On striking the balance between the fairness of service and throughput in Road Side Units (RSUs) based vehicular ad hoc networks (VANETs)

G.G. Md. Nawaz Ali*
School of Electrical & Electronic Engineering,
Nanyang Technological University,
50 Nanyang Avenue, Singapore
Email: nawaz.ali@ntu.edu.sg
*Corresponding author

Md. Abdus Salim Mollah
Department of Computer Science and Engineering,
Khulna University of Engineering & Technology,
Khulna-9203, Bangladesh
Email: salim9326@yahoo.com

Syeda Khairunnessa Samantha
Department of Computer Science,
Iowa State University,
Ames, Iowa, USA
Email: sammy@iastate.edu

Peter Han Joo Chong
Department of Electrical and Electronic Engineering,
Auckland University of Technology,
Auckland, New Zealand
Email: peter.chong@aut.ac.nz

Yong Liang Guan
School of Electrical & Electronic Engineering,
Nanyang Technological University,
50 Nanyang Avenue, Singapore
Email: eylguan@ntu.edu.sg

Abstract: Data dissemination using Road Side Units (RSUs) in Vehicular Ad Hoc Networks (VANETs) got considerable attention to assist the inter-vehicles communication to overcome the vehicle-to-vehicle frequent disconnection problem. An RSU invokes the underlying scheduler to choose a data item to broadcast in order to satisfy the pending requests of vehicles. Conventionally, a scheduler 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. In the conventional scheduling manner, normally the popular data items are
broadcasted many times for maximising the broadcast channel bandwidth. As a consequence, non-popular data items are broadcast very few times, 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 trade-off of achieving higher fairness and gaining higher system throughput. In this study, we investigate this trade-off in the context of RSU-based VANETs and propose a fairness-friendly approach with which the integration of a scheduling algorithm can balance the trade-off of the fairness of service and the system throughput. Simulation results support our proposed approach and offer the expected results.

**Keywords:** vehicular ad hoc networks; RSU; road side units; scheduling; fairness; throughput.


**Biographical notes:**

G.G. Md. Nawaz Ali is currently a Postdoctoral Research Fellow at the School of Electrical and Electronic Engineering, Nanyang Technological University (NTU), Singapore. He received his PhD in the Department of Computer Science, City University of Hong Kong, in 2013. He received his BSc degree in Computer Science and Engineering from Khulna University of Engineering & Technology, Bangladesh, in 2006. He is a Member of IEEE and IEEE VTS. His current research interests include wireless broadcasting, mobile computing, network coding and ad hoc networking with a focus on vehicular ad hoc networking.

Md. Abdus Salim Mollah is currently an Assistant Professor at the Department of Computer Science and Engineering of Khulna University of Engineering & Technology, Bangladesh. He received his MSc degree in Computer Science and Engineering from Bangladesh University of Engineering and Technology (BUET) and BSc degree from Khulna University, Khulna, Bangladesh. His research interest includes data base management system, computer networks and mobile ad hoc networks.

Syeda Khairunnesa Samantha is a PhD Candidate at the Department of Computer Science of Iowa State University, USA. She received her BSc degree in Computer Science and Engineering from Khulna University of Engineering & Technology, Bangladesh, in 2011. She published a couple of papers in the international journals and conference proceedings. Her current research interests include big data analytics, computer networking, mobile computing and ad hoc networking.

Peter Han Joo Chong is currently a Professor and Head of Department of Electrical and Electronic Engineering at Auckland University of Technology, New Zealand. He received the PhD degree in Electrical and Computer Engineering from the University of British Columbia, in 1996 and 2000, respectively. He is an Adjunct Faculty at the Department of Information Engineering, Chinese University of Hong Kong. He was previously an Associate Professor (tenured) from 2009 to 2016 and Assistant Professor from 2002 to 2009 in the School of Electrical and Electronic Engineering at Nanyang Technological University (NTU), Singapore. He is the Co-Founder of P2 Wireless Technology based in Hong Kong. He is an Editorial Board Member of *Security and Communication Networks* and *KSII Transactions on*
On striking the balance between the fairness of service and throughput

Internet and Information Systems. His research interests are in the areas of mobile communications systems including radio resource management, multiple access, MANETs/VANETs, multihop cellular networks and Internet of Things/Vehicles.

Yong Liang Guan obtained his PhD from the Imperial College of London, UK, and Bachelor of Engineering with first class honours from the National University of Singapore. He is a tenured Associate Professor at the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore. His research interests broadly include coding and signal processing for communication and storage systems.

This paper is a revised and expanded version of a paper entitled ‘A novel approach on striking the balance between the fairness and throughput in vehicular networks’ presented at the ICWN’16, 25–28 July, Las Vegas, Nevada, USA.

1 Introduction

Efficient data dissemination in Vehicular Ad Hoc Networks (VANETs) is a key of success for many VANETs applications. A number of applications have been envisioned in VANETs, such as road safety, driving assistance, emergency public service, business, entertainment, etc. (Schoch et al., 2008). Realising the importance of VANETs communication, the Federal Communications Commission (FCC) allocates the 5.850–5.925 GHz frequency band for vehicle-to-vehicle and vehicle-to-RSUs (Road Side Units) communication. In 2004, the FCC formulated the technical specifications for Dedicated Short Range Communication (DSRC), which supports transmission of large-volume data within a short range (Bera et al., 2006). In general, DSRC refers to a family of standards of Wireless Access in Vehicular Environments (WAVE). The standards include IEEE 802.11p for PHY and MAC layers, which is the modified version of IEEE 802.11. It also includes IEEE 1609.4, 1609.3 and 1609.2, respectively, for multi-channel operations, network and security services. The SAE J2735 message set dictionary specifies the message formats for vehicle-based applications (Kenney, 2011; Liu et al., 2015; Ali et al., 2016).

In VANETs, as, on the one hand, multiple vehicles may have the interest on the same data item, on the other hand, different vehicles may have interest on different data items, on-demand broadcasting is a popular approach for such kind of dynamic data dissemination (Aksoy and Franklin, 1999; Ali et al., 2014; Xu et al., 2006). Recently, researchers have proposed the use of RSUs for supporting on-demand data broadcasts, particularly where strict time constraints are involved. In this kind of model, RSUs are placed on the roadside to provide vehicle-to-infrastructure (V2I) connectivity for improved data dissemination (Ali et al., 2013; Liu and Lee, 2010; Zhang et al., 2010). This kind of model is very useful during unfriendly VANETs environment such as off-peak hour, night time, etc., and highways where vehicle density is low (Ali et al., 2014; Lochert et al., 2007), which results in lesser vehicle-to-vehicle communication possibility. 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 models, an RSU acts as a buffer point (Jiang and Du,

2015; Zhang et al., 2010) 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 services, 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 vehicles may request 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 (Chen et al., 2010; Chen et al., 2013). As a hot data item has many requests, many vehicles can be satisfied simultaneously by broadcasting a hot data item. On the other hand, cold data items are requested by less number of vehicles and therefore 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 the same service level to all types of data items (Bejerano and Bhatia, 2006). However, due to continuous broadcasting of hot data items, cold data items are deprived from broadcasting. As a result, 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 existing scheduling algorithms such as First Come, First Served (FCFS) (Wong and Ammar, 1985), Most Requested First (MRF) (Wong, 1988), Earliest Deadline First (EDF) (Xuan et al., 1997), etc., do not take care of the fairness of service issue of requests.

There is trade-off between achieving fairness of service and gaining a higher system throughput (Bejerano and Bhatia, 2006). This is because, on the one hand, maximising the fairness system needs broadcasting cold data item, which eventually reduces the throughput, and, on the other hand, maximising the throughput system needs broadcasting more hot data items, which eventually reduces the fairness. We have done some preliminary work in this area previously (Samantha et al., 2012). In that paper, we have shown that the existing 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 fairness of service and gaining higher system throughput. Our main contributions in this paper are in threefold:

- First, we analyse why the existing scheduling algorithms fail to provide fairness of service in the context of VANETs.
- Second, we propose a fairness-friendly (FF) approach, with the integration of a scheduling algorithm, which can strike a balance between fairness of service and high system throughput.
- Finally, we perform a series of simulation experiments to support our proposed approach.

The rest of the paper is organised as follows. Section 2 describes the related works. Section 3 outlines our system model. Section 4 demonstrates the performance analysis, and, finally, Section 5 concludes the paper.
On striking the balance between the fairness of service and throughput

2 Related work

A number of researchers research to find a stable data dissemination infrastructure in highly mobile and sparsely connected VANETs environments (Lochert et al., 2007; Nadeem et al., 2006; Zhao et al., 2007). Nadeem et al. (2006) formulate a data push communication model for vehicle-to-vehicle communication without any roadside infrastructure support. Zhao et al. (2007) also propose a vehicle-to-vehicle communication model where vehicles at the intersections buffer data and can disseminate to other vehicles later on. Lochert et al. (2007) analyse the data dissemination performance of different VANETs environments. They propose that the backbone connected RSUs model substantially outperforms the scenarios. Zhang et al. (2010) provide a single RSU VANETs model which maintains both upload and download queues and tries to get a balanced service among them. Yi et al. (2008) propose a mesh RSUs infrastructure based on both space and time dimensions and formulate reliability- and fairness-based algorithms. Ali et al. (2011) analyse the performance of different on-demand scheduling algorithms for incorporating upload queue with download queue in RSU-based VANETs. Chen et al. (2009) study the performance of the effectiveness of certificate revocation distribution in VANETs with and without enabling vehicle-to-vehicle communication. Liu and Lee (2010) analyse the dynamic traffic characteristics in RSU-based VANETs. They propose using different channels to disseminate different types of data and applying a push-and-pull data dissemination technique based on the volume of requests at the RSU server. Shahverdy et al. (2010) investigate the issue of large-scale data download from RSU to vehicles in multiple RSUs interconnected model. Ali et al. (2014, 2016) propose cooperative load balancing among multiple RSUs.

Some researchers have focused on the optimisation of data dissemination in real-time environments. Ng et al. (2008) study different scheduling algorithms in real-time environment. Xu et al. (2006) present Slack Time Inverse Number (SIN) of pending requests for time-critical on-demand requests. Lee et al. (2006) propose a preemptive algorithm called PRDS, which deals with request urgency, data item size and popularity. Liu and Lee (2010) study the performance of different on-demand scheduling algorithms for multi-item requests in multi-channel broadcasting. Chen et al. (2010) also device a new scheduling algorithm, Dynamic Temperature Inverse Urgency (DTIU), for serving multi-item requests efficiently. Ali et al. (2013) propose the Cooperative Query Serving (CQS) scheme for serving multi-item requests cooperatively in multi-RSU structures. However, none of the above works consider the fairness and throughput issues jointly in RSU-based VANETs.

3 System model

3.1 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 the number of vehicles that gather or pass is usually higher than in 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 Figure 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 service queue of RSU server. The RSU maintains 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, usually a number of requests become satisfied. A vehicle can generate request or receive responses only when it is within the transmission range of an RSU. A vehicle can submit request irrespective of whether its previous submitted request is successful or unsuccessful within the RSU transmission range, which is so called an open system model (Liu and Lee, 2010). Moreover, more than one vehicle can request for the same data item at the same time. The system model preliminaries are stated in the following sections.

3.2 Notations and assumptions

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

$$ R_i = (V_{id}, Req_{id}, ID_i, T_{in}^i, T_{out}^i, T_r^i, T_{deadline}^i). $$

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;

$T_r^i$: the time the request is generated;
On striking the balance between the fairness of service and throughput

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

**RSU database.** RSU database 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 $ID_i$ is 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 $R'$ reside in the RSU service 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 an RSU has a single response channel and it is non-preemptive, i.e. no other data items will be served until the current serving one is finished.

If a request $R_i \in R'$ is scheduled to be served at $t$, we call $R_i$ is satisfying if it meets

$$t \geq T_i^{\text{serv}} \text{ and } (t + T_i^{\text{serv}}) \leq T_i^{\text{out}}$$

where $T_i^{\text{serv}} = \frac{SIZE_i}{\text{ChannelBandwidth}}$ is the time needed to serve data item $ID_i$ and $\text{ChannelBandwidth}$ denotes the bandwidth of the response channel. If this condition is violated, we called it an unsatisfiable request.

Due to the broadcast nature of wireless communication, for the downloading operation, when data are 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 $R_i \in R'$ is scheduled at time $t$, its shareable requests set is denoted as $SA(R_i, t, R')$, which can be defined as

$$SA(R_i, t, R') = \{R_j \mid j \neq i, R_j \in R' \text{ and } ID_j = ID_i \text{ and } (R_j, t, R') \text{ is satisfying} \}$$

A schedule can be expressed as a sequence of the requests and their scheduling time. For example, assume a request set $R' = \{R_1, R_2, R_3, R_4, R_5, R_6\}$. A possible schedule is $S = (R_1, t_1) \rightarrow (R_2, t_2) \rightarrow (R_3, t_3) \rightarrow (R_4, t_4)$, which indicates that $R_2$ is served at $t_1$, $R_3$ is served at $t_2$, $R_1$ is served at $t_3$ and the items in () are shareable requests being served at the same time.

**Request’s lifetime.** A vehicle can generate request only within time range $[T_{i, \text{in}}, T_{i, \text{out}}]$. Assume the radius of the transmission range of an RSU is $R$ metres and the average speed of vehicle within the transmission range of an 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

$$T_{i, \text{deadline}}^{\text{first request}} = \frac{2R}{V}.$$ So, at anytime $T$, the deadline of a request is

$$T_{i, \text{deadline}} = \frac{2R}{V} - T_i^{\text{serv}}.$$
where $T_i' = \text{random} \left(0.0, \frac{2R}{V} - T_{\text{serv}}\right)$. 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 $T_{\text{deadline}} < T_{i'}$. 

Definition 3.1 (fairness of service): In the on-demand scheduling according to the access probability of data items, some accessed data items become popular than others. These popular data items are called hot data items and others are called cold data items. Traditionally, a popularity-based scheduling algorithm serves hot data items with a higher probability than cold data items. This creates unfairness to the cold data items. The fairness of service is the ability of a system to service both the hot and cold data items with the equal probability.

Definition 3.2 (system throughput): Throughput is the measurement of system efficiency. It measures how many requests a system can serve by unit time. Clearly, if a system serves the hot data items, it increases the system throughput. In contrast, if it serves the cold data items, the system throughput reduces.

From the above two definitions, 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.

3.3 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.

- **FCFS** (Wong and Ammar, 1985). It selects the requests according to the request arrival ordering in the RSU service queue. It is a baseline scheduling algorithm.

- **EDF** (Xuan et al., 1997). When a vehicle submits a request, it assigns its deadline; hence, each request has its own deadline. The EDF algorithm checks the deadline of all the requests in its service queue and selects the one with the minimum deadline.

- **MRF** (Wong, 1988). MRF chooses a data item for broadcasting based on data item characteristic rather than request characteristic. Recall that when a data item is requested by a request, its popularity increases 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)** (Wu. and Cao, 2001). Like MRF, SSTF also selects a data item according to the characteristic of data item rather than request. Recall that the 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 item in the RSU database and broadcasts the one with the minimum service time.

- **Deadline Size Inverse Number of pending request (DSIN)** (Zhang et al., 2010). 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
On striking the balance between the fairness of service and throughput

popularity of that data item. DSIN calculates the DSIN\_Value of all the requests in
the service queue and selects the one with the minimum DSIN\_Value, where,
$$DSIN\_Value = \frac{D \times S}{N}.$$ 

3.4 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. 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
  getting a 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
  time unit. Higher throughput means the system is more efficient.

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

3.5 Experimental set-up

Our simulation environment is based on the system architecture shown in Figure 1. We
perform our simulation experiment using CSIM19 (Schwetman, 2001). Other than the
CSIM default parameters, the explicit parameters used for the simulation are shown in
the Table 1.

Table 1 Simulation parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NumVehicle</td>
<td>100</td>
<td>25–300</td>
<td>Max. no. of vehicles in the RSU service range at a time</td>
</tr>
<tr>
<td>RGIV ($\lambda$)</td>
<td>0.33</td>
<td>–</td>
<td>Request generation interval of each vehicle</td>
</tr>
<tr>
<td>DBSIZE</td>
<td>500</td>
<td>–</td>
<td>Number of data items in the database</td>
</tr>
<tr>
<td>SIZEMIN, SIZEMAX</td>
<td>10, 512</td>
<td>–</td>
<td>Minimum and maximum data item size in the database</td>
</tr>
<tr>
<td>ChannelBandwidth</td>
<td>100</td>
<td>–</td>
<td>Channel broadcast bandwidth kB/s</td>
</tr>
<tr>
<td>$R$</td>
<td>350 m*</td>
<td>–</td>
<td>RSU communication range</td>
</tr>
<tr>
<td>$V$</td>
<td>40 km/h</td>
<td>–</td>
<td>Vehicle average speed</td>
</tr>
<tr>
<td>$\mu$</td>
<td>25.0</td>
<td>–</td>
<td>Poisson means for controlling requests generation</td>
</tr>
<tr>
<td>$\theta$</td>
<td>1.0, 0.0</td>
<td>–</td>
<td>Skewness parameter of Zipf distribution</td>
</tr>
</tbody>
</table>

Source: Yi et al. (2008)
Vehicles Request Generation InterVal (RGI-V) is exponentially distributed and its probability density function is \( P(x) = \lambda e^{-\lambda x} \). We use default \( \lambda \) value as 0.33 for simulation. The data item access pattern is shaped by the commonly used Zipf distribution (Zipf, 1949). \( \theta \) is the skewness parameter ranging from 0.0 to 1.0. If \( \theta \) equals 0, then there is uniform distribution and an increasing value of \( \theta \) indicates a more skewed distribution. The access probability of the \( i \)th data item is \( P(i) = \frac{1}{\sum_{n=1}^{N} \frac{1}{n^\theta}} \), where \( N = |DBSIZE| \).

A vehicle can generate requests until it exceeds the RSU transmission range. The maximum number of requests generated by a vehicle is defined by the Poisson process, where we set the Poisson mean as 25.0. We use the random data item size distribution for generating the RSU database, where Random Size Distribution (RAND) (Xu et al., 2004) is

\[
\text{DataItemSize}[i] = \text{SIZEMIN} + \left[ \text{uniform}(0.0, 1.0) \times (\text{SIZEMAX} - \text{SIZEMIN} + 1) \right]
\]

where \( \text{SIZEMIN} \) and \( \text{SIZEMAX} \) are the minimum and maximum data item sizes 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 to again go in the RSU range from the start. 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 use half of the total generated requests of vehicles to use the skewed data item distribution (\( \theta \) value of 1.0) and the other half to use the uniform data item distribution (\( \theta \) value of 0.0).

4 Performance analysis

4.1 Performance analysis of existing on-demand scheduling algorithms

Figure 2 shows the performance of different on-demand scheduling algorithms in terms of the deadline miss rate, the average throughput and the average response time. With the increasing number of vehicles in the RSU service range, more number of requests are submitted to the RSU. As more number of requests wait in the RSU service queue, while broadcasting some of the requested data items, many other pending requests may miss the deadline. As a result, with increasing number of vehicles, the deadline miss rate increases (Figure 2a). However, with increasing number of requests, the number of sharable requests increases. Hence, the accessing probability of some of the data items increases, namely some of the data items become very popular. By serving these popular data items, a scheduler can increase its throughput. That is why, from Figure 2(b), we find that increasing the number of vehicles results in all the scheduling algorithms having a better system throughput. Nevertheless, as DSIN considers all three metrics, namely request deadline, data item size and data item popularity, it outperforms the others. For the same effect of sharable requests, by broadcasting a popular data item, many requests become satisfied concurrently. Hence, pending requests of popular data items receive
quick response. However, with increase in the number of requests, some of the requests may have to wait for longer time to get service, which increases the response time of these requests. Popularity-based algorithms (such as MRF, DSIN) get a better advantage than other algorithms from the shareable requests set; hence, with the above two counter reasons, this type algorithms can minimise the average response somewhat; others have more or less the same average response time (Figure 2c).

**Figure 2** Performance of different on-demand scheduling algorithms under varying number of vehicles. (a) Deadline Missed Ratio (DMR); (b) average throughput; (c) average response time

4.2 *Fairness analysis of existing on-demand scheduling algorithms*

If we look from the fairness of service point of view, we see that popularity-based algorithms suffer badly from providing fair service to both hot and cold data items equally (Figure 3). To analyse the fairness problem, we adopt the following two performance metrics:
Average fairness. It is the ratio of the total number of different data items broadcast to the total number of different data items requested by the vehicles. Here data items mean both hot and cold data items. Higher average fairness means the system is FF.

Request served percentage. It is the percentage of the number of served requests following either access pattern (the skewed or uniform) to the total number of requests generated 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 the SDRSP of 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 represent the successfully served hot and cold data item requests, respectively, which were pending.

Figure 3 exhibits the fairness of service to both types of (hot and cold) data items of different existing on-demand scheduling algorithms for equal number of requests generated from skewed and uniform item access patterns. From Figure 3(a), we see that when the 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 uniformly distributed pending requests for cold data items and a higher number of skewed pending distributed requests 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 (Figure 3a). However, as SSTF broadcasts data items based on the data item size and ignores the item access pattern, it has the best overall fairness performance among these algorithms.

Figure 3  Fairness measurement of different existing on-demand scheduling algorithms. (a) Average fairness; (b) SDRSP versus UDRSP in DSIN; (c) SDRSP versus UDRSP in MRF; (d) SDRSP versus UDRSP in EDF; (e) SDRSP versus UDRSP in FCFS; (f) SDRSP versus UDRSP in SSTF.
On striking the balance between the fairness of service and throughput

Figure 3  
Fairness measurement of different existing on-demand scheduling algorithms.  
(a) Average fairness; (b) SDRSP versus UDRSP in DSIN; (c) SDRSP versus UDRSP in MRF; (d) SDRSP versus UDRSP in EDF; (e) SDRSP versus UDRSP in FCFS; (f) SDRSP versus UDRSP in SSTF (continued)

Figures 3(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, all the algorithms have larger SDRSPs’ portions than UDRSPs. This is because MRF and DSIN serve more skewed distributed requests. EDF serves requests according to the request deadline, and many popular data items have pending requests. Among these requests, some have critical deadlines. Satisfying these types of request means it serves indirectly popular data items; hence, EDF serves many skewed distributed requests. FCFS serves based on request arrivals; for skewed distributed requests, most of the requests for some selective hot data items are pending. Hence, by broadcasting a data item requested by an earlier arrived 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 (Figures 2 and 3), we conclude that DSIN has the best system performance but suffers from providing fair service to 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 FF scheduler which also can achieve a higher overall system performance. As we can see, none of the existing scheduling
algorithms can achieve two performance objectives together, namely achieving higher fairness of service and gaining higher system throughput. So, it is imperative to devise a new solution.

4.3 Proposed Fairness Friendly (FF) approach

We propose an FF approach which integrates the underlying scheduling algorithm to choose a request for the next broadcast and also to ensure that fair service is provided to both hot and cold data items. The pseudocode of our algorithm is shown in Algorithm 1. Initially, all the data item status is initialised to False (Line 4). According to the used scheduling algorithm, if the selected data item status is found to be False, that data item will be broadcast and then its status will be changed to True. Consequently, all the pending requests’ status for that data items will also be updated. If the selected data item status is found to be True, the selected item data will not be broadcast; rather scheduling algorithm will be invoked again to find the next prioritised data item (Lines 7–14). This mechanism ensures that irrespective of cold or hot data items, all the requested data items get a chance to be broadcast. This is one of our objectives, namely ensuring fairness of service to the data items. An already broadcast data item (that means the status of that data item changed to True) also will get a chance to be broadcast when any new request arrives in the system querying for that item (Lines 16–19). Unlike Round Robin (RR) algorithm, the proposed FF approach does not just broadcast a data item by turns; it rather broadcasts a data item according to the used scheduling algorithm, which ensures an increasing system throughput, which is the second objective of this paper. The complexity of the proposed FF approach is $O(n)$, as the approach needs to examine maximum $n$ data items for $n$ number of pending requests. Hence, the proposed approach has linear complexity and is practical to implement.

Algorithm 1 Fairness Friendly Approach

Require: Set of pending requests generated from vehicles;
Ensure: Both fairness of Service and system throughput are maximised;
1. 
2. RQ denotes the service queue of an RSU server which holds the pending requests;
3. $ID_i^u$ denotes the broadcast status of data item $ID_i$;
4. Initialise $\forall ID_i^u \in DATABASE$ to FALSE;
5. 
6. while $RQ \neq \emptyset$ do
7. Find a data item $ID_i$ using the adopted scheduling algorithm;
8. if $ID_i^u = FALSE$ then
9. Broadcast $ID_i$; /* To ensure the achieving system throughput.*/
10. Update $ID_i^u \leftarrow TRUE$;
11. Update the pending status of all the requests in $RQ$ for $ID_i$;
On striking the balance between the fairness of service and throughput

12. else
13. Reschedule and find a data item other than $ID_i$; /* To ensure the achieving fairness of service.*/
14. end if
15.
16. if While broadcasting if a new request $R_j$ arrives then
17. Insert $R_j$ into $RQ$;
18. Initialise $ID_j ←$ FALSE ;
19. end if
20. end while

From our previous analysis in Sections 4.1 and 4.2, we have found that although DSIN achieves the best throughput among the existing on-demand scheduling algorithms, it cannot provide fair service to the data items. We add our proposed FF approach with DSIN (as a scheduling algorithm), and call it FFDSIN, with the objective of providing both higher fairness of service and higher system performance. To demonstrate the superiority of the FFDSIN algorithm along with the already described on-demand scheduling algorithms (Section 3.3), we also analyse the performance of data item level RR algorithm. The 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.

4.4 System performance and fairness comparison of FFDSIN algorithm with other algorithms

Figure 4 shows the performance of FFDSIN algorithm along with other existing on-demand scheduling algorithms for an equal number of requests generated from both skewed and uniform data item access patterns. The 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, throughput and response time, as shown in Figures 4(a)–(c). Regarding all these performance metrics, DSIN may achieve a little bit better performance than FFDSIN, because DSIN does not care to provide fair service to all the pending requests; it rather 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 maximising 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, if an item has already been broadcast, FFDSIN will broadcast one of the starved data items and not the already broadcast data item to ensure fairness of service.
Figure 4 Performance comparison of FFDSIN algorithm with different exiting on-demand scheduling algorithms. (a) Deadline Missed Ratio (DMR); (b) average throughput; (c) average response time

Figure 5 exhibits the superiority of the FFDSIN algorithm in terms of achieving higher fairness of service over other on-demand scheduling algorithms. It achieves the best fairness performance among all the scheduling algorithms (Figure 5a). Figure 6(b) shows that FFDSIN has equal or nearly equal value of SDRSP and UDRSP, which means FFDSIN broadcasts both hot and cold data items equally. This is the reason why FFDSIN can have higher fairness of service. 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 Figure 5(b), we see that while 100% request is generated from the skewed access pattern (and 0% from the random access pattern), both DSIN and FFDSIN can achieve a higher fairness. However, with decreasing skewed traffic and increasing uniform traffic, both algorithms show decreasing fairness. This is because, with increasing uniform traffic, the number of pending requests for cold data item increases, which results in decreasing sharable requests. Hence, an RSU has to satisfy higher number of cold 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, system average fairness decreases. Nevertheless, FFDSIN can retain its FF superiority better than DSIN under different mix traffic for its FF characteristics (Figure 5b). RR can achieve neither a higher system performance (Figure 4a) nor a better fairness (Figure 5a), because it starts broadcasting data items from the front side of the database. As in the skewed data item access patterns, the majority of the requests access the front data items of the database, when RR is 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 a better overall system performance (Figure 4a). For the same reason, RR has greater portion of SDRSP than UDRSP (Figure 6a) and lower average fairness (Figure 5a).

Figure 5  Fairness comparison of FFDSIN algorithm. (a) FFDSIN versus other algorithms; (b) FFDSIN versus DSIN

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 a nearly equal overall system performance of DSIN. However, it has distinguishable better fairness performance than all the algorithms including DSIN and RR.

Figure 6  Fairness comparison of RR and FFDSIN algorithms. (a) SDRSP versus UDRSP in RR; (b) SDRSP versus UDRSP in FFDSIN
5 Conclusions

In this paper, we formulate an RSU-based VANETs model and analyse both the system throughput and fairness of service of different existing on-demand scheduling algorithms. We find that the scheduling algorithm DSIN has a higher system throughput but suffers from achieving a decent fairness of service. In contrast, the SSTF algorithm shows higher fairness of service but fails to achieve a higher system throughput. To achieve these two performance metrics in one algorithm, we propose an FF approach. After applying FF in DSIN (FFDSIN), the fairness problem of the original DSIN algorithm is alleviated, i.e. FFDSIN strikes a balance between achieving higher fairness of service and gaining higher overall system performance. We perform a series of simulation experiments to demonstrate the superiority of FFDSIN 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 to that of DSIN, which is the best algorithm among all the existing algorithms.

References

On striking the balance between the fairness of service and throughput


