Volume 2 No. 2, February 2012 ISSN 2223-4985
International Journal of Information and Communication Technology Research

http://www.esjournals.org

An Approach for Alleviating the Starvation Problem in Road Side Units (RSUs)-based Vehicular Ad Hoc Networks (VANETs)

Syeda Khairunnesa Samantha, Nusrat Nur Afrose Shoma, K. M. Azharul Hasan
Department of Computer Science and Engineering
Khulna University of Engineering & Technology
Khulna, Bangladesh

ABSTRACT

Scheduling for data exchange in Vehicular Ad Hoc Networks (VANETs) is being considered as a vital issue by the researchers. Road Side Units (RSUs) being the standalone buffer point is to compensate the less connectivity problem due to mobility of vehicles. Here we formulate an RSU-based VANETs model. Due to vehicle mobility and RSU range constraint scheduling is an important issue to meet a vehicle's issued demand in hard real-time. We investigate existing scheduling algorithm serve a data item only based on vehicles' submitted requests' characteristics and avoiding the relationship between a vehicle and its submitted requests which leads a vehicle level starvation problem. In this paper, we address this starvation problem and propose a solution to minimize the problem. Simulation result also supports our proposed approach and offers expected result.

Keywords: *On-demand scheduling algorithms, Vehicular Ad Hoc Networks (VANETs), Road Side Units (RSUs)*

I. INTRODUCTION

A number of applications (road safety, internet access, entertainment etc.) have been envisioned in Vehicular Ad Hoc Networks (VANETs) [16], [22], [23]. Efficient data dissemination mechanism is a key challenge to provide successful VANETs applications. In VANETs, usually vehicles move pretty fast leading to short vehicle to vehicle connectivity time; moreover in the case of VANET roll out phase (when vehicles density is low, night time/off-peak hour, highways etc.), there is very little chance to get the required information from other vehicles. Hence, installing RSU at the important places in a planned way [6] and get responses from it is an important consideration in this environment.

RSU is a stationary substance unit having wireless access point (Dedicated Short Range Communication (DSRC) [15]), memory storage and computational capabilities. As RSU transmission range is short and vehicles are usually on the move, hence duration between request submission and getting response from RSU is a key consideration for a successful VANETs data dissemination. To achieve better performance in this circumstance, an RSU needs to provide services to the vehicles so that it can achieve minimum deadline miss rate, high throughput and minimum response time.

Broadcasting is an efficient approach in this environment as many vehicles' requests can be served by a single broadcast. Broadcasting can be done in two ways: 1) Periodic broadcasting and 2) On-demand broadcasting. Periodic broadcasting is not scalable for handling large database [20], [21]. Moreover, client access patterns are not same all the time. So, on-demand broadcasting is more suitable than periodic broadcasting in VANETs.

In spite of the development and maintenance cost VANETs has its advantages to the drivers. With rapid growth of real-time services and business oriented applications such as traffic conditions, stock quotes, internet access, road safety [5], [22]; information must reach users within strict time period to be useful and meet the user demand.

In VANETs a Road-side Unit (RSU) may get some requests that have less popularity comparatively to others. But most of the scheduling algorithms work based on either popularity (MRF, R×W and DSIN) or requests' assigned deadline (EDF). So those less popular requests have high probability to miss the deadline as a consequence these less popular or unpopular requests submitting vehicles don't get their desired information which is called *vehicle level starvation problem.* In this paper, we dig dipper into this kind of problem. Again, if an RSU serves too many of such unpopular requests, throughput decreases. Therefore, we concentrate on formulating a scheduling procedure which would be the tradeoff of minimizing the starvation problem and maximizing the VANETs throughput. If we deliberately handle throughput and starvation problem, more requests will be satisfied and in turns more vehicles will get their desired information.

The rest of the paper organized as follows. Section II describes related work; section I shows our VANETs system model; section 0 describes existing on-demand scheduling algorithms; section V elaborates the problem statement; section VI and VII show our proposed solution and its complexity respectively. We describe our adopted performance metrics, simulation model and experimental results in section VIII, IX and 0 respectively. We finish by a discussion and stating our future work in section XI.

II. RELATED WORK

Many efforts have been carried out to find an efficient data dissemination procedure. Due to the high mobility of vehicles which is a unique characteristic of

VANETs, many researchers try to adopt different techniques for finding a stable data dissemination procedure. Chen et al. [1] propose messages relayed technique where data is stored at the moving vehicles until favorable data delivering opportunities come. MDDV [2] also uses the intermediate nodes to buffer the data and carry it until any of the forwarding approaches (opportunistic, trajectory based and geographical forwarding) is encountered by the environment. VADD [3] uses the store and forward procedure and to reduce the data delivery delay it considers predictable traffic pattern and road layout. In DP-IB [4] technique data are periodically broadcasted to vehicles; vehicles buffer that data and rebroadcast it at the intersections. T. Nadeem et al. [5] propose periodic broadcast approach to disseminate both generated and relayed data. Lochert et al. [6] recommend few wired connected RSUs to provide better data dissemination than many standalone RSUs. Zhang et al. [15] propose a Two-step scheduling to balance the upload and download services. Yi et al. [24] study the scheduling issue in the mesh RSU environments.

[7], [8], [11] Study the classical broadcasting system. [12], [13] Research on data dissemination in asymmetric communication environment. Some researches focus on the on-demand scheduling for information dissemination [10], [14]. Some researchers focus on the scheduling issue in real-time system [9], [20], [21].

However none of the above works consider vehicle level starvation problem in VANETs.

III. SYSTEM MODEL

A. System Architecture

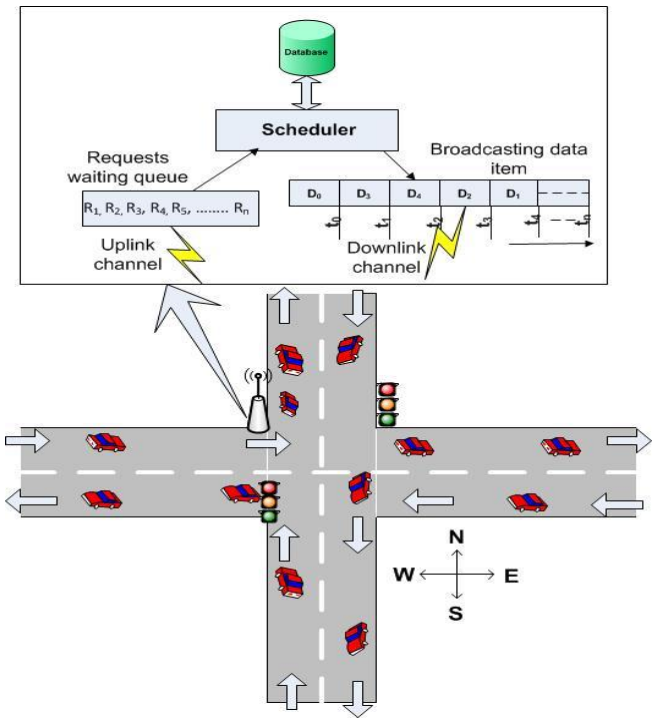

Figure 1. System architecture

Our system model is similar to Figure 1. We assume an RSU server maintains the updated data item by system administrator. When a vehicle is in the transmission range of an RSU, it can generate requests for the updated data item through the uplink channel and sense the downlink channel until satisfied or leaves the RSU transmission range. Generated requests will be queued in the RSU waiting queue for being served. The scheduler selects a suitable request among the queued requests in the waiting queue following the used scheduling algorithm principle. The data item requested by the selected request then broadcast through the downlink channel. All the vehicles requested for the broadcasted data item satisfied concurrently. A vehicle can send requests and get responses until it passes the transmission range of an RSU.

B. Notations and Assumptions

Assume An RSU transmission range is D. When a vehicle Vh_i submits a request RQ_i to an RSU, it submits the following information with that request RQ_i:

$RQ_i = \{Vh_i, DID_i, T^{deadline}\}$, where, DID_i is the data item requested by request RQ_i and $T^{deadline}$ is the time the request RQ_i will be valid. A request RQ_i will be valid only vehicle Vh_i is in the RSU transmission range.

Vehicle Vh_i can calculate this period also called deadline from vehicle driving speed and RSU transmission range D. If a request RQ_i hasn't served before vehicle left the RSU transmission range, RQ_i will be dropped from the RSU waiting queue.

Before going to the detailed approach, here we give some definitions:

Definition 1: The un-ordered request set in the RSU waiting queue at time t is, $REQ(t) = \{RQ_1(Vh_i), RQ_2(Vh_j), \cdots, RQ_n(Vh_k) \cdots\}$, which means the request RQ_1, RQ_2 *and* RQ_n are submitted by vehicle Vh_i, Vh_j *and* Vh_k respectively and so on. $Vh(t)$ denotes the vehicles set whose submitted requests are in the RSU waiting queue at time t.

Definition 2: An RSU maintains a database of the data items the vehicles request. $DBSize$ denotes the total RSU database size, $Requested(DID(t))$ denotes the set of data items requested by unserved requests at time t and $DSize(DID_i)$ denotes the size of the data item DID_i. The popularity of a data item DID_i at time t means the total number of unserved requests in the RSU waiting queue which ask for the data item DID_i and it is denoted by $Popu(DID_i(t))$. If at time t, DID_i is requested by n number unserved requests, $Popu(DID_i(t)) = n$. For every subsequent one more submitted request for DID_i, $Popu(DID_i(t))$ is increased by 1. If DID_i is broadcast by RSU at time t', at that moment there will be no unserved request waiting for DID_i as a consequence $Popu(DID_i(t')) = 0$.

Definition 3: If a vehicle average speed in the RSU range is S, calculative deadline of a request RQ_i is:

$$T_{cal}^{deadline}(RQ_i) = \frac{2D}{S} - T^{gnr}.$$

where T^{gnr} is the request generation time in the RSU service range. As there is a chance that a vehicle may stop or change its driving speed while it is in the RSU transmission range, the general assigned deadline of a request RQ_i is:

$$T^{deadline}(RQ_i) = \min\left\{\left(random(\omega_{min}, \omega_{max}) * T^{ServiceTime}(RQ_i)\right), T_{cal}^{deadline}(RQ_i)\right\}.$$

where, ω is a random number and

$$T^{ServiceTime}(RQ_i) = \frac{DID_i}{CBW}$$

is the time needed to serve requested data item DID_i by an RSU having CBW channel bandwidth.

As long as a request RQ_i is not served or its deadline $T^{deadline}(RQ_i)$ doesn't expire it resides in the RSU waiting queue and considered as unserved request RQ_i. After serving or expiring deadline the request RQ_i will be discarded from waiting queue.

IV. EXISTING ON- DEMAND SCHEDULING ALGORITHMS

To select a request from the RSU waiting queue requests for serving in the next service cycle, an RSU needs to use an on-demand scheduling algorithm. Followings are some widely used on-demand scheduling algorithms.

1. First Come First Served (FCFS): This is a base line scheduling algorithm. It serves the requests according to their arrival order. We just use this to take into consideration the performance of other different on-demand algorithms; how far they vary from the base line scheduler.

2. Most Request First (MRF): This algorithm works according to the popularity of the data item. It broadcasts the data item from the database which has the maximum popularity. At time t data item DID_i will be selected where:

$$DID_i = max\{Popu(DID_i) | \forall DID_i \in Requested(DID(t)\}.$$

3. Earliest Deadline First (EDF): EDF works according to the deadline of the requests. The data item which is requested by the request which has most urgent deadline among the unserved request set will be served first. At time t request RQ_i will be selected for satisfying where:

$$RQ_i = min\{T^{deadline}(RQ_i) | \forall RQ_i \in REQ(t)\}.$$

4. Number of pending requests Multiply Waiting time (R×W): R×W works based on the two factors. R means the number of outstanding requests waiting for a data item also called popularity of the data item and W means waiting time of the submitted request. So, before making serving decision, this algorithm calculates every request's R×W value, i.e. the multiplication of popularity of the requested data item and waiting time of the request. The request having maximum R×W value will be broadcast first. It selects a request RQ_i for satisfying at time t:

$$RQ_i = max\{R \times W(RQ_i) | \forall RQ_i \in REQ(t)\},$$

where, $R(RQ_i) = Popu(DID_i(t))$ *and* $W(RQ_i) = t - T^{gnr}(RQ_i)$.

5. Longest Wait First (LWF): LWF measures the sum of the waiting time of all the outstanding requests for a data item. A data item with maximum LWF value will be chosen for broadcasting. LWF incorporates directly request's deadline and indirectly popularity of the requested data item. It serves a data item DID_i at time t:

$$DID_i = \max\{LWF(DID_i) | \forall DID_i \in Requested(DID(t)\},$$

where

$$LWF\ (DID_i) = \sum_{\forall RQ_j \in REQ(t)\ requests\ DID_i} \left(t - T^{gnr}(RQ_j)\right).$$

6. Shortest Service Time First (SSTF): Yu et al. [14] study the performance of SSTF algorithm in heterogeneous environment. SSTF picks out the data item from the requested data item according to their service time. The data item which needs minimum service time to serve will be broadcasted first, where *service time is the time necessitated to broadcast a data item when the system is idle*. SSTF directly depends on the data item size. At time t, data item DID_i will be selected for serving:

$$DID_i = min\{DSize(DID_i) | \forall DID_i \in Requested(DID(t)\}.$$

7. Deadline Size Inverse Number of pending requests (DSIN): DSIN algorithm is used in [15] where they call it as D*S/N. DSIN combines deadline of the request, size and popularity of the requested data item. Before making broadcasting decision scheduler determines the DSIN value of the all the requests in the waiting queue and serves the request which has minimum DSIN value. A request RQ_i gets priority for satisfying at time t, where:

$$RQ_i = min\{DSIN(RQ_i) | \forall RQ_i \in REQ(t)\},$$

where $DSIN(RQ_i) = \frac{(T^{deadline}(RQ_i) \times DSize(DID_i)}{Popu(DID_i(t))}$).

V. PROBLEM STATEMENT

A scheduler usually serves a request based on the RSU waiting queue request characteristic (request deadline) or requested data item characteristics (popularity and size). Such as if we see our above discussed existing on-demand scheduling algorithm, FCFS and EDF are request deadline based, MRF is data item popularity based, SSTF is data item size based, $R \times W$ and LWF are based on data item popularity and request deadline, DSIN is based on request deadline along with data item size and popularity. As requests are selected based on the requests and their requested data item characteristics irrespective of the relationship of request and its submitted vehicles, a vehicle submitting requests having critical deadline or generating requests for popular data item will always be served than the vehicle submitting requests for non-popular data item or generating requests having soft deadline. When this situation continues for long time and the later kind vehicles can't get serviced, we call this problem as *vehicle level starvation problem.* To clearly understand this problem, let us have the following simple example.

At time t at the RSU waiting queue we have the following requests set having n number requests and their particulars:

$$Req = \{RQ_1(Vh_1), RQ_2(Vh_2), RQ_3(Vh_1), RQ_4(Vh_2)\}$$

$$T^{deadline}(RQ_1) = 10;\ T^{deadline}(RQ_2) = 20;$$
$$T^{deadline}(RQ_3) = 15;\ T^{deadline}(RQ_4) = 35;$$

$$Popu(DID_1(RQ_1)) = 5;\ Popu(DID_5(RQ_2)) = 1;$$
$$Popu(DID_3(RQ_3)) = 10;\ Popu(DID_{100}(RQ_2)) = 3;$$
$$DSize(DID_1) = 10;\ DSize(DID_5) = 125;$$
$$DSize(DID_3) = 36;\ DSize(DID_{100}) = 56;$$

Using the above metrics value every existing on-demand scheduling algorithm (defined above) selects request $RQ_1 and\ RQ_2$ before $RQ_3\ and\ RQ_4$. As $RQ_1 and\ RQ_3$ are issued by vehicle Vh_1 and $RQ_2\ and\ RQ_4$ are by Vh_2, hence we can conclude from this example that Vh_1 gets priority over Vh_2 for request serving and Vh_2 only can use the RSU response channel after Vh_1 finishes. Now if by this time Vh_2 leaves the RSU transmission range, none of its submitted requests will be successful. In this case, we call Vh_2 is suffering from starvation problem. To alleviate this *vehicle level starvation* problem we propose a solution which considers the characteristics of submitted request, requested data item and the relationship between a request and its submitted vehicle.

VI. THE PROPOSED SOLUTION

The simplest possible solution to remove *vehicle level starvation* problem is to serve the waiting queue requests according to vehicle identity in a circular fashion. In this type of Round Robin (RR) service cycle, the vehicle having the lowest identity number is served in the first service cycle and the one having highest identity number served in the last service cycle. This approach continues until the RSU waiting queue in not empty. However, this simple RR approach only considers the vehicle characteristics and avoids its submitted request characteristics. We propose to integrate the cycling RR approach to DSIN scheduling while scheduling a request to reflect the characteristics of vehicle, its submitted request and requested data item. We call this approach as RRDSIN approach which also considers the relationship between vehicle and its requests. Algorithm 1 shows the pseudo code of our proposed RRDSIN approach.

Algorithm 1. RRDSIN Algorithm

1. **Require:** Queue **REQ** holds the requests submitted by vehicles
2. **Require:** $MinPriority \leftarrow 1$
3. **Require:** $\forall Vh_i \in Vh(t), PriorityStatus[Vh_i] \leftarrow MinPriority$ /* Initialize all vehicles' priority to MinPriority 1*/
4. **Ensure:** Vehicle level starvation problem is minimized
5. **while** $REQ \neq \phi$ ***do***

6. **if** $PriorityStatus[Vh_i] \leq MinPriority$ **then**
7. $DSIN(RQ_i(Vh_i)) = \frac{(T^{deadline}(RQ_i) \times DSize(DID_i)}{Popu(DID_i(t))})$
8. Find $RQ_i(Vh_i) \in REQ$ *having minimum* $DSIN(RQ_i(Vh_i))$
9. $PriorityStatus[Vh_i]$ increment by 1 /*Vh_i will no longer will be considered for serving until all the vehicles $\in Vh(t)$ served*/
10. Serve request $RQ_i(Vh_i)$
11. **end if**

12. **if** $\forall Vh_i \in Vh(t), PriorityStatus[Vh_i] > MinPriority$ **then** /*Ensures requests submitted all the vehicles served (end of a cycle) */
13. MinPriority increment by 1
14. **end if**
15. **end while** /*Line 6 to 14 continue until all the vehicles pass the RSU transmission range */

RRDSIN approach selects a request having minimum DSIN value among the requests submitted by those vehicles which yet not served in the current serving round (Line 6 - 10). If a vehicle submitted request is served, it will not be considered for serving in the current round. When all the vehicles in the current round served once, the next round start and RRDSIN will consider all the vehicles' submitted requests to choose one having minimum DSIN value (Line 12-14). An RSU repeats this process until all the vehicles passes the RSU range or waiting queue will be empty.

VII. COMPUTATION COMPLEXITY

We estimate the computation complexity of our proposed RRDSIN algorithm from the number of unserved requests is examined before making a serving decision. Without loss of generality, we assume at time t, there are n number vehicles in the RSU service range and on the average their number of submitted requests is k. Hence before making serving decision scheduler needs to check total $N = k \times n$, number requests, hence the computation complexity for all the existing on-demand scheduling algorithms defined above is $O(N)$. RRDSIN approach in the circular first service cycle examines N number requests but in the second service cycle examines $N - k$ number requests. Similarly in the i^{th} service cycle RRDSIN examines $N - i \times K$ and in the last service cycle in that circular round examines only $N - (n-1) \times k$ number requests. Hence on the average, in a circular service cycle RRDSIN examines $N - \frac{k(n-1)}{2}$ number requests. Clearly for $k > 0$ *and* $N > 1$ RRDSIN reduces the number of examined requests in a service cycle than any of the above defined scheduling algorithms. Hence average complexity of RRDSIN is $O(N - \frac{k(n-1)}{2}) \equiv O(N)$.

VIII. PERFORMANCE METRICS

We use the following performance metrics to evaluate the performance differences of our proposed RRDSIN approach with different existing on-demand scheduling algorithms.

1. Deadline Miss Rate: It measures the percentage of number of requests missed the deadline to the total number of requests received by the RSU. If the deadline miss rate is low, means scheduling algorithm is better.

2. Throughput: Throughput is the number of requests successfully served by an RSU in unit time. Hence, if a scheduler broadcasts the most popular data item, many requests will be served concurrently and throughput increased. High throughput means better system performance.

3. Average Response Time: The average amount of time required to get the response from an RSU after submission of a request. Low average response time initiates system is better.

4. Satisfied Vehicles Ratio: is the ratio of number of vehicles (which generate requests) satisfied to the total number of vehicles generate requests in the RSU service range. This metric measures the alleviation of the vehicle level starvation problem. If a scheduling algorithm achieves higher degree of satisfied vehicles ratio, it has higher capability to mitigate the vehicle level starvation problem.

IX. SIMULATION MODEL

Our simulation model is similar to our system architecture shown in Figure 1. We use CSIM19 [17] for our simulation experience and the explicit used parameters are shown in Table 1, other parameters are CSIM default.

At a time only one data item will be served by the RSU and servicing is non-preempted. RSU range is limited i.e. the approximated radius it can serve has a limit considering the fact of wireless broadcasting strength. A vehicle can continuously generate requests after entering into the transmission range of an RSU till moving out irrespective of its previous requests successful or not. At a time a vehicle can send single item request and the vehicle request generation interval is exponentially distributed defined by IATM (Table 1). If IATM value is low, request generation interval period is short so heavy load comes to the RSU. Vehicles data item access pattern is distributed by

Zipf [18] distribution. Here, the access probability of j^{th} data item is:

$$p(j) = \frac{\frac{i}{j^{\theta}}}{\sum_{n=1}^{N} \frac{1}{n^{\theta}}}$$

Where 0≤ θ ≤1, 0 means uniform and 1 means strict Zipf distribution.

For generating RSU database, we use increment data item size distribution (INCRT) [19], [20]. In the combined Zipf data item access pattern and INCRT size data item distribution, vehicles requested popular data item will be smaller sized and unpopular data item will be bigger sized.

INCRT size distribution is:

$$DSize_i = DSMin + \frac{(i-1) \times (DSMax - DSMin + 1)}{DBSize}$$

where i = 1,2,3, ……..,DBSize.

TABLE I. SIMULATION PARAMETERS

Parameter	Default	Range	Description
IATM	0.3	0.1-1.0	Request generation interval
NVehicle	100	25-200	Number of vehicles
THETA	0.7	0.0-1.0	Zipf distribution parameter
DBSize	500	--	Number of data items in the database
CBW	100KB/sec	--	Broadcasting bandwidth
Commun Range	350 m	--	RSU communication range
$\omega_{min}, \omega_{max}$	5, 10	--	Range of min. and max. random number
DSMin, DSMax	15, 512 KB	--	Min. and max. size of data item

X. PERFORMANCE EVALUATION

To mimic the real time traffic, we generate vehicles from one side of a road and then let the vehicles to enter and go forth the RSU transmission range. At a time the maximum number of vehicles that may generate requests is defined by the value NVehicle (TABLE I). We continue our simulation until we receive stable simulation data and 95% confidence interval achieved. In the following portion we discuss our experimental outcomes.

A. Performance Analysis for Varying RSU Workload

We analyze the effect of varying workload in the RSU service range of all the existing on-demand scheduling algorithms including the naive RR and our proposed RRDSIN algorithms in context of our defined performance metrics. We see the performance variation in the same parameters settings in terms of deadline miss rate (Figure 2(A)), average response time (Figure 2(B)), throughput (Figure 3(C)), and Fairness Ratio (Figure 3(D)) by increasing number of vehicles in the RSU transmission range.

Deadline Miss Rate: By increasing workload deadline miss rate increases for all the algorithms. When number of vehicles increases, number of requests generation also increased, as a consequences an RSU gets many requests in the waiting queue. Then while RSU servicing a request many requests may miss their deadlines during that time period, hence overall deadline miss rate increased. From Figure 2(A), EDF, RR and FCFS suffer worst when number of vehicles increased. This is because EDF only consider the deadline of the requests and neglects the size of the data item, so while serves a big size data item of an urgent deadline request, it takes long time to server, hence during that time many others urgent requests miss their deadline. FCFS does not consider either deadline or data item size so it also suffers worst.

As here we use INCRT size distribution, small sized data are most popular, so considering size of the data item is an important metric for decreasing deadline miss rate. SSTF considers data item size, hence it has moderate deadline miss rate. R×W and MRF both use popularity and their performance almost same. Although LWF does not consider data item size, its indirect popularity measure helps to improve the deadline miss rate.

DSIN which considers deadline, size and popularity outperforms all the other algorithms. Its combination of data item size and deadline metrics help to achieve better deadline miss rate in INCRT size distribution. However, RRDSIN gives a moderate result for varying workloads. When workload is low, it performs as well as DSIN, but with increasing workload its performance degrades a bit, still it is better than other naive schemes.

Average Response Time: Except MRF and R×W algorithms, there is no significant change for average response time with increasing workload. From Figure 2(B) with increasing workload MRF and R×W get more popular smaller sized data item for broadcasting at θ value 0.7 and INCRT size distribution, hence MRF and R×W average response time decreases with increasing workload. All other algorithms' average response time value remain almost same except FCFS and EDF, their value slightly rise from the initial stage with high workload because they do not get the advantage either from popular or smaller sized data item. Although with increasing workload waiting queue increases for DSIN, SSTF and LWF, they get the advantage for disseminating more popular sized smaller data item among the many waiting requests, so they can retain their average response time value stable. However, DSIN can maintain the stable lowest average response time in high workload condition. As RRDSIN works based on both the DSIN and RR characteristics, although it can't achieve response time value as good as DSIN, it just lays

behind DSIN which is better than a number of naive schemes.

Throughput: All algorithms' but FCFS, RR and EDF's throughput significantly increases with increasing workload (Figure 3(A)). We do this experiment by setting Zipf distribution parameter at default 0.7. Hence, with increasing number of requests generation, many requests ask for the same data item, then by disseminating that popular data item throughput increases dynamically.

However, FCFS, RR and EDF do not consider the popularity, so they have not much significant improvement in throughput for increasing workload. By disseminating smallest size data and long delayed requests for popular data item SSTF and LWF achieve better throughput respectively. But DSIN achieves best throughput among all with increasing workload for disseminating smallest sized popular data item. RRDSIN also has a moderate increase in throughput when workload is increased.

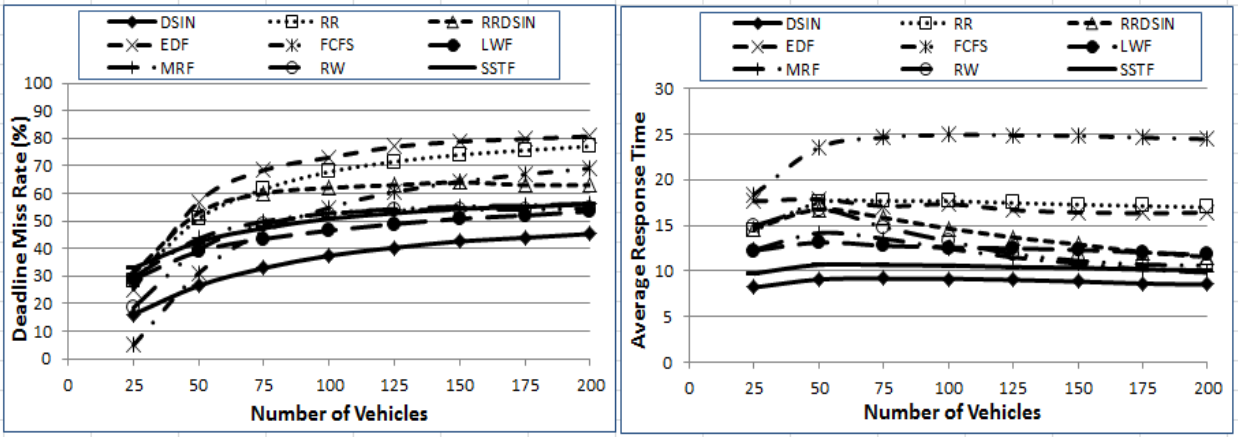

Figure 2. Impact of varying RSU workload: (A) Deadline Miss Rate (B) Average Response Time.

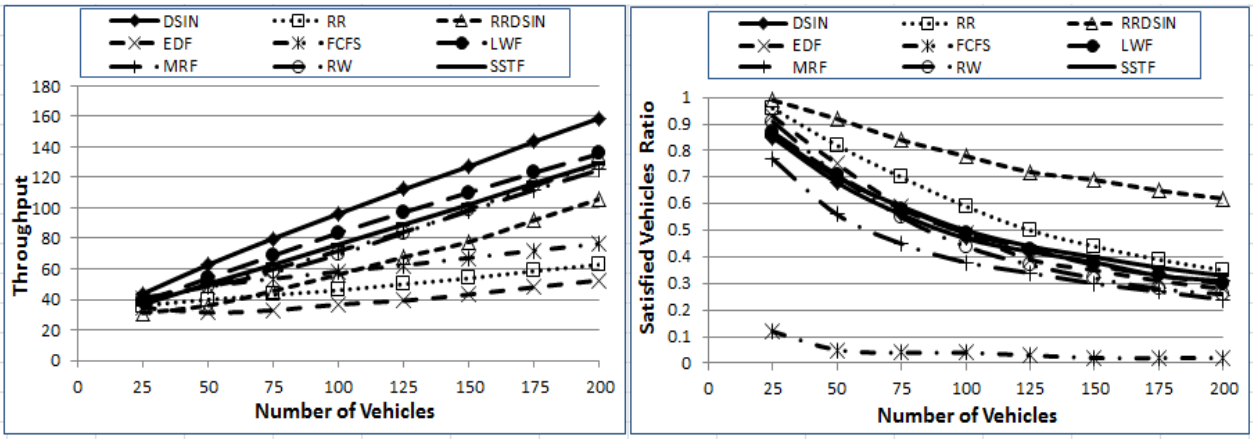

Figure 3. Impact of varying RSU workload. (A) Throughput (B) Satisfied Vehicles Ratio.

Satisfied Vehicles Ratio: In fixed size number of request per client, except FCFS all other algorithms give fair service to all vehicles when the workload is much lower. With increasing workload MRF, R×W, EDF, DSIN, SSTF and LWF get the advantage for disseminating more popular sized smaller data item among the many waiting requests. However problem arises here that with increasing workload considering these factors all algorithms except RR and RRDSIN might yield in giving service to some particular vehicles always requesting for urgent or popular data items or perhaps small sized data. As a result satisfied vehicles ratio seriously degrades in these cases. But RR considers fair service to vehicles levels and results in a better satisfied vehicles ratio according to the Figure 3(B). Finally RRDSIN, which considers all the important factors of the algorithm DSIN as well as optimizes the starvation problem, outperforms all other algorithms when it comes to providing fair service to all the vehicles in an RSU range.

B. Impact of Vehicles Data Access Pattern

Here we analyze the performance variation of our proposed RRDSIN algorithm from RR and other existing on-demand scheduling algorithms in the effect of varying the vehicles data item access pattern (Zipf distribution parameter θ) from 0.0 to 1.0. Figure 4(A) and (B) depict performance variation regarding deadline miss rate and average response time respectively. Figure 5(A) and (B) show the performance regarding throughput and satisfied vehicles ratio respectively.

Deadline Miss Rate: In Figure 4(A) when θ is 0, vehicle data access pattern is purely random distribution, so all the algorithms have high deadline miss rate. But with increasing θ value, vehicles request popular smaller sized data item. Then by a single broadcast many requests been

served with short time, hence performance increased dramatically.

MRF and R×W shows modest performance increases with increasing θ value for their popularity metric. However, here also DSIN algorithm performs better than all others for its combined request selection criteria (especially popularity and deadline influence here much). FCFS and EDF however fails avoid higher deadline miss rate with increasing value of θ.

RRDSIN achieves decent performance regarding deadline miss rate for varying θ value. However its performance stands behind DSIN or other popularity based algorithms, because when access pattern is much skewed for the tendency of RRDSIN to ensure fair service, it fails to serve most popular data item as its first priority is to serve the other vehicles in its service cycle even if they want unpopular data.

Average Response Time: Observing Figure 4(B) EDF and FCFS have no significant improvement for decreasing average response time with increasing θ, because they do not consider data item size or popularity which effect a lot for decreasing average response time especially in INCRT size distribution.

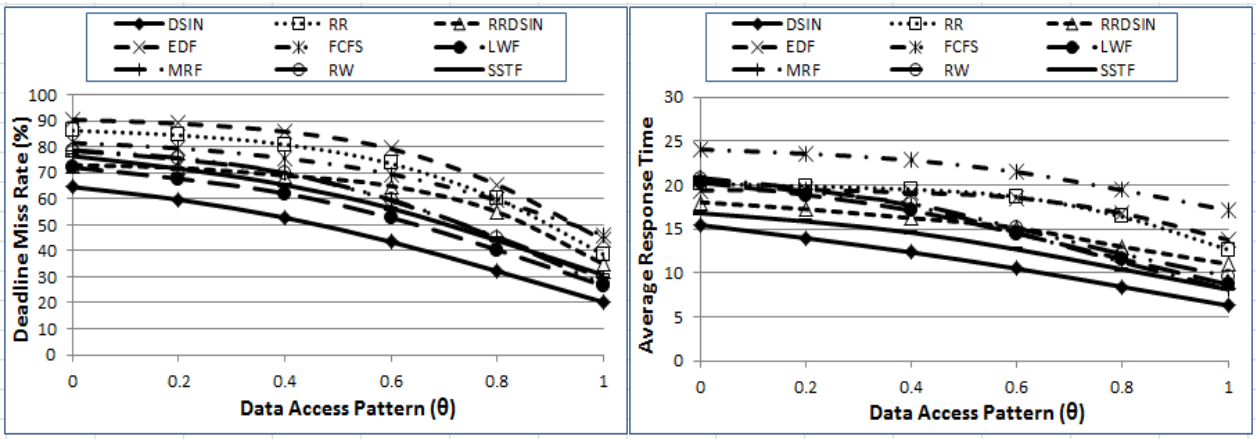

Figure 4. Impact of vehicles data access pattern: (A) Deadline Miss Rate, (B) Average Response Time

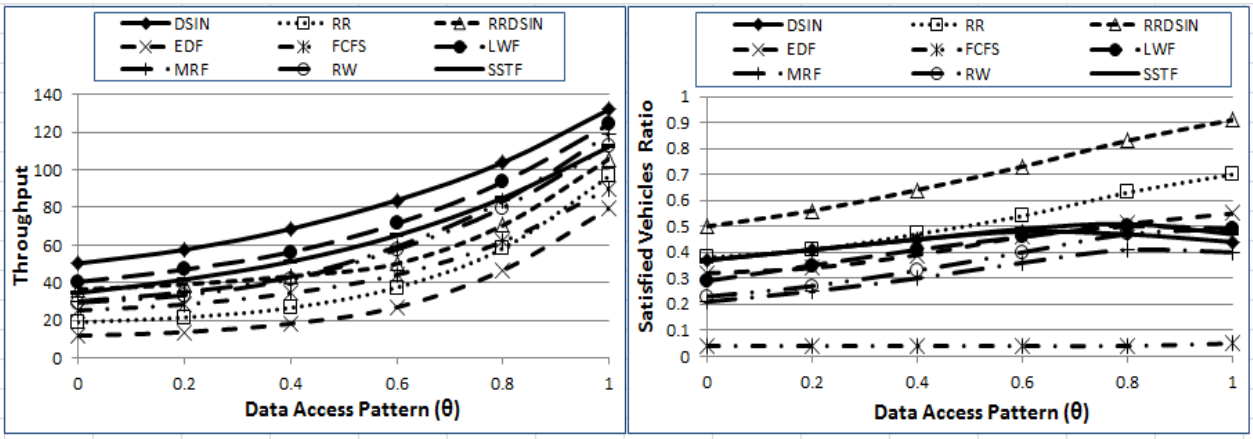

Figure 5. Impact of vehicles data access pattern: (A) Throughput, (B) Satisfied Vehicles Ratio

As MRF and R×W counts the popularity for requests selection, by broadcasting popular data item (which are smaller sized too) they can reduce the average response time with increasing θ value. But DSIN achieves the best average response time value by broadcasting popular smaller sized data item. Naive RR considering only the number of time vehicles got service does not perform well for varying θ value. For RRDSIN, with increasing value of θ it gives moderate performance considering popular small sized data and service count of vehicles.

Throughput: As increasing θ value in Figure 5(A) many requests ask for the same popular data item, by servicing such hot data items scheduler throughput increases significantly. By broadcasting deadline urgent and popular date item DSIN algorithm outperforms all other on-demand algorithms in terms of throughput. RRDSIN presents a moderate throughput as it can balance between popularity as well as vehicle level fairness.

Satisfied Vehicles Ratio: Shown in Figure 5(B), FCFS performs worst throughout the increasing value of θ as it does not consider fair service to vehicles at all and rather a blunt scheme. As MRF and R×W counts the popularity for requests selection, by broadcasting popular data item they however may continue to serve some vehicles with popular data leaving those vehicles requesting for colder data. DSIN achieves the moderate satisfied vehicles ratio value by broadcasting popular small sized data item. Naive RR

considering only the number of time vehicles got service performs well for varying θ value in this case. SSTF considering small sized data and EDF taking care of earliest deadline faces similar starvation problem serving urgent deadline and smaller data and leaving vehicles craving for larger data size or larger deadline. However, for RRDSIN, with increasing value of θ it gives best performance considering popular smaller sized data and service count of vehicles.

XI. DISCUSSIONS AND FUTURE WORKS

In this paper, we study the *vehicle level starvation* problem in RSU-based VANETs which occurred when an RSU use an existing on-demand scheduling algorithm. We demonstrate that this problem arise because of avoiding the relationship between vehicles and its submitted requests. Considering this relationship along with the characteristics of both the submitted request and its request data item, we formulize RRDSIN scheduling algorithm. Simulation results show that our propose algorithm can achieve moderate deadline miss rate, response time and throughput performance metrics value, however in term of alleviating vehicle level starvation problem it outperforms all the existing algorithms.

In the future work, we want to apply RRDSIN scheduling algorithm in the multiple RSUs based VANETs environment.

REFERENCES

[1] Z. D. Chen, HT Kung and D. Vlah, "Ad Hoc Relay Wireless Networks over Moving Vehicles on Highways,"Proc. 2nd ACM international symposium on Mobile ad hoc networking and computing, 2001.

[2] H. Wu, R. Fujimoto, R. Guensler and M. Hunter, "MDDV: A mobility-centric data dissemination algorithm for vehicular networks," Proc. ACM VANET, 2004.

[3] J. Zhao and G. Cao, "VADD: Vehicle-Assisted Data Delivery in Vehicular Ad Hoc Networks," IEEE Transaction on Vehicular Technology, vol. 57(3), May 2008.

[4] J. Zhao, Y. Zhang and G. Cao, "Data Pouring and Buffering on the Road: A New Data Dissemination Paradigm for Vehicular Ad Hoc Networks," IEEE transaction on Vehicular Technology, Nov. , 2007.

[5] T. Nadeem, P. Shankar and L. Iftode, A Comparative Study of Data Dissemination Models for VANETs," Proc. ACM VANET, 2006.

[6] C. Lochert, B. Scheuermann, M. Caliskan and M. Mauve, "The Feasibility of Information Dissemination Vehicular Ad-Hoc Networks," Proc. 4th Annual Conference on Wireless On-demand Network Systems and Services (WONS), 2007.

[7] J. W. Wong and M. H. Ammar, "Analysis of Broadcast delivery in Videotex System," IEEE Transactions on Computers, vol. C34(9), Sept. 1985.

[8] J. W. Wong, "Broadcast Delivery," Proceedings of the IEEE, vol. 76(12), 1988, pp. 1566-1577.

[9] C. Liu and J. Layland, "Scheduling Algorithms for Multiprogramming in Hard Real-time Traffic Environments," ACM, vol. 20(1), 1973, pp. 179-194.

[10] D. Aksoy and M. Franklin, "R×W: A Scheduling Approach for Large-Scale On-demand Data Broadcast," IEEE/ACM Transactions on Networking, Vol. 7(6), 1999, pp. 846-860.

[11] H. Dykeman and J. Wong, "A Performance Study of Broadcast Information Delevery Systems," INFOCOM, 1988, pp. 739-745.

[12] J. Fernandez and K. Ramamritham, "Adaptive Dissemination of Data in Time-Critical Asymmetric Communication Environments", MobNets'04.

[13] N. H. Vaidya and S. Hameed, "Scheduling Data Broadcast in Asymmetric Communication Environments", WirelessNets'09.

[14] Y. Wu and G. Cao, "Stretch-Optimal Scheduling for On-Demand Data Broadcasts," ICCCN, 2001.

[15] Y. Zhang, J. Zhao and G. Cao, "On Scheduling Vehicle-Roadside Data Access," Proc. ACM VANET, 2007.

[16] M. D. Dikaiakos, A. Florides, T. Nadeem and L. Iftode, "Location-aware Services over Vehicular Ad-Hoc Networks Using Car-to-car Communication", IEEE Journal on selected areas in Communications, vol. 25(8), 2007.

[17] H. Schwetman, "CSIM19: A powerful tool for building system models," Proc. 33th IEEE Winter Simulation Conference, Arlington, VA, USA, 2001, pp. 250-255.

[18] G. Zipf, "Human Behaviour and the Principle of Least Effort: An Introduction to Human Ecology," Addison-Wesley, Cambridge, MA, 1949.

[19] J. Xu, Q. Hu, WC. Lee and D. L. Lee, "Performance Evaluation of an Optimal Cache Replacement Policy for Wireless Data Dissemination," IEEE Transactions on Knowledge and Data Engineering, vol. 16(1), Jan. 2004, pp. 125-139.

[20] V.C.S. Lee, X. Wu and J. K. Ng, "Scheduling Real-time Requests in On-demand Data Broadcast Environments,"Real-Time System, Vol. 34(2), Oct. 2006, pp. 83-99.

[21] J. Chen, K. Liu and V. C.S. Lee, "Analysis of Data Scheduling Algorithms in Supporting Real-Time Multi-item Requests in On-demand Broadcast Environments," IPDPS, 2009.

[22] E. Schoch, F. Kargl, M. Weber and T. Leimüller, "Communication Patterns in VANETs," IEEE Communication Magazine, November, 2008.

[23] A. Boukerche, H. A.B.F. Olivera, E. F. Nakamura and A. A.F. Laureiro, "Vehicular Ad Hoc Networks: A New Challenge for Localized-Based Systems," Computer Communications, vol. 31, 2008, pp. 2838-2849.

[24] L. Z. Yi, L. Bin, Z. Tong and Y. Wei, "On Scheduling of Data Dissemination in Vehicular Networks with Mesh Backhaul," Proc. IEEE ICC, 2008.