
22         Update  $R^t \leftarrow \{R^t - \{R_i\}\}$ ;  
23         /* remove both the infeasible and transferred requests from  $RSU_i$  */  
24     end  
25 begin LoadBalancing{ $Vh_i, R_i$ }  
26 /* Steps 3.1 ~ 3.4. */  
27 For current  $RSU_i$ , destination  $RSU_d$  generate all the possible routes of  $Vh_i$  denoted by  $route(Vh_i(RSU_i))$ ;  
28 Randomly select a  $route_i \in route(Vh_i(RSU_i))$ ;  
29  $Neighbor(RSU_i(route_i)) \leftarrow$  all the RSUs in  $route_i$ ;  
30 for  $\forall RSU_j \in Neighbor(RSU_i(route_i))$  subject to the remaining delay tolerance value of  $R_i$  do  
31     Find  $RSU_j$  which satisfies (Equation A.2 in Appendix A):  
32      $\delta_j \cdot T_{max}^{dwell} \geq \{N_{usr}(RSU_j) + T_{max}^{dwell} \times (\frac{N_{vehicle}(RSU_j(\tau)) \cdot \lambda_v}{\tau} + \frac{N_{tr}(RSU_j(\tau))}{\tau})\}$ ;  
33     where  $N_{usr}(RSU_j)$  is calculated using equation A.1 in Appendix A;  
34     Return:  $RSU_j$ ;  
35 if there is no such  $RSU_j$ , find  $RSU_k \in Neighbor(RSU_i(route_i))$  which has minimum  $N_{usr}(RSU_k)$ ;  
36 Return:  $RSU_k$ ;  
37 end  
38 begin Scheduling_TransferredRSU{ $R^t + R'$ }  
39 /* at time  $t$ ,  $RSU_i$  received transferred requests set,  $R'$  from its neighbor RSUs. */  
40 while  $(R^t(RSU_i) + R') \neq \emptyset$  do  
41     /* Schedule a transferred  $R_i$  when its corresponding vehicle  $Vh_i$  in  $RSU_i$ 's  
     range. /// Step 4. */  
42     According to the scheduling algorithm schedule the most rewarded request  $R_i$  which request  $d_i$ ;  
43     Broadcast  $d_i$ ;  
44     Update  $\{R^t + R'\} \leftarrow \{\{R^t + R'\} - \{R_i + SA(R_i, R^t)\}\}$ ;  
45     if  $Vh_i$  does not arrive  $RSU_i$  within  $T_i^{tm}$  then  
46         /* discard  $R_i$ . */  
47         Update  $R^t \leftarrow \{R^t - R_i\}$ ;  
48     end  
49 end
```

```

47 begin Scheduling_JunctionRSU{ $R^t + R'$ }
48   /* At time  $t$ ,  $RSU_{jun}$  received requests set,  $R'$  from its neighbor RSUs. /// Step 5. */
49   if  $R_i$  is transferred to  $RSU_{jun}$  as a temporary transferred RSU then
50     if  $T_i^{tm}$  is expired then
51       if  $Vh_i$  deviates its route then
52         /*  $Vh_i$  deviates its route from  $RSU_{jun}$  */
53          $RSU_i \leftarrow \text{LoadBalancing}(Vh_i, R_i)$ ;
54         Transfer  $R_i$  to  $RSU_i$ ;
55         Setup new timer  $T_i^{tm}$ ;
56       /* discard  $R_i$  */
57       Update  $R^t \leftarrow \{\{R^t + R'\} - \{R_i\}\}$ ;
58 end

```

Figure 3: Load transferring scenario in CLB and ECLB.

Table 4: Summary of notations.

NOTATION	DESCRIPTION
μ_i	Arrival of sharable requests rate of R_i
T_i^{serv}	Required serving time of R_i
n_s	Number of serving round
$P(R_i^{shar})$	Probability that R_i is served as a sharable request
$P(R_i^{sch})$	Probability that R_i is served as a scheduled request

transferred RSU among the RSUs it travels upto reach to the next junction point, namely among RSU_2 and RSU_3 , that is, $Neighbor(RSU_i(route_i)) = \{RSU_2, RSU_3\}$. Assume that the chosen RSU is RSU_3 . Now it may be possible that RSU_3 has been chosen due to its minimum possible workload among the two RSUs, not because of its available residual bandwidth for serving R_i (Step 3.4 Algorithm 1). Next, R_i may or may not be served at RSU_3 , which depends on the underlying scheduling algorithm and the urgency of the pending requests in the service queue. If R_i is not served at RSU_3 , according to CLB next it will be transferred to any of the RSU among $RSU_4 \sim RSU_6$. If still it is not served, it needs to be transferred again. Let next transfer happened to RSU_7 , and there it is served, as RSU_7 has enough residue bandwidth to serve R_i . If this is the case, CLB also does not hinder any of the transferred RSUs to serve their own workload. Now, it is very likely that R_i will be scheduled and serve at any of the transferred RSUs before Vh_i reaching RSU_7 . If it happens, due to R_i is being scheduled at the transferred RSU, other deadline conflicted requests (Definition 8) miss their deadlines, although R_i could have been served at RSU_7 without interfering the serving of the other requests of the transferred RSU. Due to the transferred R_i , the conflicted requests at the transferred RSU turn to deadline missed requests.

Note that if a request is transferred to an RSU, the request can be served either as a sharable request, namely without being scheduled itself, or as a scheduled request. If the former case happens, the served request does not create the deadline conflict with other requests, however recalling Definition 8, if the request is served as a scheduled request at the transferred RSU, there is a chance of deadline conflict with the other requests at that RSU. As even for the same route CLB transfers an unserved request more number of times than ECLB, CLB has a higher chance of creating deadline conflict at the transferred RSU than ECLB. Below we show that a request for each transfer increases the chances of being served as a scheduled request, in other words increases the chance of deadline conflict of other requests at the transferred RSU. However, before that we have to find out at what probability the request might be served as a sharable request at that transferred RSU. Thus, we have the following lemmas.

Lemma 1. *If a request R_i of vehicle Vh_i is transferred to another RSU_j , R_i will be satisfied as a sharable request with a probability of $P(R_i^{shar})$.*

Proof. Assume requests generation rate per vehicle in RSU_j is λ_j , shareable request arrival rate of a request R_i is μ_i , average service time of a request is T_i^{serv} , and average number of vehicles dwell inside RSU_j is $N_{Vh}(RSU_j)$. Hence within the dwelling duration of R_i , namely (T_i^{dwell}) , the total number of serving round at RSU_j is $n_s = \lfloor \frac{T_i^{dwell}}{T_i^{serv}} \rfloor$. The primary notations used in this section are stated in Table 4.

In each serving round, e.g., T_i^{serv} , the RSU_j received queue receives $N_{Vh}(RSU_j) \cdot \lambda_i \cdot T_i^{serv}$ re-

quests, among which $\mu_i.T_i^{serv}$ are shareable requests of R_i (Definition 6). Recalling that if any shareable request of R_i is served, R_i also will be served at the same time. Hence, the probability that R_i or any of its shareable requests will be served in the current broadcast is $P(R_i^{shar}(T_i^{serv}(1)))$ is,

$$P(R_i^{shar}(T_i^{serv}(1))) = \frac{\mu_i.T_i^{serv} + 1}{N_{Vh}(RSU_j).\lambda_i.T_i^{serv}}$$

While completing the previous broadcast, $N_{Vh}(RSU_j).\lambda_i.T_i^{serv}$ new requests will arrive which includes $\mu_i.T_i^{serv}$ shareable requests of R_i . The probability that in this broadcast (T_2^{serv}), R_i or any of its sharable requests will be served denoted as $P(R_i^{shar}(T_2^{serv}))$,

$$P(R_i^{shar}(T_i^{serv}(2))) = \frac{2\mu_i.T_i^{serv} + 1}{2.N_{Vh}(RSU_j).\lambda_i.T_i^{serv} - (1 + |SA(R_i, R^t)|)}$$

Similarly, in the k^{th} broadcast, R_i or any of its sharable requests will be served denoted as $P(R_i^{shar}(T_i^{serv}(k)))$,

$$P(R_i^{shar}(T_i^{serv}(k))) = \frac{k.\mu_i.T_i^{serv} + 1}{k.N_{Vh}(RSU_j).\lambda_i.T_i^{serv} - (k-1)(1 + |SA(R_i, R^t)|)}$$

Hence, the probability that R_i will be served as a sharable request at RSU_j within its dwelling duration is,

$$P(R_i^{shar}) = 1 - \prod_{k=1}^{n_s} \left(1 - \frac{k.\mu_i.T_i^{serv} + 1}{k.N_{Vh}(RSU_j).\lambda_i.T_i^{serv} - (k-1)(1 + |SA(R_i, R^t)|)} \right)$$

□

Lemma 2. If R_i is transferred to another RSU, its serving probability as a scheduled request increases by a factor of $P(R_i^{sch})$.

Proof. If R_i is transferred to another RSU_j , there are three possible cases:

- case 1: R_i is served as a sharable request without being scheduled itself. We denote this probability by $P(R_i^{shar})$;
- case 2: R_i is served as a scheduled request. We denote this probability by $P(R_i^{sch})$;
- case 3: R_i is not served at RSU_j .

Let A denotes " R_i is served as a sharable request", and B denotes " R_i is served as a scheduled request". Hence $\bar{B}$ denotes " R_i is not served as a scheduled request". Accordingly from the above three cases,

$$P(R_i^{sch}) = P(B) = P(B|A)P(A) + P(B|\bar{A})P(\bar{A}) \quad (3)$$

According to the scheduling principal, if R_i is already served as a sharable request, it can not be served as a scheduled request, hence $P(B|A) = 0$. So, we can write the simplified form of $P(B)$ as,

$$\begin{aligned} P(B) &= P(B|\bar{A})P(\bar{A}) \\ &= P(B|\bar{A})(1 - P(R_i^{shar})) \end{aligned}$$

Now we need to find $P(B|\bar{A})$, namely the probability of R_i is served as a scheduled request under the given condition of R_i is not served as a shareable request. Recalling from Lemma 1 the maximum possible number of serving round, R_i will get to be satisfied at RSU_j is n_s . Hence, the probability of R_i is not served as a scheduled request under the given condition of R_i is not served as a shareable request is $(1 - P(R_i^{shar}))^{n_s}$. So, $P(B|\bar{A}) = 1 - (1 - P(R_i^{shar}))^{n_s}$. From Eq. (3),

$$\begin{aligned} P(B) &= P(R_i^{sch}) \\ &= (1 - (1 - P(R_i^{shar}))^{n_s})(1 - P(R_i^{shar})) \end{aligned}$$

Assume that R_i is transferred N_{RSU} times, the probability that R_i is served as a scheduled request at RSU_j is $P(R_i^{sch}(RSU_j))$, that is,

$$P(R_i^{sch}(RSU_j)) = (1 - P(\mathbb{1}))(1 - P(\mathbb{2})) \cdots P(\mathbb{j}) \cdots (1 - P(\mathbb{N}))$$

where, $P(\mathbb{N}) = P(R_i^{sch}(RSU_N))$, denotes the probability that R_i is served as a scheduled request at RSU_N , $N \in N_{RSU}$. Hence if R_i is transferred to RSU_j , its serving probability as a scheduled request increases, specifically by factor of $P(R_i^{sch}(RSU_j))$. In the performance evaluation section, we will show this analysis in quantitative form. $\square$

Second: If a vehicle changes its predefined route in the middle, the ECLB approach can readjust the workload. The ECLB approach always performs better or as good as the CLB approach.

The performance improvement of ECLB over CLB comes due to the more flexibility of load balancing, namely more neighbor RSUs towards the destination for making load balancing decision. If a vehicle deviates from the initial route this flexibility may be reduced, namely the number of valid neighbor RSUs for load balancing may be reduced. However, this reduced number of neighbor RSUs will never be less than the available neighbor RSUs of CLB. Hence, the performance of ECLB will never fall below CLB. For example, consider Fig. 3. Assume that the ECLB approach is operating. Vehicle Vh_i selects $route_i$ (which comprises of $\{RSU_2 \sim RSU_7, RSU_{15}\}$) from its starting point (RSU_1) towards its destination (RSU_{15}). Assume that R_i is not satisfied at RSU_1 , but considering the remaining delay tolerance of R_i and using the load balancing procedure in ECLB request R_i is transferred to RSU_7 . Now it may happen that Vh_i changes its route from the next junction point, namely at RSU_3 and chooses another route which consists of $\{RSU_8, RSU_{12} \sim RSU_{15}\}$. Here complicity arises that the corresponding request (R_i) of Vh_i is transferred to RSU_7 , but due to changing the route Vh_i will not visit RSU_7 . So, there are two challenges to meet. The first challenge is how and when RSU_7 understands that the transferred R_i is no longer valid at RSU_7 . The second challenge is how R_i will be transferred to appropriate RSU according to the new heading direction of Vh_i . To overcome this situation, recalling that ECLB has two properties. Firstly, when a request is transferred, the transferred RSU maintains a timer which is usually a little bigger than the traveling time of Vh_i from the transferee RSU to the transferred RSU. When the timer expires, R_i will be removed from RSU_7 . So the first challenge is met. Secondly, when a request is transferred to the transferred RSU, ECLB also sends the copy of that request to the next junction-RSU. Hence, when Vh_i deviates its route from RSU_3 , R_i will be transferred to any of the RSU ($RSU_8, RSU_{12} \sim RSU_{15}$) in its new route according to the load balancing procedure and remaining delay tolerance value. So, the second challenge is also met.

Table 5: Simulation parameters.

Parameter	Default	Range	Description
N_{RSU}	24	—	Number of RSUs in the simulation topology
N_{Vh}	70	30-130	Generated number of vehicles
Poisson mean (μ)	—	20-30	Mean number of requests generated by a vehicle per RSU
VGIV	0.50	—	Vehicle generation interval (Exponential distribution)
RGIV	0.75	—	Request generation interval (Exponential distribution)
m	1000	—	Number of data items in the database
R	300	100-600	RSU communication range (m)
V	40	—	Vehicle average speed (km/h)
Distance	100	—	Inter-RSU distance (m)
SD	5.6	—	Safety distance between vehicles (m)
THETA (θ)	0.6	0.0-1.0	Zipf distribution parameter
a, b	4,3	2~6, 1~5	The Min. and max. laxity for calculating delay tolerance
TimeWindow (τ)	5	—	Duration of time window for calculating EWMA (sec)

4.4. Scheduling algorithms

An efficient scheduling algorithm plays a significant role for the effective and the time efficient data dissemination. A scheduling algorithm decides which pending request(s) will be served in the next broadcast. So the integration of a chosen scheduling algorithm with the load balancing system has the impact on the overall system performance. As a representative of on-demand scheduling algorithms, we integrate the following algorithms with the proposed ECLB, the CLB and the standalone approaches for performance evaluation, which will be discussed in Section 5.

(1) First Come First Served (FCFS) [31]: This is a base line scheduler. It selects a request according to the request arrival order in the RSU service queue. As FCFS does not consider the urgency of request, it may not perform well in the real-time environment such as in VANETs where requests have deadlines. FCFS is used here as a representative of non-real-time scheduling algorithm and to demonstrate the inability of a non-real-time algorithm in VANETs.

(2) Earliest Deadline First (EDF) [33]: EDF serves the request with the highest urgency. It broadcasts the data item of a corresponding request which has the lowest slack time. As EDF considers the urgency of request, it is called a real-time scheduling algorithm.

(3) Slack time Inverse Number of pending queries (SIN) [16]: SIN serves a request with the smallest SIN value, where SIN value is defined as the ratio of the slack time of the most urgent request of a data item to the popularity of the requested data item. Recalling that the slack time is the remaining deadline value of a request (Definition 7). Hence SIN integrates the urgency of a request and the popularity of the requested data item. SIN is a real-time scheduling algorithm.

5. Performance Evaluation

5.1. Simulation model

Our simulation model is based on the system architecture as shown in Fig. 1 and implemented using CSIM19 [36]. The used explicit parameters for the simulations are shown in Table 5. The

Figure 4: Simulated Multiple RSUs Area.

simulation environment is set up based on the topology as shown in Fig. 4. Vehicle mobility in the simulation region follows the Manhattan mobility model [37]. A vehicle generates requests and receives responses until it leaves the RSU transmission range. The deadline of a generated request is calculated as follows:

$$T_i^{deadline}(RSU_i) = \frac{2R}{V} - T_i^r$$

T_i^r varies from request to request. It depends when the request is generated during the dwelling period of the corresponding vehicle inside an RSU service range.

For assigning the delay tolerance value to a request R_i we use the following formula:

$$T_i^{DTolerance} = rand(T_i^{Maxdwell}, (a \times T_i^{Maxdwell} + b \times T_{hop}^{travel})) \quad (4)$$

Here, $T_i^{Maxdwell}$ and $(a \times T_i^{Maxdwell} + b \times T_{hop}^{travel})$ are the minimum and the maximum range, respectively for calculating the delay tolerance value of a request. For varying the delay tolerance, we vary the maximum range value by varying the value of a and b .

We generate vehicles from the bottom left and the top right corner of the simulation area on the opposite directed lanes (Fig. 4) and let the vehicles move. The Vehicle Generation Interval (VGIV) from each generation point follows the Exponential distribution. A vehicle arbitrarily picks up an exit point in the simulation area (Fig. 4) as its destination. In the ECLB approach, from all the alternative routes towards the destination, a vehicle will randomly choose any route as its initial route and start routing. At any junction point a vehicle can change its heading direction which is determined by the probabilistic model of the Manhattan mobility [37]. While passing an RSU a vehicle generates requests and receives responses, however when it reaches the destination point, it will not generate any further requests and will be removed from the simulation. Each road consists of two lanes and vehicles in the same lane always maintain a safety distance (SD).

In simulation, we follow the DSRC communication characteristics, specifically the radius of RSU communication range is set to 300 m, inter-vehicle safety distance is set to 5.6 m, and the blind zone distance between two RSUs is set to 100 m. Recalling that in this work we focus on the impact of load balancing on the system performance. Hence to clearly visualize the advantage of cooperative load balancing over the standalone approach and to provide the same VANETs environment to all the approaches, we assume that there is no channel fading or path loss. In the RSU database each data item size equals 1.0 unit. The data channel bandwidth of an RSU equals to 1.0, that means an RSU broadcasts one data item per unit time. Request generation follows the Poisson process, with the Poisson mean request generation μ value ranging from 20 to 30 for each vehicle. The data item access pattern in the RSU database for the generated request follows the Zipf [38] distribution, where the skewness is controlled by the parameter θ . The access probability of the i^{th} data item is $\frac{1}{\sum_{j=1}^m \frac{1}{j^\theta}}$, $1 \leq i \leq m$; $0 \leq \theta \leq 1$. θ equals 0 means the Uniform distribution and the increasing θ means the more skewed distribution. The simulation completes when all the generated vehicles exit the simulation topology. We continued the simulation until the system came into a steady state. For the performance analysis we take the average data of ten simulation runs for the same simulation parameter settings with the different seed values.

5.2. Performance metrics

We adopt the following performance metrics for the performance analysis.

(1) Deadline Miss Ratio (DMR): The ratio of the total number of requests missing the assigned delay tolerance value to the total number of generated requests. The lower DMR value means the more number of generated requests receive their requested data items successfully within the stipulated tolerance time. The ratio of the number of missed requests at the junction-RSUs to the total number of generated requests at the junction-RSUs is called the DMR at the junctions RSUs. Similarly, the ratio of the number of missed requests at the edge-RSUs to the total number of generated requests at the edge-RSUs is called the DMR at the edge RSUs.

(2) Success Rate of request Transfer Ratio (SRTR): The ratio of the number of satisfied transferred requests to the total number of transferred requests. The higher SRTR value means more number of transferred requests are successfully served at the transferred RSUs before the assigned delay tolerance values expire.

(3) Percentage Conflict Request Count per Broadcast: The percentage average number of requests are conflicted due to the scheduled request per broadcast. Recalling that the conflicted requests are those request which are going to miss their deadlines before completing the transmission of the next broadcast data item (Definition 8).

Figure 5: Impact of workload.

5.3. Performance Analysis

In the following we analyze the comparative performances of the proposed ECLB, CLB and standalone approaches. On the one hand, the comparison between ECLB and CLB approaches demonstrates the superiority of the proposed enhanced load balancing approach over the existing load balancing approach. On the other hand, the comparison between the load balancing approach and the standalone approach demonstrates the necessity of cooperative load balancing among the RSUs in vehicular network environment. Note that each of these approaches is integrated with three well-known on-demand scheduling algorithms: FCFS, EDF and SIN. The corresponding descriptions of these algorithms are shown in Section 4.4.

5.3.1. Impact of workload

Fig. 5 shows the impact of workload in terms of number of vehicles on the performance of different approaches. Fig. 5(a) shows the DMR of different scheduling algorithms when RSUs are operating in standalone manner, namely without load transferring. Among the three scheduling algorithms SIN performs the best, followed by EDF. FCFS performs the worst. This results support the finding in [16]. As SIN considers both the request urgency which is an important metric for real-time scheduling, and the popularity of the data item which is important for maximizing the use of the broadcast channel bandwidth, it performs better than EDF which considers only the request urgency. Furthermore, FCFS neither considers the request urgency nor considers the popularity of data item, it only considers request waiting time which is not a crucial criteria for real-time scheduling. Hence, FCFS ranks the last.

Fig. 5(b) shows the deadline miss ratio (DMR) of different scheduling algorithms with the CLB and ECLB approaches under different workloads. With the increasing number of vehicles, the system workload increases. An RSU service queue receives more number of requests, hence the DMR increases. However, a scheduling algorithm with the ECLB approach shows significantly lower DMR value than with the CLB approach under different workloads. This verifies the superiority of our proposed ECLB approach over CLB approach. Note that in the relative performance difference of the different scheduling algorithms with the ECLB approach, ECLB-SIN ranks the best followed by ECLB-EDF and ECLB-FCFS places the last. Although ECLB-EDF

(a) DMR of the standalone approach with different scheduling algorithms.

(b) DMR of different approaches (Standalone vs. CLB vs. ECLB) with SIN.

Figure 6: Overall DMR vs. DMR due to the potential requests under different workload.

(a) DMR at the Junction-RSU vs. Edge-RSU with the standalone approach. (b) DMR at the Junction-RSU vs. Edge-RSU with SIN.

(c) DMR at the Junction-RSU vs. Edge-RSU with EDF. (d) DMR at the Junction-RSU vs. Edge-RSU with FCFS.

Figure 7: DMR at the Junction-RSU vs. Edge-RSU with different scheduling algorithms.

Figure 8: SRTR and conflict request count under different workload.

has close performance of ECLB-SIN under different workloads, ECLB-FCFS ranks far behind ECLB-SIN. This identifies the inability of the non-real-time algorithm in the VANETs environments. With the CLB approach, SIN, EDF and FCFS also have the same relative performances. Nonetheless a standalone approach endures a very high DMR in compare to the CLB and ECLB approaches as shown in Fig. 5(a), which supports the necessity of cooperative load balancing approach in RSU-based VANETs.

Fig. 6 shows the insight reasons of performance difference between the standalone and the cooperative approaches. Fig. 6(a) exhibits the comparative performance of *overall DMR* and *DMR only due to the potential requests of serving by transfer* of the algorithms under different workload with the standalone approach. Each algorithm has the almost equal value of overall DMR and the DMR due to the potential requests of serving by transfer. It infers that the deadline misses of the potential requests are the main contributor in the overall DMR. Recalling our previous discussion (Definition 4), a *potential request of serving by transfer* is a request which has enough remaining delay tolerance value which could be served from at least one hop apart RSU, if it cannot be served at the current RSU. So, it also infers that the standalone approach fails to satisfy those requests which had the sufficient delay tolerance value, and these requests could be satisfied with the help of neighbor RSUs. With the presence of load balancing among RSUs both the CLB and ECLB approaches reduces the deadline misses of the potential requests, hence overall DMR reduces which is shown in Fig. 6(b). Nevertheless, the ECLB approach performs far better than the CLB approach.

Now we will show the reasons of performance difference between the CLB and the ECLB approaches. Fig. 7 shows the DMRs at the junction-RSUs and edge-RSUs under different workloads. Before applying load balancing approach the difference between the DMR at the junction-RSU and at the edge-RSU is not significant (Fig. 7(a)). However, after applying load balancing, although overall DMR improves in the CLB approach than the standalone approach, the difference between the DMR at the junction-RSU and at the edge-RSU increases significantly as shown in Figs. 7(b)~ 7(d). So, there is still room for improvement in the load balancing. This inspires us to devise a new approach, ECLB which can reduce the difference between the DMR

at the junction-RSU and at the edge-RSU by applying load balancing with a more flexible way and improves the overall DMR further. RSUs at the junctions with the CLB approach endures very higher DMR than the DMR of RSUs at the edges. On the contrary, in the ECLB approach the DMRs at the junction-RSUs are not significantly higher than that of the edge-RSUs (Figs. 7(b)~ 7(d)). In addition, the DMRs with the ECLB approach at the junction-RSUs are significantly lower than the DMRs of junction-RSUs with the CLB approach. The results point out our claims of providing more flexibility in load transferring of RSUs in the ECLB approach than the CLB approach. Hence, in the ECLB approach RSUs at the junctions can distribute its workload to the edges' RSUs more freely. So, the ECLB approach finds the more suitable RSU with the residual bandwidth than the CLB approach for serving the transferred request. Fig 8(a) shows the superiority of the ECLB approach over the CLB approach in respect of another performance metric, Success Rate of request Transfer Ratio (SRTR). It demonstrates that in the ECLB approach more transferred requests are served successfully than the CLB approach. Moreover, by having the more flexible load balancing scope, ECLB can minimize the number of deadline conflicted requests (Section 4.3) as shown in Fig. 8(b). Recalling that the deadline conflicted request(s) (Definition 8) are those requests which are going to miss their deadlines before the completion of the transmission of the next broadcast data item. These are the major reasons why a scheduling algorithm with the ECLB approach experiences a significantly lower DMR than with the CLB approach which is shown in Fig. 5(b).

5.3.2. Impact of RSU transmission range

Fig. 9 exhibits the impact of RSU transmission range on the system performance. A shorter RSU transmission range means a vehicle dwells a shorter duration inside the RSU service region. Hence, with the smaller value of RSU transmission range the system endures a higher DMR. On the contrary, the DMR declines with the increasing RSU transmission range as shown in Figs. 9(a) and 9(b). Similar to the previous results, a scheduling algorithm with the ECLB approach exhibits the lower DMR values than with the CLB approach under different RSU transmission ranges. Although with the increasing transmission range both in the standalone and cooperative (CLB and ECLB) approaches DMRs reduce, the DMR reducing pace is much slower in the standalone approach (Fig. 9(a)) than in the cooperative approaches (Fig. 9(b)). This is because, if the RSU transmission range increases, both in the CLB and ECLB approaches not only at the current RSU but also at the transferred RSU, a vehicle dwells longer time, which provides a request more time to be satisfied. On the contrary, in the standalone approach only at the current RSU, a request gets more time to be satisfied. Nevertheless, a scheduling algorithm with the ECLB approach outperforms both with the CLB and the standalone approaches (Fig. 9(b)), because in the ECLB approach more transferred requests are successfully served than the CLB approach (9(c)).

5.3.3. Impact of data item access pattern

Fig. 10 shows the impact of the data item access pattern on the system performance. Skewness parameter, $THETA$ (θ) equals 0 means the data item access pattern is random, which means all the data items are accessed with the equal probability. With the increasing value of $THETA$, the data item access pattern becomes more skewed which means the probability of some of data items being accessed increases. Similar to the previous studies [30, 16, 39], with the increasing $THETA$, all the algorithms have the improved performance. However, from Figs. 10(a) and 10(b), it is clearly understandable that under different data item access patterns a scheduling algorithm with the cooperative approach (both CLB and ECLB) shows a better performance than

(a) DMR with the Standalone approach.

(b) DMR with the CLB and ECLB approaches.

(c) SRTR with the CLB and ECLB approaches.

Figure 9: Impact of transmission range of an RSU.

(a) DMR with the Standalone approach.

(b) DMR with the CLB and ECLB approaches.

(c) SRTR with the CLB and ECLB approaches.

Figure 10: Impact of data item access pattern.

Figure 11: Impact of delay tolerance value.

with the standalone approach. However with the ECLB approach a scheduling algorithm shows superior performance than with the CLB approach (Fig. 10(b)). With the ECLB approach a scheduling algorithm gains better SRTR than with the CLB approach (Fig. 10(c)). The relative performance of the scheduling algorithms in respect of DMR and SRTR shows the consistent results of our previous set of experiments.

5.3.4. Impact of delay tolerance value

Fig. 11 shows the impact of the delay tolerance value of a request in the system performance. With the increasing value of a and b (Eq. (4)), a request can assign a higher delay tolerance value. A request with a higher delay tolerance value means it remains feasible for serving more duration in the system (Definition 3). Hence an unsatisfied request gets increased chances to be transferred more time to be satisfied, namely load balancing system can operate more efficiently. From Fig. 11(c), with the increasing value of delay tolerance more transferred requests are satisfied. This impacts positively on the whole system performance of the CLB and the ECLB approaches (Fig.

11(b)). However, as the standalone approach does not transfer any request, it does not take the benefit of the increased delay tolerance value (Fig. 11(a)). Nevertheless, an algorithm with the ECLB approach outperforms both the CLB and the standalone approaches and ECLB with SIN performs the best. The relative performance of the algorithms remains consistent with previous results.

6. Conclusion and Future Work

VANETs is an increasingly popular and emerging technology in the Intelligent Transportation System (ITS). Many safety and non-safety related applications have been envisioned in VANETs, however efficient data dissemination is necessary to make these applications in reality. One of the main characteristics of VANETs is the mobility of vehicle, which makes the data dissemination challenging. There are many techniques have been proposed to tackle this challenge. Installing RSUs along roadside is one of them. However, it is not unusual that an RSU can be overloaded at the rush hours by the passing vehicles. A cooperative load balancing (CLB) approach [12] can play an effective role in this scenario which can use the residual bandwidth of the neighbor RSU(s) cooperatively to serve the overloaded requests. However, the existing CLB approach does not have much flexibility of load transferring which restricts the maximization of system performance. In this paper, we investigate that with considering the remaining delay tolerance of submitted requests and the knowledge of fixed road layout, the performance of the cooperative load balancing system can be further improved significantly. In this paper, we propose an enhanced CLB (ECLB) approach. The ECLB approach can balance the overload of junction-RSUs and edge-RSUs more freely and can maximize the overall system performance. The ECLB approach reduces the number of deadline conflict requests which helps positively to improve the overall system performance. The ECLB approach even can operate efficiently if a vehicle deviates from the initial selected route in the middle of the routing. Simulation results support our claims and provide a better result for the proposed ECLB compared to the existing CLB and standalone approaches under different circumstances.

In this work, we consider satisfying as many requests as possible considering the urgency of request and the mobility of vehicle, however, we do not consider the status of the request-submitting vehicle. In the future work we may focus on providing fair service in vehicle-level, so that all vehicles may have chances to receive services from RSUs. Note that achieving both higher system performance and higher fairness has a trade-off. An efficient balanced system should be devised, which can balance these two inter-conflicting performance metrics. Again, we want to study the data dissemination performance of the proposed approach with multi-item requests where request size can vary. Moreover, the impacts of MAC and PHY layers and potential security vulnerabilities on data dissemination should be incorporated to validate the model in realistic vehicular network environment.

Acknowledgement

This work was partially supported by a grant from EDB, Singapore (Project: Development of NTU/NXP-Intelligent Transport System Test-Bed Living Lab Fund (LLF)).

References

- [1] E. Schoch, F. M. Kargl, T. Leinmüller, Communication patterns in VANETs, *IEEE Communication Magazine* 46 (11) (2008) 2–8.
- [2] H.-J. Cho, Continuous range k-nearest neighbor queries in vehicular ad hoc networks, *Journal of Systems and Software* 86 (5) (2013) 1323 – 1332. doi:<http://dx.doi.org/10.1016/j.jss.2012.12.034>. URL <http://www.sciencedirect.com/science/article/pii/S0164121212003433>
- [3] A. Bohm, M. Jonsson, Supporting real-time data traffic in safety-critical vehicle-to-infrastructure communication, in: *Proceedings of 33rd IEEE Conference on Local Computer Networks (LCN'08)*, Montreal, Que, 2008, pp. 614–621.
- [4] G. N. Ali, E. Chan, W. Li, Supporting real-time multiple data items query in multi-rsu vehicular ad hoc networks (vanets), *Journal of Systems and Software* 86 (8) (2013) 2127–2142. doi:10.1016/j.jss.2013.03.073.
- [5] Q. Xu, T. Mak, J. Ko, R. Sengupta, Vehicle-to-vehicle safety messaging in dsrc, in: *Proceedings of the 1st ACM international workshop on Vehicular ad hoc networks (VANET'04)*, 2004, pp. 19–28.
- [6] K. Liu, V. C. Lee, Rsu-based real-time data access in dynamic vehicular networks, in: *Proceedings of the IEEE Annual Conference on Intelligent Transportation Systems (ITSC'10)*, Portugal, 2010, pp. 1051–1056.
- [7] K. Liu, J. Ng, V. Lee, S. Son, I. Stojmenovic, Cooperative data scheduling in hybrid vehicular ad hoc networks: Vanet as a software defined network, *Networking, IEEE/ACM Transactions on PP* (99) (2015) 1–1. doi:10.1109/TNET.2015.2432804.
- [8] J. Kenney, Dedicated short-range communications (dsrc) standards in the united states, *Proceedings of the IEEE* 99 (7) (2011) 1162–1182. doi:10.1109/JPROC.2011.2132790.
- [9] M.-F. Tsai, P.-C. Wang, C.-K. Shieh, W.-S. Hwang, N. Chilamkurti, S. Rho, Y. S. Lee, Improving positioning accuracy for vanet in real city environments, *The Journal of Supercomputing* (2014) 1–21.
- [10] Y. Zhang, G. Cao, V-pada: Vehicle-platoon-aware data access in vanets, *Vehicular Technology, IEEE Transactions on* 60 (5) (2011) 2326–2339. doi:10.1109/TVT.2011.2148202.
- [11] G. Md Nawaz Ali, M. Salim Mollah, S. Samantha, S. Mahmud, An efficient cooperative load balancing approach in rsu-based vehicular ad hoc networks (vanets), in: *Control System, Computing and Engineering (ICCSCE)*, 2014 IEEE International Conference on, 2014, pp. 52–57. doi:10.1109/ICCSCE.2014.7072688.
- [12] G. Ali, E. Chan, W. Li, On scheduling data access with cooperative load balancing in vehicular ad hoc networks (vanets), *The Journal of Supercomputing* 67 (2) (2014) 438–468. doi:10.1007/s11227-013-1011-y.
- [13] D. Aksoy, M. Franklin, $R \times W$: A scheduling approach for large-scale on-demand data broadcast, *Journal of IEEE/ACM Transactions on Networking (TON)* 7 (6) (1999) 846–860.
- [14] Y. Zhang, J. Zhao, G. Cao, Service scheduling of vehicle-roadside data access, *Mobile Networks and Applications* 15 (1) (2010) 83–96.
- [15] J. Fernandez, K. Ramamritham, Adaptive dissemination of data in time-critical asymmetric communication environment, *Journal of Mobile Networks and Application* 9 (5) (2004) 491–505.
- [16] J. Xu, X. Tang, W. Lee, Time-critical on-demand data broadcast algorithms, analysis and performance evaluation, *IEEE Transactions on Parallel and Distributed Systems* 17 (1) (2006) 3–14.
- [17] K. Liu, V. Lee, Adaptive data dissemination for time-constrained messages in dynamic vehicular networks, *Transportation research part C: emerging technologies* 21 (1) (2012) 214–229.
- [18] C. Lochert, B. Scheuermann, M. Caliskan, M. Mauve, The feasibility of information dissemination vehicular ad-hoc networks, in: *Proceedings of the 4th Annual Conference on Wireless On-demand Network Systems and Services (WONS'07)*, Oberguergl, 2007, pp. 92–99.
- [19] M. Shahverdy, M. Fathy, S. Yousefi, Scheduling algorithm for vehicle to road-side data distribution, *Special issue of Communications in Computer and Information Science* 66 (2010) 22–30.
- [20] Q. Wang, P. Fan, K. Letaief, On the joint V2I and V2V scheduling for cooperative VANETs with network coding, *IEEE Transactions on Vehicular Technology* 61 (1) (2012) 62–73.
- [21] T.-Y. Wu, M. S. Obaidat, H.-L. Chan, Qualityscan scheme for load balancing efficiency in vehicular ad hoc networks (vanets), *Journal of Systems and Software* 104 (2015) 60 – 68. doi:<http://dx.doi.org/10.1016/j.jss.2015.01.052>. URL <http://www.sciencedirect.com/science/article/pii/S0164121215000321>
- [22] T. Nadeem, S. Dashtinezhad, C. Liao, L. Iftode, Trafficview: Traffic data dissemination using car-to-car communication, *ACM SIGMOBILE Mobile Computing and Communications Review* 8 (3) (2004) 6–19.
- [23] T. Nadeem, P. Shankar, L. Iftode, A comparative study of data dissemination models for VANETs, in: *Proceedings of the 3rd Annual International Conference on Mobile and Ubiquitous Systems - Workshops*, San Jose, CA, 2006, pp. 1–10.
- [24] J. Wu, R. Fujimoto, R. Guensler, M. Hunte, MDDV: A mobility-centric data dissemination algorithm for vehicular networks, in: *Proceedings of the 1st ACM international workshop on Vehicular ad hoc networks (VANET'04)*, 2004, pp. 47–56.

- [25] Z. D. Chen, H. Kung, D. Vlah, Ad hoc relay wireless networks over moving vehicles on highways, in: Proceedings of the 2nd ACM international symposium on Mobile Ad Hoc Networking & Computing (MobiHoc'01), 2001, pp. 247–250.
- [26] W. Yee, S. Navathe, E. Omiecinski, C. Jermaine, Efficient data allocation over multiple channels at broadcast servers, IEEE Transactions on Computers 51 (10) (2002) 1231–1236.
- [27] J. Zhao, Y. Zhang, G. Cao, Data pouring and buffering on the road: A new data dissemination paradigm for vehicular ad hoc networks, IEEE transaction on Vehicular Technology 56 (6) (2007) 3266–3277.
- [28] J. Zhao, G. Cao, VADD: Vehicle-assisted data delivery in vehicular ad hoc networks, IEEE Transactions on Vehicular Technology 57 (3) (2008) 1910–1922.
- [29] J. Chen, V. C. Lee, K. Liu, G. Ali, E. Chan, Efficient processing of requests with network coding in on-demand data broadcast environments, Information Sciences 232 (0) (2013) 27 – 43. doi:http://dx.doi.org/10.1016/j.ins.2012.12.044.
- [30] J. Chen, V. Lee, K. Liu, On the performance of real-time multi-item request scheduling in data broadcast environment, Journal of Systems and Software 83 (8) (2010) 1337–1345.
- [31] J. W. Wong, M. H. Ammar, Analysis of broadcast delivery in videotex system, Journal of IEEE Transactions on Computers 34 (9) (1985) 863–866.
- [32] J. W. Wong, Broadcast delivery, Proceedings of the IEEE 76 (12) (1988) 1566–1577.
- [33] P. Xuan, S. Sen, O. Gonzalez, J. Fernandez, K. Ramamritham, Efficient and timely dissemination of data in mobile environments, in: Proceedings of the 3rd IEEE Real Time Technology and Applications Symposium(RTAS'97), Montreal, Canada, 1997.
- [34] T. Mak, K. Laberteaux, 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), Cologne, Germany, 2005, pp. 1–9.
- [35] B. Hassanabadi, S. Valaei, Reliable periodic safety message broadcasting in vanets using network coding, IEEE Transactions on Wireless Communications 13 (3) (2014) 1284–1297. doi:10.1109/TWC.2014.010214.122008.
- [36] H. Schwetman, CSIM19: A powerful tool for building system models, in: Proceedings of the 33th IEEE Winter Simulation Conference, Arlington, VA, USA, 2001.
- [37] F. Bai, N. Sadagopan, A. Helmy, The IMPORTANT framework for analyzing the impact of Mobility on Performance Of Routing protocols for Adhoc Networks, Journal of Ad Hoc Networks 1 (2003) 383–403.
- [38] G. K. Zipf, Human Behaviour and the Principle of Least Effort: An Introduction to Human Ecology, Addison-Wesley, 1949.
- [39] K. Liu, V. Lee, On-demand broadcast for multiple-item requests in a multiple-channel environment, International Journal of Information Sciences 180 (22) (2010) 4336–4352.

Appendix A. Appendix

Appendix A.1. Cooperative Load Transferring

Here we describe our updated load transferring procedure that is used in ECLB. The load transferring procedure of CLB can be found in [12]. In our system, the movement of vehicles is assumed to follow the Manhattan mobility model [37]. Each RSU is assumed to periodically broadcast its workload status to its neighbor RSUs. The notations used in this section are given in Table A.6.

Before leaving the starting point a vehicle Vh_i estimates the route which it follows toward the destination using the onboard communication facility. There may be couple of possible routes towards the destination. The vehicle picks one according to the convenience and facilities. For example, in Fig. A.12 from the starting point RSU_1 to the destination point RSU_{15} , there are several routes available. Such as,

$$route_1 = \{RSU_2 \sim RSU_7, RSU_{15}\};$$

$$route_2 = \{RSU_9 \sim RSU_{15}\};$$

⋮

$$route_i = \{RSU_9 \sim RSU_{12}, RSU_8, RSU_3, RSU_4 \sim RSU_7, RSU_{15}\};$$

Figure A.12: RSUs in the cooperative model.

⋮

$route_n = \{RSU_2, RSU_3, RSU_8, RSU_{12} \sim RSU_{15}\};$

Assume that Vh_i selects $route_i$. Hence the neighboring RSUs set for vehicle Vh_i from RSU_i denoted by $Neighbor(RSU_i(route_i))$, is $\{RSU_9 \sim RSU_{12}, RSU_8, RSU_3, RSU_4 \sim RSU_7, RSU_{15}\}$. RSU_i needs to find out the target transferred $RSU_j \in Neighbor(RSU_i(route_i))$ by calculating their servicing capability so that RSU_j will not be overwhelmed by RSU_i 's transferred requests.

Without loss of generality, we consider that RSU_i wants to transfer request R_i (submitted by vehicle Vh_i) to RSU_j , where the installed location of RSU_j is within the valid range of remaining delay tolerance value of R_i at time t . Here RSU_i and RSU_j are known as the transferee and transferred RSU, respectively. Assume that the two RSUs are n_{hops} apart, where n_{hops} is a positive integer. Hence after leaving RSU_i , the total traveling time of the vehicle Vh_i from RSU_i to RSU_j including blind zone traveling and dwelling duration in the intermediate RSUs is,

$$T_{ij}^{travel} = n_{hops} \times T_{hop}^{travel} + (n_{hops} - 1) \times T^{Maxdwell}$$

Before making the transfer decision of R_i to RSU_j , RSU_i has to estimate the possible workload of RSU_j when vehicle Vh_i will arrive there. Assume that at RSU_j the request generating rate per vehicle is λ_j and the request serving rate is δ_j . The latest update about RSU_j 's workload status received by RSU_i is at time t_u . At t_u , RSU_j has $N_{pr}(RSU_j)$ requests to serve. Vehicle Vh_i starts its journey from RSU_i to RSU_j at time t (current time). Here, $\Delta t = t - t_u$.

To calculate the estimated total load at RSU_j after $(T_{ij}^{travel} + \Delta t)$, RSU_i needs to know the expected number of vehicles $N_{vehicle}(RSU_j)$ that may arrive (and generate requests) as well as requests transferred $N_{tr}(RSU_j)$ by this time at RSU_j . For calculating this value we use the historical data of RSU_j to estimate the number of vehicles $N_{vehicle}(RSU_j(\tau))$ at RSU_j in the next time window using the *exponential weighted moving average (EWMA)*,

$$N_{vehicle}(RSU_j(\tau)) = (1 - \beta) \cdot N_{vehicle}(RSU_j(prev)) + \beta \cdot N_{vehicle}(RSU_j(latest))$$

where $N_{vehicle}(RSU_j(latest))$ and $N_{vehicle}(RSU_j(prev))$ are the number of vehicles which have

Table A.6: Summary of notations.

NOTATION	DESCRIPTION
λ_j	Request generating rate at RSU_j
δ_j	Request serving rate at RSU_j
τ	Time window for EWMA
T_{ij}^{travel}	Required travel time from RSU_i to RSU_j
$N_{vehicle}$	Number of vehicles
N_{pr}	Number of unserved previous requests
N_{tr}	Number of transferred requests
N_{sr}	Number of served requests
N_{usr}	Number of unserved requests
N_{nr}	Number of newly arrived requests

just arrived at RSU_j in the latest and previous time window and β (set to 0.25) is a parameter to give priority to the *latest value* over the preceding ones.

Similarly, the estimated number of transferred requests at RSU_j in the next time window, $N_{tr}(RSU_j(\tau))$ is,

$$N_{tr}(RSU_j(\tau)) = (1 - \beta) \cdot N_{tr}(RSU_j(prev)) + \beta \cdot N_{tr}(RSU_j(latest))$$

Hence when the vehicle Vh_i arrives at RSU_j , the total number of vehicles that will generate requests at RSU_j , $N_{vehicle}(RSU_j)$ is,

$$N_{vehicle}(RSU_j) = \frac{N_{vehicle}(RSU_j(\tau))}{\tau} \times (T_{ij}^{travel} + \Delta t)$$

The total number of requests that will be transferred at RSU_j , $N_{tr}(RSU_j)$ is,

$$N_{tr}(RSU_j) = \frac{N_{tr}(RSU_j(\tau))}{\tau} \times (T_{ij}^{travel} + \Delta t)$$

Finally, the total number of requests will be served is $N_{sr}(RSU_j)$,

$$N_{sr}(RSU_j) = \delta_j \times (T_{ij}^{travel} + \Delta t)$$

Therefore, the total number of requests that still remain unserved at RSU_j including transferred requests R_i while Vh_i will just arrive there is,

$$\begin{aligned}
N_{usr}(RSU_j) &= 1(R_i) + N_{pr}(RSU_j) + N_{vehicle}(RSU_j) \cdot \lambda_j + N_{tr}(RSU_j) - N_{sr}(RSU_j) \\
&= 1 + N_{pr}(RSU_j) + (T_{ij}^{travel} + \Delta t) \times \left\{ \frac{N_{vehicle}(RSU_j(\tau)) \cdot \lambda_j}{\tau} + \right. \\
&\quad \left. \frac{N_{tr}(RSU_j(\tau))}{\tau} - \delta_j \right\}
\end{aligned} \tag{A.1}$$

A request R_i can have a maximum dwelling duration $T^{Maxdwell}(= \frac{2R}{V})$, hence R_i will be transferred to RSU_j , if and only if all unserved $N_{usr}(RSU_j)$ and new arrived requests $N_{nr}(RSU_j)$

(generated from RSU_j 's own vehicles and received transferred requests from its neighbor RSUs when Vh_i dwells at RSU_j) can be satisfied within time period $T^{Maxdwell}$ (this ensure that RSU_j will not be overwhelmed by transferred request R_i). That is,

$$\begin{aligned}\delta_j.T^{Maxdwell} &\geq \{N_{usr}(RSU_j) + N_{nr}(RSU_j)\} \\ &\geq \{N_{usr}(RSU_j) + \frac{2R}{V} \times (\frac{N_{vehicle}(RSU_j(\tau)).\lambda_j}{\tau} + \frac{N_{tr}(RSU_j(\tau))}{\tau})\} \quad (A.2)\end{aligned}$$

When request transfer is required for a vehicle, RSU_i performs the above load calculation to find the best suitable target transferred RSU among the neighbor RSUs in the selected route.