A Novel Approach on Striking the Balance between the Fairness and Throughput in Vehicular Networks


∗School of Electrical & Electronic Engineering, Nanyang Technological University, Singapore
∗Department of Computer Science and Engineering, Khulna University of Engineering & Technology, Bangladesh
∥Department of Computer Science, Iowa State University, Ames, Iowa, USA
∗∗Department of Electrical and Electronic Engineering, Auckland University of Technology, New Zealand

Email: {nawaz.ali,eylguan}@ntu.edu.sg, salim9326@yahoo.com, sammy@iastate.edu, peter.chong@aut.ac.nz

Abstract—Road Side Units (RSUs) based data dissemination in VANETs got considerable attention to assist the inter vehicles communication. It not only overcomes the frequent vehicle-to-vehicle disconnection problem, but also provides predictable connectivity. In on-demand schedule, the RSU server invokes the underlying scheduling algorithm to choose the next broadcast data item. Majority of the scheduling algorithm selects a data item based on different metrics, which are usually: the deadline of a request, the size of an item and the popularity of an item. If the popular data items are broadcasted many times for maximizing the broadcast channel bandwidth, non-popular data items are starved, which creates unfairness to the pending requests for the non-popular data items. However, the non-popular data items may also be important for a vehicle. Hence, there is tradeoff of achieving the higher fairness and gaining the higher system throughput. In this study, we investigate this tradeoff in the context of RSU-based VANETs, and propose a fairness friendly approach. With the integration of a scheduling algorithm, the proposed approach can balance the tradeoff of the fairness of service and the system throughput. Simulation results support our claims.

I. INTRODUCTION

Efficient data dissemination in Vehicular Ad Hoc Networks (VANETs) is a key of success for many VANETs applications. A number of applications has been envisioned in Vehicular Ad Hoc Networks (VANETs) such as road safety, driving assistance, emergency public service, business, entertainment etc. [1], [2]. Realizing the importance of VANETs communication, the Federal Communications Commission (FCC) allocates the 5.850 to 5.925 GHz frequency band for vehicle-to-vehicle and vehicle-to- Road Side Units (RSUs) communication. FCC formulates the technical specifications for Dedicated Short Range Communication (DSRC) which support the transmission of large volume data within a short range [3].

In VANETs, as different vehicles may have interests to different data items, on-demand broadcasting is a popular approach for data dissemination [4], [5]. Recently, researchers have proposed the use of Road Side Units (RSUs) for supporting on-demand data broadcasts, particularly where strict time constraints are involved. In this kind of model, RSUs are placed along roadside to provide vehicle-to-infrastructure (V2I) connectivity [6], [7]. This kind of model is very useful during unfriendly VANETs environment such as off peak hour, night time etc. and on highways where vehicle density is low [8]. On the other hand, as an RSU is a stationary unit, connectivity is predictable and even connecting duration is longer. This reduces the vehicle-to-vehicle frequent disconnect problem. In RSU-based model, an RSU acts as a buffer point [9], [10] which stores the information that is useful for the vehicle such as present weather or road condition, accident warning, real time traffic update, digital map, location about the nearby gas station or restaurant, entertainment data or form download, value added service etc.

When a vehicle enters into the communication range of an RSU, it gets the chance to generate requests for its necessary information. A number of vehicle may requests for the same data item which is called popular data item also called hot data item, when a data item is accessed less frequently, we call it non-popular data item or cold data item [11]. As a hot data item is requested by many requests, by broadcasting a hot data item many vehicles can be satisfied simultaneously. On the other hand, cold data items are requested by less number of vehicles and get lower priority for broadcasting. However, cold data items are also important for a vehicle.

Fairness of service is the ability of the system to provide same service level to all types of data items [12]. However, due to continuous broadcasting of hot data items, cold data items are deprived from broadcasting. As a result, the fairness of service is violated and sometimes even these cold data items are not broadcast at all. Hence, usually the pending requests of cold data items have higher probability for missing deadlines and the corresponding vehicles cannot get the required information. None of the exiting scheduling algorithms take care about the fairness of service issue of requests.

There is tradeoff between achieving the fairness of services and gaining the higher system throughput [12]. This is because, on the one hand, to maximize the fairness system needs to broadcast cold data item, which eventually reduces the throughput. On the other hand, to maximize the throughput system needs to broadcast more hot data items which eventually reduces the fairness. Previously we have done some preliminary work in this area [13]. In that paper, we have shown that the exiting scheduling algorithms suffer from the request level starvation problem while satisfying requests. However, in this paper, we are trying to strike the balance between achieving the fairness of service and gaining the higher system throughput. To gain that we propose a fairness friendly approach. Extensive simulations confirm that the proposed approach integrated with a scheduling algorithm can strike the balance between the fairness of service and the high system throughput.

The rest of the paper is organized as follows: Section II outlines our system model. Section III demonstrates the performance analysis and finally Section IV concludes this paper.
II. SYSTEM MODEL

A. System architecture

We assume that VANETs services are provided to the vehicles at the hot spot zones such as at the gas stations or at the intersections of the roads where number of vehicles gather or pass usually higher than other areas. When a vehicle is in the transmission range of an RSU, it can generate requests for requesting either hot or cold data items from the RSU database.

Our system architecture is shown in Fig. 1. Each RSU supports two channels, one for user request and another for response. Through the request channel vehicles submit their requests which are inserted into the RSU service queue. RSU has a scheduler, before making the servicing decision, the scheduler invokes the underlying scheduling algorithm to select the item for the next broadcast. Then, the RSU broadcasts the selected data item through the downlink response channel. If an RSU broadcasts a hot data item, other requests set is denoted as $S(A(R_i,t,R^i))$, which can be defined as: $S(A(R_i,t,R^i)) = \{R_j[t] \neq i, R_j \in R^i & ID_j = ID_i & (R_i,t,R^i)_{t} = \text{satisfiable}\}$.

**Request’s Life Time:** A vehicle can generate request only within time range $[T_{in}, T_{out} - T_{serv}]$. Assume, the radius of the transmission range of an RSU is $R$ meter and average speed of vehicle within the transmission range of RSU is $V$ m/s. So, if the vehicle reaches at the transmission range of an RSU at time $t = 0$, and generates the first request at the same time, then the average deadline of the first request of the vehicle is, $T_{deadline} = \frac{2R}{V}$. So, at anytime $T$, the deadline of a request is,

$$T_{i,\text{deadline}} = \frac{2R}{V} - T_i$$

where $T_i = \text{random}(0.0, (\frac{2R}{V} - T_{i,\text{serv}}))$. Here, $T_i^*$ estimates the time when the request $R_i$ is generated within the transmission range of an RSU. As time passes, the deadline value of a request decreases. Request $R_i$ will be discarded from the scheduler when $T_{i,\text{deadline}} < T_{i,\text{serv}}$.

Note that the fairness of service is the ability of a system of servicing both the hot and cold data items with the equal probability. On the other hand, the throughput is the measurement of a system efficiency, which can be maximized by serving more hot data items than cold data items. Hence, it is not difficult to understand that, achieving the higher fairness of service and gaining the higher system throughput are two inter-conflicting issues and needs to reconcile these two issues.

B. Notations and assumptions

**Request:** When a vehicle submit a request $R_i$, it submit the following tuples:

$R_i = (V_{ID}, Req_{ID}, ID_i, T_{in}^i, T_{out}^i, T_i^*, T_{i,\text{deadline}})$.

Here,

$V_{ID}$: Vehicle ID;

$Req_{ID}$: Request ID;

$ID_i$: The ID of the requested data item;

$T_{in}^i$: The time the vehicle enters into the communication range of the RSU;

$T_{out}^i$: The time the vehicle leaves the communication range of the RSU, a vehicle can estimate this by its driving speed and RSU transmission range;

$T_i^*$: The time the request is generated;

$T_{i,\text{deadline}}$: The deadline assign by the request, beyond this time the request $R_i$ will be dropped;

**RSU Database (RSUDB):** RSUDB stores the updated data to satisfy vehicles’ on-demand requests. Here we use the following notations:

$DBSIZE$: Database size of an RSU;

$SIZE_i$: Size of the requested data item $ID_i$;

$P_i$: Popularity of the requested data item $ID_i$. Each time data item $ID_i$ requested, $P_i$ is increased by 1.

**Schedule:** When a vehicle submits a request, the request needs to be scheduled for getting the desired data item. Assume, at time $t$, a set of requests $R^i$ reside in the RSU received queue to be scheduled. Since each request needs to occupy the communication channel for data transmission, it should make sure that the servicing operation finishes before the vehicle moves out of the communication range. We assume that the RSU has single response channel and it is nonpreemptive, i.e. no other data items will be served until current serving one finished.

If a request $R_i \in R^i$ is scheduled to be served at $t$, we call $R_i$ is **satisfiable** if it meets, $t \geq T_{in}^i + (T_{i,\text{serv}} - T_{i,\text{out}})$. where, $T_{i,\text{serv}} = \frac{SIZE_i}{ChannelBandwidth}$, is the time needed to serve data item $ID_i$. Here, $ChannelBandwidth$ denotes the bandwidth of the response channel. If this condition is violated, we called it **unsatisfiable** request.

Due to the broadcast nature of wireless communication, for the downloading operation, when a data is broadcast, a set of requests waiting for the same data can be satisfied at the same time. We call such set of requests **shareable requests**. If a request $R_i \in R^i$ is scheduled at time $t$, its shareable requests set is denoted as $S(A(R_i,t,R^i))$, which can be defined as: $S(A(R_i,t,R^i)) = \{R_j[t] \neq i, R_j \in R^i & ID_j = ID_i & (R_i,t,R^i)_{t} = \text{satisfiable}\}$.

**RSU Database (RSUDB):** RSUDB stores the updated data to satisfy vehicles’ on-demand requests. Here we use the following notations:

$DBSIZE$: Database size of an RSU;

$SIZE_i$: Size of the requested data item $ID_i$;

$P_i$: Popularity of the requested data item $ID_i$. Each time data item $ID_i$ requested, $P_i$ is increased by 1.

**Schedule:** When a vehicle submits a request, the request needs to be scheduled for getting the desired data item. Assume, at time $t$, a set of requests $R^i$ reside in the RSU received queue to be scheduled. Since each request needs to occupy the communication channel for data transmission, it should make sure that the servicing operation finishes before the vehicle moves out of the communication range. We assume that the RSU has single response channel and it is nonpreemptive, i.e. no other data items will be served until current serving one finished.

If a request $R_i \in R^i$ is scheduled to be served at $t$, we call $R_i$ is **satisfiable** if it meets, $t \geq T_{in}^i + (T_{i,\text{serv}} - T_{i,\text{out}})$. where, $T_{i,\text{serv}} = \frac{SIZE_i}{ChannelBandwidth}$, is the time needed to serve data item $ID_i$. Here, $ChannelBandwidth$ denotes the bandwidth of the response channel. If this condition is violated, we called it **unsatisfiable** request.

Due to the broadcast nature of wireless communication, for the downloading operation, when a data is broadcast, a set of requests waiting for the same data can be satisfied at the same time. We call such set of requests **shareable requests**. If a request $R_i \in R^i$ is scheduled at time $t$, its shareable requests set is denoted as $S(A(R_i,t,R^i))$, which can be defined as: $S(A(R_i,t,R^i)) = \{R_j[t] \neq i, R_j \in R^i & ID_j = ID_i & (R_i,t,R^i)_{t} = \text{satisfiable}\}$.

**Request’s Life Time:** A vehicle can generate request only within time range $[T_{in}^i, T_{out}^i - T_{i,\text{serv}}]$. Assume, the radius of the transmission range of an RSU is $R$ meter and average speed of vehicle within the transmission range of RSU is $V$ m/s. So, if the vehicle reaches at the transmission range of an RSU at time $t = 0$, and generates the first request at the same time, then the average deadline of the first request of the vehicle is, $T_{i,\text{deadline}} = \frac{2R}{V}$. So, at anytime $T$, the deadline of a request is,

$$T_{i,\text{deadline}} = \frac{2R}{V} - T_i$$

where $T_i = \text{random}(0.0, (\frac{2R}{V} - T_{i,\text{serv}}))$. Here, $T_i^*$ estimates the time when the request $R_i$ is generated within the transmission range of an RSU. As time passes, the deadline value of a request decreases. Request $R_i$ will be discarded from the scheduler when $T_{i,\text{deadline}} < T_{i,\text{serv}}$.

Note that the fairness of service is the ability of a system of servicing both the hot and cold data items with the equal probability. On the other hand, the throughput is the measurement of a system efficiency, which can be maximized by serving more hot data items than cold data items. Hence, it is not difficult to understand that, achieving the higher fairness of service and gaining the higher system throughput are two inter-conflicting issues and needs to reconcile these two issues.

C. Existing on-demand scheduling algorithms

Different on-demand scheduling algorithms use different criteria for request selection for broadcasting an item. The following existing on-demand scheduling algorithms are adopted in our RSU-based VANETs settings.
• **First Come First Serve (FCFS)** [15]: It selects the requests according to the request arrival ordering in the RSU received queue. It is a base line scheduling algorithm.

• **Earliest Deadline First (EDF)** [16]: When a vehicle submits a request it assigns its deadline, hence each request has its own deadline. EDF algorithm checks the deadline of all the requests in its received queue and selects the one with the minimum deadline.

• **Most Request First (MRF)** [17]: MRF chooses a data item for broadcasting based on data item characteristic rather than request characteristic. Recalling that when a data item is requested by a request, its popularity increased by one. MRF broadcasts the data item with the maximum popularity, in other words, it broadcasts the current hottest data item in the database.

• **Shortest Service Time First (SSTF)** [18]: Like MRF, SSTF also selects a data item according to the characteristic of data item rather than request. Recalling that service time of a data item is the ratio of the size of that data item to the channel bandwidth. SSTF calculates the service time of all the requested data items in the RSU database and broadcasts the one with the minimum service time.

• **Deadline Size Inverse Number of pending request (DSIN)** [9]: DSIN incorporates three metrics: the deadline of the request (D), the size of the requested data item (S) and the number of pending requests of the data item (N), namely, the popularity of that data item. DSIN calculates the $DSIN\_Value$ of all the requests in the received queue and selects the one with the minimum $DSIN\_Value$, where, $DSIN\_Value = \frac{DSIZEMAX}{N\times S}.$

**D. Performance metrics**

To evaluate the system performance we adopt the following performance metrics:

• **Deadline Missed Ratio**: It is the ratio of the number of missed requests to the number of received requests. So, $DeadlineMissRate(DMR) = \frac{DMN}{DMN + SN}$, where DMN and SN are the number of deadline missed requests and the number of satisfied requests, respectively. The lower DMR means the system can serve more requests before their deadlines expire.

• **Average Response Time**: It is the average wait time from submitting the request to get the response. A lower average response time means the system can serve more quickly, namely the system is more efficient.

• **Throughput**: Throughput is the total number of requests served by the system by unit time. Higher throughput means system is more efficient.

Hence to achieve the higher system performance, a scheduling algorithm should achieve the lower deadline miss rate and the lower response time but the higher throughput. To analyze the system performance, we set up the following simulation model.

**E. Experimental setup**

Our simulation environment is based on the system architecture shown in Fig. 1. We perform our simulation experiment using CSIM19 [20]. Other than the CSIM default parameters, the explicit parameters used for the simulation are shown in the Table I.

Vehicles Request Generation InterVal (RGIV) is exponentially distributed and its probability density function is: $P(x) = \lambda e^{-\lambda x}.$ We use default $\lambda$ value is 0.33 for simulation. The data item access pattern is shaped by the commonly used Zipf distribution [21]. $\theta$ is the skewness parameter ranging from 0.0 to 1.0. $\theta$ equals 0 means the Uniform distribution and the increasing $\theta$ means the more skewed distribution. The access probability of the $i^{th}$ data item is: $P(i) = \frac{1}{\sum_{n=1}^{N_n} \frac{1}{\theta^n}},$

where $N_n = |DBSIZE|$

A vehicle can generate requests until it exceeds the RSU transmission range. Maximum number of requests generation by a vehicle is defined by the Poison process, where we set the Poisson mean is 25.0. We use the random data item size distribution for generating RSU database, where Random Size Distribution (RAND) [22] is, $DataItemSize[i] = SIZEMIN + [uniform(0.0, 1.0) * (SIZEMAX − SIZEMIN + 1)].$

Here, SIZEMIN and SIZEMAX are the minimum and the maximum data item size in the database, respectively and $i = 1, 2, \ldots, |DBSIZE|.$

For the experimental data generation, we let all the vehicles pass the RSU transmission range and generate requests, when they cross the RSU range, we let these vehicles go in the the RSU range from starting. We do this repeatedly, until we get the stable data from the same parameter settings. For performance evaluation, we take the mean data when 95% confidence interval has been achieved.

To generate requests for both hot and cold data items, we let half of the total vehicles generate requests using the skewed data item distribution ($\theta$ value 1.0) and other half using the Uniform data item distribution ($\theta$ value 0.0).

**III. PERFORMANCE ANALYSIS**

**A. Performance analysis of existing on-demand scheduling algorithms**

In this subsection we have analyzed the performance of different existing scheduling algorithms in terms of deadline missed ratio (DMR), average throughput and average response time. The simulation results show that DSIN which considers the request deadline, the data item size, and the data item popularity in scheduling, outperforms a number of existing algorithms such as MRF, FCFS, EDF, and SSTF. Due to space limitation, we omit the corresponding descriptions and graphs.

**B. Fairness analysis of existing on-demand scheduling algorithms**

If we look from the fairness of service point of view, we see that the popularity based algorithms suffer badly from providing the fair service to both the hot and cold data items equally (Fig. 2). To analyze the fairness problem, we adopt the following two performance metrics:

• **Average Fairness**: It is the ratio of the total number of different data items that are broadcast to the total number of different data items that are requested by the vehicles. Here data items means both hot and cold data items. Higher average fairness means the system is fairness friendly.

• **Request Served Percentage**: It is the percentage number of served requests following of either access pattern (the skewed or uniform) to the total number of requests generates in that particular access pattern. We feed the
mix traffic (requests generated using both the skewed and the uniform item access patterns) in the system and then we measure both the Uniformly Distributed Request Served Percentage (UDRSP) and Skewed Distributed Request Served Percentage (SDRSP). Then we compare the UDRSP and SDRSP for each algorithm to estimate its fairness of service to the data items. If UDRSP and SDRSP are equal or close to equal, it means that algorithm provides fair service. Hence, SDRSP and UDRSP represents the successfully served requests were pending for the hot and cold data items, respectively.

Fig. 2 exhibits the fairness of service to the both types of (hot and cold) data items of different existing on-demand scheduling algorithms for the equal number of requests generated from the skewed and the uniform item access patterns. From 2(a), we see that when number of vehicles equals 50, most of the algorithms can achieve higher fairness (around 50%). However, with the increasing number of vehicles, fairness declines to around 10% (for 300 vehicles). It is because, a higher number of vehicles generate both a higher number of Uniform distributed requests pend for cold data items and a higher number of skewed distributed requests pend for popular data items. Since popularity based algorithm considers the data item popularity, it broadcasts popular data items most of the time and suffers much from providing fair service, as a consequence, MRF shows the lowest fairness performance (Fig. 2(a)). However, as SSTF broadcasts data items based on the data item size and ignores the item access pattern, hence it has the best overall fairness performance among these algorithms.

Figs. 2 (b)-(f) explain the reasons of the fairness performance of an algorithm. It exhibits the percentage measurement of SDRSP and UDRSP values of DSIN, MRF, EDF, FCFS and SSTF. Except SSTF algorithm all the algorithms have SDRSPs’ portions larger than UDRSPs. The reason behind this is, MRF and DSIN serve more skewed distributed requests. EDF serves requests according to the request deadline, and popular data items are requested by many pending requests, among these requests some have critical deadlines, satisfying these type request means it serves indirectly popular data items, hence EDF serves many skewed distributed requests. FCFS serves based on the requests arrivals, for skewed distributed requests, most of the requests pend for some selective hot data items, hence broadcasting a data item requested by an earlier arrived one skewed distributed request, FCFS can satisfy many requests simultaneously. Hence FCFS also suffers from the fairness of service problem. However, due to the inherent nature, SSTF can achieve nearly equal portion of SDRSP and UDRSP and have a better fairness performance.

From the above analysis, we conclude that DSIN has the best system performance but suffers from providing the fair service to the both hot and cold data items. On the other hand, though SSTF has a better fairness performance, its system performance is very poor. Our target is to provide a fairness friendly scheduler which also can achieve a higher overall system performance. As we can see that none of the existing scheduling algorithms cannot achieve two performance objectives together namely, achieving the higher fairness of service and gaining the higher system throughput. So, it is imperative to devise a new solution.

C. Proposed fairness friendly (FF) approach

We propose a fairness friendly (FF) approach which integrates the underlying scheduling algorithm to choose a request for the next broadcast and also to ensure for providing the fair service to both the hot and cold data items. The pseudocode of our algorithm is shown in Algorithm 1. Initially, all the requested data items’ (for the pending requests) status are initialized to Unserved (Lines 5-6). According to the underlying scheduling algorithm, the next most prioritized data item will be selected among the data items with the Unserved status. The selected data item will be broadcast and then its status will be updated to Served. Consequently the service queue (SQ) will be updated by removing all the satisfied requests (Lines 7-13). This mechanism ensures that irrespective of cold or hot data items, all the requested data items get chance to be broadcast.

<table>
<thead>
<tr>
<th>TABLE I</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PARAMETERS</strong></td>
</tr>
<tr>
<td><strong>Parameter</strong></td>
</tr>
<tr>
<td>NumVehicle</td>
</tr>
<tr>
<td>RGIW (A)</td>
</tr>
<tr>
<td>DBSIZE</td>
</tr>
<tr>
<td>SIZEMIN, SIZEMAX</td>
</tr>
<tr>
<td>ChannelBandwith</td>
</tr>
<tr>
<td>N</td>
</tr>
<tr>
<td>V</td>
</tr>
<tr>
<td>μ</td>
</tr>
<tr>
<td>θ</td>
</tr>
</tbody>
</table>

**Algorithm 1**: Fairness Friendly (FF) approach.

1 /* SQ denotes the service queue of pending requests */
2 /* ID denotes the serving status of data item */
3 /\ ID ∈ RSUDB
4 Upon arrival of a new request, the request will be added in SQ;
5 /* Initialize the serving status of all requested data items to ‘unserved’ */
6 for ∃ Rk ∈ SQ do
7 -\ IDk ← Unserved;
8while SQ ≠ ∅ do
9 -\ IDk ← Invoke the underlying scheduling algorithm for ∀ IDk = Unserved, where IDk ∈ RSUDB;
10 BroadcastIDk;
11 /* Update SQ by removing the already satisfied requests */
12 for ∃ Rj ∈ SQ requests for IDj do
13 -\ SQ ← SQ − {Rj};
14 Upon arrival of a new request Rm during scheduling and broadcasting process /* update SQ and the data item serving status */
15 SQ ← SQ + {Rm};
16 IDm ← Unserved;
This is one of our objectives, namely to ensure the fairness of service to the data items. An already broadcast data item (that means the status of that data item changed to *Served*) also will get chance to be broadcast when any new request arrives in the system querying for that item (Lines 14-16). Unlike round robin (RR) algorithm, the proposed FF approach doesn’t just broadcast a data item by turns, rather broadcasts a data item according to the used scheduling algorithm which ensures the increasing system throughput which is the second objective of this paper.

From our previous analysis in Sections III-A and III-B, we have found that although DSIN achieves the best throughput among the existing on-demand scheduling algorithms, cannot provide fair service to the data items. We add our proposed FF approach with DSIN (as a scheduling algorithm) which is called FFDSIN with the objective of FFDSIN will provide both the higher fairness of service and higher system performance. To demonstrate the superiority of FFDSIN algorithm along with the already described on-demand scheduling algorithms (Section II-C), we also analyze the performance of data item level Round Robin (RR) algorithm. RR algorithm blindly serves the requested data items from the fist data item to the last data item of the database sequentially irrespective of vehicles data access pattern i.e. irrespective of hot and cold data items.

D. System performance and fairness comparison of FFDSIN algorithm with other algorithms

Fig. 3 shows the performance of FFDSIN algorithm along with other existing on-demand scheduling algorithms for an equal number of requests generated from both the skewed and uniform data item access patterns. FFDSIN algorithm has nearly equal performance of DSIN algorithm and better performance than all other existing on-demand scheduling algorithms in terms of the deadline miss rate, the throughput and the response time, as shown in Figs. 3(a), 3(b), and 3(c), respectively. Regarding all these performance metrics, DSIN
may achieve a little bit better performance than FFDSIN, because, DSIN does not care to provide the fair service of all the pending request, rather it broadcasts the data item with the minimum DSIN_Value. In this context, DSIN may broadcast same smaller sized popular data item again and again for maximizing the system performance. However, for ensuring fairness of service FFDSIN broadcasts the non-broadcast data item with the minimum DSIN_Value. Hence, though a data item may have the minimum DSIN_Value among all the requested items, however, already been broadcast and others data items are starving for service, FFDSIN will broadcast one of the starved data items not the already broadcast one to ensure the fairness of service.

Fig. 4 exhibits the superiority of FFDSIN algorithm in terms of achieving the higher fairness over other on-demand scheduling algorithms. It achieves the best fairness performance among all the scheduling algorithms (Fig. 4(a)). This is because, FFDSIN broadcasts both the hot and the cold data items equally. This is the reason why FFDSIN can have the higher fairness. To testify the potentiality and scalability of higher fairness achievement of FFDSIN over DSIN, we change the data item access pattern by generating varying number of requests from both patterns for 100 vehicles. From Fig. 4(b), we see that while 100% request generated from the skewed access pattern (and 0% from the random access pattern), both DSIN and FFDSIN can achieve a higher fairness. However,
with the decreasing skewed traffic and the increasing uniform traffic both the algorithms show decreasing fairness. This is because, with the increasing of the uniform traffic, the number of pending requests for the cold data item increases which results in decreasing the sharable requests. Hence an RSU has to satisfy higher number of cold data than hot data items. In this regard, to improve the fairness a scheduler needs to satisfy a higher number of cold data items. However, to satisfy many cold data items, some of the requests may miss their deadlines. The data items requested by those deadline missed requests have no chance to be broadcast, hence the system average fairness decreases. Nevertheless, FFDSIN can retain its fairness friendly superiority than DSIN under different mix traffic for its fairness friendly characteristics (Fig. 4(b)). RR cannot achieve neither a higher system performance (Fig. 3(a)) nor a better fairness (Fig. 4(a)), because, it starts broadcasting data items from the front side of the database. As in the skewed data item access patterns majority of the requests access the front data items of database, when RR serving these popular data items sequentially, the requests in the uniform data access patterns pending for the data items residing throughout the whole database randomly, miss their deadlines. Hence RR cannot achieve the higher overall system performance as shown in Fig. 3(a). Now we conclude here by restating that our proposed FF approach with DSIN algorithm can achieve a better performance than all other existing scheduling algorithms and have nearly equal overall system performance of DSIN. However, it has distinguishable better fairness performance than all the algorithms including DSIN and RR.

IV. CONCLUSIONS

In this paper, first of all, we analyze the system throughput and fairness of service of different existing on-demand scheduling algorithms in RSU-based VANETs model. We find that scheduling algorithm DSIN has the higher system throughput but suffers from achieving a decent fairness of service. On the contrary, SSTF shows the higher fairness of service but fails to achieve a higher system throughput. To achieve these two performance metrics in one algorithm, we propose a new approach called FF (fairness friendly). We perform a series of simulation experiments to demonstrate the superiority of FFDSIN (DSIN with the integration of FF) over the existing algorithms. It shows that FFDSIN outperforms all other algorithms in terms of achieving higher fairness and shows equal or nearly equal system performance of DSIN, which is the best algorithm among all the studied algorithms.

REFERENCES