Data Scheduling for Vehicular Ad-hoc Networks Augmented with Road Side Units

Yiqing Gui and Edward Chan

***Abstract*—Timely data dissemination to vehicles is a core function in Vehicular Ad-hoc Networks (VANETs), and the use of Road Side Unit (RSU) has been proposed by researchers to augment vehicle to vehicle communication. RSU can serve as data buffer point from which vehicles can request to access data items stored there. When scheduling vehicle to RSU data access, the mobility pattern of vehicles can be made use of to facilitate multi-RSU data dissemination. In this paper, we propose a motion prediction based scheduling scheme for vehicle to RSU data access which enables cooperative work among a set of Road Side Units. Vehicle requests can be transferred and thus balanced among a group of RSUs, which helps to result in better scheduling performance. Simulation results show that our scheme outperforms simple scheduling schemes under various circumstances. We also examined the extension of our scheme to the case of multi-item requests.**

***Index Terms*—Mobility model, road side unit, simulation, scheduling, vehicular ad-hoc networks**

I. Introduction

Vehicular Ad-Hoc Networks (VANETs) is receiving increasing research focus these days. Timely data dissemination to vehicles is a core function in VANETs, and the use of Road Side Unit (RSU) has been proposed by researchers to augment vehicle to vehicle communication. An RSU is typically installed at busy road intersections, serving as data buffer point so that vehicles can upload and download data items when they pass through the coverage area of the RSU [1]. Location and time dependent data can be temporarily stored at RSUs, which can help to provide services including location-based advertisement, real-time traffic notification, and digital map downloading and so on.

Vehicle to roadside communication has its own characteristics due to the fast moving speed of running vehicles. Requests in vehicle-roadside data access has strict time constrains in nature since they must get completely served before their issuing vehicles leave the transmission range of the RSU. An efficient scheduling scheme called the D*S/N scheme is proposed in [1] to address this challenge. However, only one single RSU is considered. In real life, a group of RSUs are required to enable effective data dissemination in VANETs [2]. Hence, it is necessary to design a new scheduling scheme which works with multiple RSUs.

Manuscript received November 27, 2011; revised January 3, 2012.

This work is supported by a City University of Hong Kong Strategic Research Grant (Project No. 7002702).

Yiqing Gui and Edward Chan are with the Department of Computer Science, Faculty of Science and Engineering, City University of Hong Kong, Kowloon, Hong Kong. (e-mail: yiqinggui2@student.cityu.edu.hk); (e-mail: csedchan@cityu.edu.hk).

In this paper, we propose a cooperative scheduling scheme called the Motion Prediction Optimization (MPO) scheme to enable multi-RSU cooperative data dissemination. In MPO, vehicle requests can be transferred among a group of RSUs based on certain conditions, which helps to balance work load among RSUs as well as to better utilize overall RSU bandwidth. Finally we explore how our work can be extended to the case of multi-item requests and present some results to show the effectiveness of our schemes.

The rest of this paper is organized as follows: Section II summarizes the related work. Section III provides background information including our system model and performance metric. The proposed MPO scheme is elaborated in Section IV. In the same section, MPO is extended for the case of multiple data items and a new scheme called MMPS is presented. Section V presents simulation results where the performance of the MPO scheme and three other classical schemes are compared. The performance of MMPS is also studied. The paper concludes with Section VI.

II. Related Work

Vehicle-roadside data access has received much attention in research [3], [4].

The idea of the deployment of specialized but simple and inexpensive RSUs was proposed in [2]. The authors found that by installing a small number of inter-connected RSUs in a city area, the dissemination performance can be increased dramatically for VANETs.

The placement scheme of RSUs was considered in [5]. An RSU placement scheme was designed to improve the connectivity for a given number of RSUs on the road network of a Korean city.

An efficient scheduling scheme specially tailored for vehicle-roadside data access involving one RSU and many vehicles is proposed in [1]. In this model, vehicles listen to the communication channel once they enter the transmission range of an RSU. Vehicles are allowed to either upload data or download data from RSU and both operations compete for the same bandwidth. Download broadcasting is utilized to yield better scheduling performance. However, only one single RSU is considered in this work. Similarly, the issue of multiple-item requests has also been considered by a number of researchers [6], however the work is once again for a single server and not applicable to multiple broadcast servers (RSUs) with cooperation. To the best of our knowledge this paper is the first work on cooperative scheduling among multiple RSUs based on motion prediction for both single and multiple data item requests.

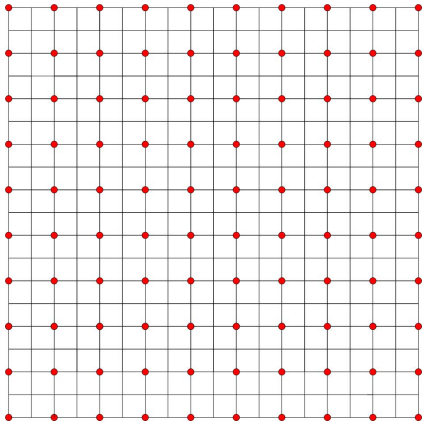

Fig. 1. System model

III. Background

A. System Model

We model part of a city area by a 19 by 19 square road grid (Fig. 1). Each road contains two lanes and vehicles are allowed to move in either direction. We assume that the distance between each two roads is *D* meters. RSU servers are placed at road intersections once every other roads (black spots). So there are totally 100 RSUs serving vehicles passing by. The transmission radii of RSU servers are set to be less than *D* meters. As a result, there exists some area in the grid region which is not covered by the transmission range of any RSU.

Vehicles listen to the transmission channel when they enter the transmission range of an RSU. We assume that a vehicle generates request only when it is within the transmission range of a certain RSU. When driving through regions that are not covered by any RSU, no requests are generated by a vehicle. Request can either be a downloading one or an uploading one and each contains a deadline. We also assume that the RSUs serve vehicle requests non-preemptively.

The movement pattern of vehicles within the square region follows the *Manhattan Mobility Model* [7].

During each scheduling round, the RSU will scan through the whole set of waiting requests and pick the one with the highest priority to serve based on the scheduling scheme. Besides, requests that are not likely to be served before their deadline will either be deleted or transferred depending on certain conditions. These conditions may involve the workload of nearby RSUs, the velocity of the vehicle that generates the request, as well as the request deadline. The details will be specified in the next section in which the details of our algorithm are discussed. Each RSU will try to transfer the transferrable request to one of its nearby RSUs based on the motion prediction of the vehicle. Motion prediction is based on the probabilistic mobility nature of the *Manhattan Mobility Model*. If motion prediction is correct, the transferred request will have a chance to be served after its issuing vehicle notified its arrival to the RSU where the request is transferred to. A transferred request will be considered in each scheduling round only after its issuing vehicle notifies its arrival to the RSU where the request is transferred to. We also assume that no transfer overhead will be incurred, and requests can be transferred instantly among RSUs with no delay. By employing such a cooperative scheduling scheme, workload can be balanced if certain RSU server suffers from overloading and the overall RSU bandwidth can be better utilized. As a result, there is expected to be increase in overall scheduling performance.

We also set some real life deadlines for each request rather than assuming it to be the time when its issuing vehicle leaves the RSU transmission range.

A non-transferred request is considered successfully served if it is completely served on or before its deadline. For a transferred request to be considered successfully served, the following two conditions need to be satisfied. Firstly, the motion prediction should be correct, say the vehicle should enter the transmission range of the RSU where its request is transferred to. Secondly, the request must be completely served within the time interval from the time the issuing vehicle enters the transmission range of this RSU to the time it leaves.

B. Performance Metric

We will use *service ratio* as our performance metric. It is defined as the ratio of the total number of successfully served requests to the total number of requests that are generated by all the vehicles. Obviously, the higher the service ratio, the better the algorithm performs.

IV. The Proposed Scheduling Schemes

To make use of the mobility pattern of the *Manhattan Mobility Model*, we proposed the MPO scheme, which enables cooperative work among a set of inter-connected RSUs.

The MPO scheme is based on the D*S/N scheme proposed in [1].

A. Notations and Terminology

1) *request*: A request is defined as a 9-tuple: <data_size, x_cor, y_cor, generation_time, valid_time, deadline, direction, duration, finish_time>, where
   a) *data_size*: the size of the data item that is requested by the vehicle
   b) *x_cor*, *y_cor*: the coordinates on the map where the request is generated. The lower left corner of the square region is set to be the origin.
   c) *generation_time*: the time at which the request is generated by the vehicle
   d) *valid_time*: the period of time after the generation of the request, within which the request is valid
   e) *deadline*: the deadline of the request, which is equal to *generation_time* plus *valid_time*
   f) *direction*: the moving direction of the vehicle generating the request. The direction can be one of the following four values: North(N), South(S), West(W) and East(E).
   g) *duration*: the total service time of a request, which is equal to *data_size* divided by the bandwidth of the RSU server
   h) *finish_time*: the RSU will sort the request set in the waiting queue based on the scheduling scheme. For the first request in the waiting queue after sorting, the finish_time is equal to its *duration*. The finish_time of the second request is defined as the finish_time of the first request plus the *duration* of the second request. The

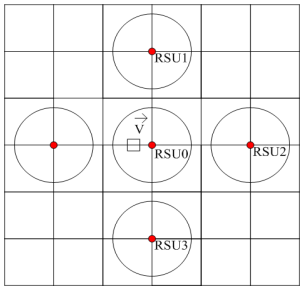

Fig. 2. Scenario A

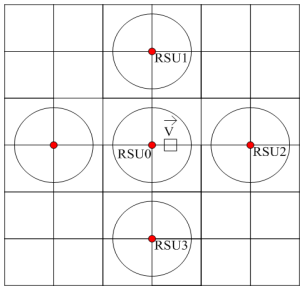

Fig. 3. Scenario B

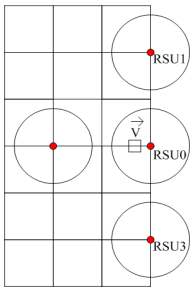

Fig. 4. Scenario C

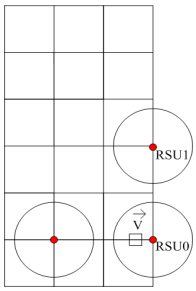

Fig. 5. Scenario D

finish_time of the following requests can be calculated by this recurrence relation. It can be seen from the definition that the finish_time of the request gives an approximation by what time the request will be completely served at the RSU under the current workload.

2) *qualified request*: A request is said to be qualified if the *finish_time* of this request is less than or equal to its *deadline*. If not, the request is said to be *unqualified*. Qualified requests are likely to be served completely on or before their deadline because their *finish_time* is earlier than *deadline*. In other words, they will be successfully served according to their current positions in the waiting queue of the RSU where they are queuing. Requests that are unqualified are likely to miss their *deadline* since they are scheduled to be served in a late time.
3) *RSU_available_time*: it is the biggest *finish_time* of all the *qualified* requests in the waiting queue of an RSU.
4) *RSU_x*: the x coordinate of the RSU in the map
5) *RSU_y*: the y coordinate of the RSU in the map

B. D*S/N Scheme

In the *D*S/N* scheme, each request is given a value called the *DSN_value* defined as:

$$DSN\_value = \frac{(deadline - current\_time) \times data\_size}{Number}$$

The *Number* attribute is equal to the number of downloading requests that request for the same data item if the request is a downloading one. For uploading requests, this attribute is always 1.

The *D*S/N* algorithm computes *DSN_value* for each request in the waiting queue. It sorts the requests by their *DSN_value* and the request with the minimum *DSN_value* is chosen to be served first. Requests that miss their *deadlines* are discarded during each scheduling round. This scheme takes both *data_size* and *deadline* into consideration. It also takes advantage of download broadcasting, where a single broadcast can serve the whole set of downloading requests that request for the same data item.

C. MPO Scheme

The MPO scheme is based on the *D*S/N* scheme, in which the RSU scans through the whole set of the requests and choose the qualified request with the smallest *DSN_value* to serve first during each scheduling round. What's different is that requests that are likely to miss their *deadline* are attempted to be transferred in MPO. The RSU will transfer *unqualified* requests to nearby RSUs based on the motion prediction of their issuing vehicles if the situation falls in one of the following scenarios.

Scenario A: $x\_cor \le RSU0\_x$ for a given request R

The *unqualified* request is generated by a vehicle moving towards east within the transmission range of RSU0 and to the left of RSU0 (Fig. 2). In this case, MPO will check the *RSU_available_time* of RSU2 first since the vehicle will have the highest probability to move in this direction under the *Manhattan Mobility Model*. If the predicate $RSU2\_available\_time + duration \le deadline$ holds, then this request will be transferred to RSU2. We choose this predicate to ensure that the request *deadline* is relative long so that there will be a smaller probability for the *deadline* to be missed even before the issuing vehicle arrive at RSU 2. This predicate also helps to reduce the probability that the transferred request will cause overloading problem for RSU 2 since the request can be successfully served even if it is queued after the last qualified request in the waiting queue according to the current workload of RSU2.

If this inequality does not hold, the *RSU_available_time* of RSU1 and RSU3 will be checked. The request will be transferred to one of these RSUs if the above inequality holds for either of them. If both of the RSU1 and RSU3 satisfy the condition, MPO will randomly pick one of them. If none of them satisfies the inequality, the request will be discarded.

Scenario B: $x\_cor > RSU0\_x$ for a given request R

The *unqualified* request is generated by a vehicle moving towards east within the transmission range of RSU0 and to the right of RSU0 (Fig. 3). In this case, MPO will check the *RSU_available_time* of RSU2 only. If the predicate $RSU2\_available\_time + duration \le deadline$ holds, then this request will be transferred to RSU2. If not, the request will be discarded.

Scenario C: RSU is on the edge for a given request R

The *unqualified* request is generated by a vehicle moving towards east within the transmission range of RSU0 and RSU0 is on the edge (but not on the corner) (Fig. 4). In this case, MPO will check the *RSU_available_time* of both RSU1 and RSU3. The request will be transferred to one of these RSUs if the same inequality defined in the above scenario holds for either of them. If none of them satisfies the inequality, the request will be discarded.

Scenario D: RSU is on the corner for a given request R

The *unqualified* request is generated by a vehicle moving towards east within the transmission range of RSU0 and RSU0 is on the corner (Fig. 5). In this case, MPO will check the *RSU_available_time* of RSU1 only. If the predicate $RSU1\_available\_time + duration \le deadline$ holds, then this request will be transferred to RSU1. If not, the request will be discarded.

Finally, if none of the above four cases are met, the request will be deleted.

Note that in each of the above four scenarios, the corresponding north, south and west cases can be derived similarly

by symmetry.

A transferred request is stored in a separate buffer at the target RSU, and it will be moved to the waiting queue of the RSU only when its issuing vehicle arrives at the transmission range of the target RSU.

To address the problem of motion prediction failure, where the transferred request will be stored in the RSU buffer forever, the transferred request will be deleted from the buffer when its *deadline* is missed.

D. MMPS Scheme

We consider multi-item vehicle request scheduling in this section. The following problem needs to be considered when designing scheduling scheme for multi-item requests in vehicle-roadside data access. Multi-item requests usually contain a mixture of popular and non-popular data items and this unique characteristic needs to be considered when making scheduling decisions [8]. Communication from RSU to vehicles is broadcast in nature. As a result, one RSU broadcast is able to serve multiple vehicle requests at the same time, which could potentially help to better utilize the limited bandwidth of RSU. We call this download broadcasting in this paper. Hence, popular data items should be given high precedence when making scheduling decision since a bigger number of vehicles requests can be served by broadcasting popular data items. On the other hand, vehicle requests may contain non-popular data items as well. Giving high precedence to popular data items will leave requests with non-popular data items to wait indefinitely since new requests with popular data items continues to arrive. Poor performance will be resulted because many request deadline will be missed due to excessive wait for the non-popular items. This is known as the *request starvation* problem [8]. Hence, both data popularity and request deadline need to be considered when making scheduling decisions.

Before discussing the design strategy and our proposed multiple-item scheduling scheme, we need to introduce a number of new definitions:

1) *Request*: A request *req* is a 4-tuple defined as *req* = < *deadline*, *unserved_set*, *unserved_size*, *scheduled_finish _time*>, where
   a) *deadline*: It is the deadline of a vehicle request.
   b) *unserved_set*: It *is* a set that contains all the data items that are requested but not received by the vehicle. The *unserved_set* contains all the data items of a vehicle request when a request is just generated. Later, members of the *unserved_set* will be removed gradually if they are broadcast by the RSU. All the data items within the *unserved_set* should be served on or before request *deadline*. Otherwise, the request is considered fail to be served.
   c) *unserved_size*: It is the sum of the size of all the data items that belong to the *unserved_set*. This quantity measures how long the request still needs to be served.
   d) *scheduled_finish_time*: It is the time at which all the data items of a request are to be received by the vehicle according to the current workload of the RSU and the scheduling scheme used. Even though new requests will keep on coming, which will change the workload and thus the scheduling decision of the RSU server, this quantity can give an approximation by which time the request will be completely served.
2) *Request Valid Time*: The period of time in between request generation and request *deadline* is defined as the request valid time. The relationship of request *deadline*, request valid time and request generation time is expressed as: *request generation time* + *request valid time* = *deadline*. A request is considered *invalid* if its *deadline* is missed.
3) *Request Waiting Queue*: For one particular RSU *R**, its request waiting queue contains all the requests that are received but not completely served by RSU *R**. It is denoted as *Q(R*)*
4) *Request Priority*: The request priority of one particular request *req* is denoted as *RP*(*req*) where
   $RP(req) = unserved\_size \times (deadline - current\_time)$
5) *Data Priority*: For a particular data item *d*, its data priority in RSU *R** is denoted as *DP*(*d*), where
   $DP(d) = |\{req: req \in Q(R^*) \wedge d \in unserved\_set\}|$
   It is the number of requests stored at RSU *R** which contains *d* in their *unserved_set*
6) *RSU_available_time*: The RSU_available_time for an RSU equals to the biggest *scheduled_finish_time* of all the waiting requests in *Q(R*)*
7) *Request Slack Time*: The Request Slack Time is defined as: *deadline – current_time*

We propose a new scheduling scheme based on the following four strategies to achieve better scheduling performance for scheduling multi-item requests under given RSU bandwidth.

a) For each data item *d* which belongs to the *unserved_set* of a request, the data item with the highest *data priority* should be served first to allow more download broadcasting since the biggest number of requests contain this data item in their *unserved_set*.

b) Given two requests with the same *deadline*, the request with the smaller *unserved_set* should be served first. Request with smaller *unserved_size* can be completely served in a shorter time, which results in a higher probability for the request to get served completely before its *deadline*. The *request starvation problem* is alleviated as well because request with non-popular data item remain unserved with also be given high scheduling precedence due to its small *unserved_size*.

c) Given two requests with the same *unserved_size*, the request with smaller *slack time* should be served first since it is more urgent.

d) If one request is not likely to be served successfully by the RSU where it is currently queuing, say the *scheduled_finish_time* of the request is bigger than its deadline, the request can be considered for transfer to a nearby RSU with a lighter workload so that the probability for this request to be successfully served will increase provided that the issuing vehicle pass through the target RSU later. Essentially this technique aim to perform load balancing based on motion prediction.

Based on the above four strategies, we propose a cooperative scheduling scheme called the Multi-item Motion Prediction Scheduling Scheme (MMPS) which enables workload transfer among a group of RSUs for scheduling multi-item requests in vehicle to roadside data access and it is

summarized in Algorithm 1 and Algorithm 2.

Algorithm 1 Select-Data-Item

```
Request req* ← Q(R*)[1]
for i ← 2 to |Q(R*)| do
    if RP(req*) > RP(Q(R*)[i]) then
        req* ← Q(R*)[i]
    end if
end for
Data Item d* ← req*.unserved_set[1]
for i ← 2 to |req*.unserved_set| do
    if DP(d*) < DP(req*.unserved_set[i]) then
        d* ← req*.unserved_set[i]
    end if
end for
return d*
```

Algorithm 2 MMPS

```
for i ← 1 to |Q(R*)| do
    if Q(R*)[i] is invalid then
        Delete Q(R*)[i] from Q(R*)
    end if
end for
Compute the scheduled_finish_time of each waiting request in Q(R*) using Algorithm 1 Select-Data-Item as the scheduling scheme
for i ← 1 to |Q(R*)| do
    if Q(R*)[i]. scheduled_finish_time > Q(R*)[i]. deadline then
        if Q(R*)[i] satisfies the requirement to be transferred to the RSU where the vehicle is most likely to visit next then
            Transfer Q(R*)[i] to this RSU
        else if Q(R*)[i] satisfies the requirement to be transferred to the RSU where the vehicle is second most likely to visit next then
            Transfer Q(R*)[i] to this RSU
        else
            Delete Q(R*)[i] from Q(R*)
        end if
    end if
end for
Data Item d* ← Select-Data-Item
Broadcast Data Item d*
```

V. Performance Evaluation

A. Experiment Setup

We have built a C-SIM [9] based simulator to evaluate the performance of our proposed scheduling scheme and three other schemes including the First Come First Serve (FCFS) scheme, the Smallest Datasize First (SDF) scheme and the Earliest Deadline First (EDF) scheme.

The distance between two roads is set to be 400 meters. So the whole simulation area becomes a 7200m by 7200m square grid with 100 RSUs installed at road junctions once every other roads.

Vehicle movement pattern follows the *Manhattan Mobility Model* with the intra-vehicle and inter-vehicle relationship defined below:

1) $V(t+1) = V(t)$ + Normal Dist. (μ, σ), where V(t) is the velocity of one vehicle at time slot t

2) The distance between two vehicles running on the same street in the same direction is at least *Safety Distance* meters.

Vehicles can generate either downloading or uploading requests during the simulation. The request generation time follows the exponential distribution with mean value *T_request* seconds for each node. The data access pattern is based on the *Zipf* distribution [10], which is commonly used to model data access pattern in Mobile Ad Hoc Networks [1][11]. In *Zipf* distribution, the probability of accessing the i^{th} data item is represented as:

$$P(i) = \frac{1}{i^{\theta}\sum_{k=1}^{n} 1/k^{\theta}}$$, where n is the number of data items

When $\theta = 1$, it is strictly *Zipf* distribution. When $\theta = 0$, it becomes uniform distribution. The θ value is chosen to be 0.8 in our model according to studies on real web trace [12].

Request valid time also follows the normal distribution.

Each RSU server has its own waiting queue. They run scheduling algorithm to serve vehicle requests and transfer work load among them during the simulation. Each RSU server has access to the full set of data items. We assume that the data set which can be accessed by vehicles remains fixed. Vehicles can only upload and download data item from a fixed list of data items that are stored in the RSU. They are not allowed to add new entries into this data set.

We also assume no background load for servers. Requests are allowed to contain multiple data items. The number of data items in one request is uniformly distribution between 1 and 4. The valid time of requests follow the normal distribution with μ=15s and σ=2s.

Simulation parameters and their default values are listed in Table I. In Sections B to F below we first study the performance of the MPO scheme which works with single-item request. In Section G we will explore the performance of MMPS which is capable of scheduling multi-item requests.

B. The Effect of Request Valid Time

It is shown in Fig. 6 that in general, all four schemes performs better when request valid time increases since it is more likely for each request to be served before its *deadline* when it is valid for a longer time. And the MPO scheme outperforms other three scheduling schemes under various request valid time as is expected.

TABLE I. Simulation Parameters

Parameter	Default Value
Simulation Time	900s
RSU Bandwidth	625 KB/s
Vehicle Velocuty	V(0) = 15m/s, μ = 0, σ = 0.03
Max. Vehicle Speed	20m/s
Min. Vehicle Speed	12m/s
RSU Trans. Radius	300m
Data Size	50K ~ 5M
Data Set Size	25
Zipf Parameter theta	0.8
Inter-arrival Time between Requests	Mean = 5s
Request Valid Time	μ = 15s, σ = 2
Number of Vehicles in the Simulation	80
Safety Distance	2m

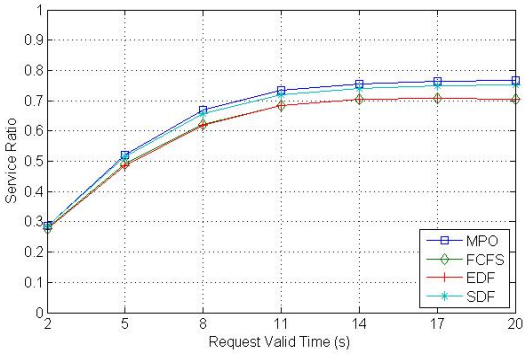

Fig. 6. Effect of request valid time

C. The Effect of Request Inter-Arrival Time

When request inter-arrival time decreases, more requests will be generated within a certain period of time, which results in heavier RSU workload. We can see from Fig. 7 that the performance of the MPO scheme degrades more slowly than other three schemes when RSU workload increases.

This result shows that work load transfer enables the MPO scheme to serve more requests within a certain period of time.

D. The Effect of Data Access Pattern

The effect of data access pattern is shown in Fig. 8.

We can see from the picture that the service ratio of MPO grows faster than that of the other three scheduling schemes when theta increases. The skewness of data access pattern becomes larger as theta grows. As a result, vehicles will tend to request for the same few data items when theta is large. The effect of download broadcasting is hence increased at the same time since more requests will be served simultaneously by a one single broadcast. Because MPO utilizes download broadcasting to achieve better scheduling performance, it achieves a higher service ratio when theta grows.

E. The Effect of Vehicle Speed

As is shown in Fig. 9, the service ratios of all four schemes

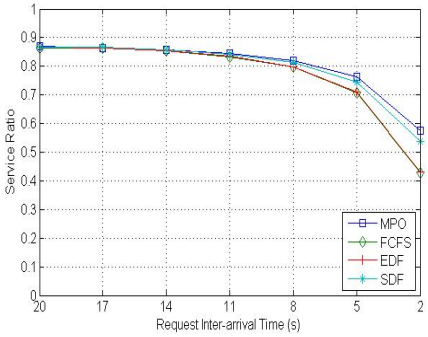

Fig. 7. Effect of request inter-arrival time

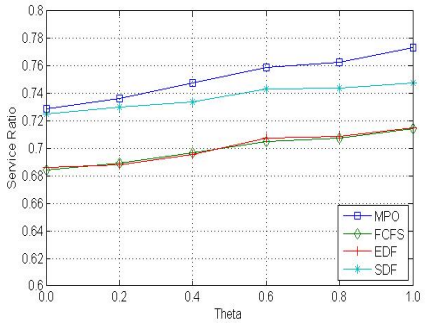

Fig. 8. Effect of data access pattern

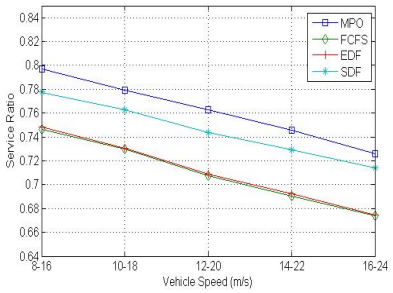

Fig. 9. Effect of vehicle speed

Decrease as vehicles run faster on the road. This is because vehicles will stay within the transmission range of a certain RSU for less time when they move at higher speed on the road. RSU server will thus have less time to serve requests before their issuing vehicles leave, which lowers down the service ratio.

F. The Hotspot Effect

When driving in downtown area, people may be attracted by the breaking news shown on the huge screen at busy cross roads. Many of them may try to learn more about the news by accessing data through nearby RSUs. Similarly, when there is a big jam on the road, many drivers will try to know what has happened and will try to find out which way to go at the next crossroads to get out of this jam by accessing real time traffic data via RSUs along the road. As a result, vehicles may generate many more requests within a specific region than they will in other areas. We call these specific regions hotspots.

Hotspots are common in real life and in this section we evaluate the performance of the MPO scheme under the effect of hotspots. The major idea underlies the MPO scheme is the motion prediction based work transfer. Under this optimization, requests that are *unqualified* in one RSU may have a chance to be served successfully by nearby RSUs.

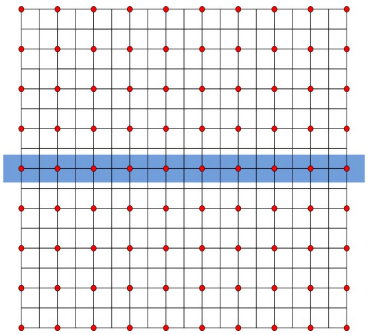

Fig. 10. Line distribution of hotspot rsus

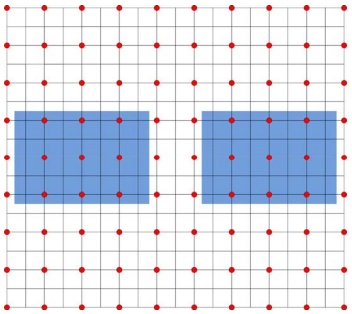

Fig. 11 Two cluster distribution of hotspot rsus

Under hotspot effect, the MPO scheme is expected to perform even better than other three schemes. With the presence of hotspots, RSUs will have a much bigger difference in workload so that overloaded and non-overloaded RSUs will scatter in the region. Request transfer will have a higher chance to be successful since it is more likely to find non-overloaded RSUs around overloaded RSUs. But the other three schemes cannot benefit from this change.

To verify this, we modified our simulation model in the last section. We select some of the RSUs to be hotspot RSUs and we set the request inter-arrival time within the region that is 100m from hotspot RSUs to be much lower than that of other regions. So when vehicles move across hotspot RSUs, They will generate a lot more requests compared to

non-hotspot regions.

We examine the performance of the MPO scheme and compare it with EDF, FCFS, SDF under two different location distribution of hotspot RSUs, the line distribution and two cluster distribution (Fig. 10 and 11). Hotspot RSUs are shaded.

Simulation parameters are the same as in Table 1 except that the inter-arrival time between requests in non-hotspot region is set to be 50s but only 0.5s for hotspot region.

As shown in Fig. 12 and 13, MPO performs the best in both two location distribution of hotspot RSUs. The performance difference between MPO and other three schemes is larger compared to Fig 6. as expected. Hence, the MPO does perform even better under the hotspot effect.

Comparing Fig. 12 and 13, it is interesting to see that the performance gap is larger when hotspots RSUs are distributed in line shape. This can be explained by the ratio between the total number of non-hotspots RSUs which are next to hotspot RSUs to the total number of hotspot RSUs. The bigger the ratio, the more likely the work transfer will be successful since the chance for finding a nearby non-overloaded RSU will be higher. As mentioned before, MPO leverages on successful work transfer to achieve good performance. Hence, the performance gap will increase as the ratio increases.

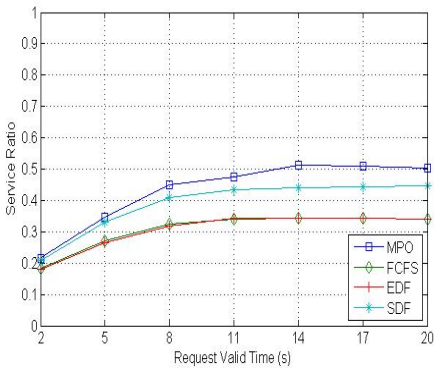

Fig. 12 Simulation result of the line distribution

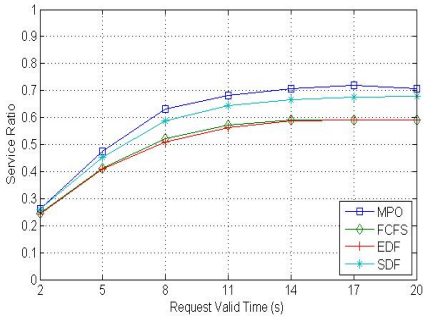

Fig. 13 Simulation result of the two cluster distribution

G. Performance of MMPS

In this section, we will briefly explore the performance of the proposed MMPS scheme.

Figure 14, 15 and 16 illustrates the effect of request inter-arrival time, request valid time and data access pattern respectively on the performance of MMPS. Two classical scheduling schemes, Earliest Deadline First (EDF) and Most Requested First (MRF) are used for comparison in these experiments. Note that data size is set to be between 20k and 2048k when evaluating these three schemes and vehicles are only allowed to download but not upload data.

Figure 14 illustrates the service ratio of the three scheduling schemes under different request inter-arrival time. The three schemes perform similarly under low RSU workload. However, when RSU workload increases, the performance of MMPO degrades slower than that of EDF and MRF, because MMPS utilizes both download broadcasting and workload transfer while avoiding the request starvation problem to serve more requests with the given RSU bandwidth. MRF does benefit from download broadcasting, but it also suffers from the request starvation problem as well since it gives high priority to hot data items solely. So it performs in between EDF and MMPS under high workload. EDF cannot benefit from broadcast at all. It will even serve a request of big size first if the request deadline is the earliest, which will cause other requests to wait for a long time. As a result, EDF performs the worst under small request inter-arrival time.

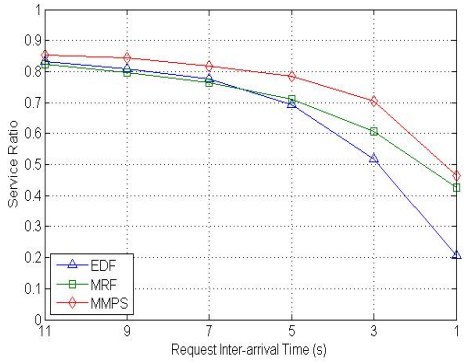

Fig. 14 Effect of request inter-arrival time on MMPS, EDF and MRF

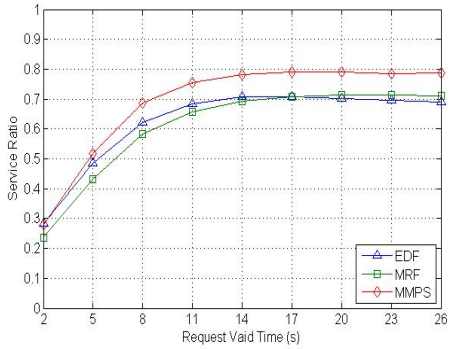

Fig. 15 effect of request valid time on MMPS, EDF and MRF

The effect of request valid time is shown in Fig. 15. As is shown in the picture, the performance of the three schemes increases as request valid time increases and MMPS still performs the best by utilizing request transfer and download broadcasting. It is also interesting to see that the curve of MRF and EDF crosses with each other. When request valid time is short, MRF suffers from request starvation problem and performs the worst. As request valid time increases, request deadline will be lengthened as well. Requests can thus afford to spend more time waiting for their cold data item to be served without missing their deadline, which alleviates the starvation problem to a certain degree. Hence, the performance of MRF increases and finally exceeds EDF when request valid time grows.

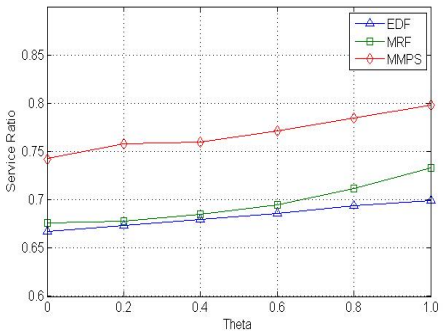

Fig. 16 Effect of data access pattern on MMPS, EDF and MRF

Fig. 16 shows the effect of data access pattern in this section. The service ratio of both MMPS and MRF increases faster than that of EDF when θ grows. This is because both MMPS and MRF make use of download broadcasting to achieve better performance. When theta grows, more requests can be served by broadcast, which enhance the effect of download broadcasting. However, there is no benefit to EDF as it schedules requests solely on the basis of the request deadline. As a result, the skewness of data access pattern has much less effect on EDF compared to MMPS and MRF.

VI. Conclusion

In this paper, we propose a cooperative scheme for scheduling vehicle to RSU data access based on the motion prediction of vehicles. Vehicle mobility pattern is utilized to predict future vehicle position, which helps to achieve more accurate workload transfer among RSUs. Request transfer then enables more requests to be served successfully by balancing RSU workload and better utilizing overall RSU bandwidth. Our proposed scheme performs even better under hotspots effect where requests are generated unevenly among different regions, which is the typical case in real life. Furthermore, we have also demonstrated that a modified version of MPO for multi-item requests also significantly outperform classical scheduling algorithms.

References

[1] Y. Zhang, J. Zhao, and G. Cao, "On scheduling vehicle-roadside data access", in *Proc. of 4th ACM International Workshop on Vehicular Ad Hoc Networks (VANET '07)*, 2007, pp. 9-18

[2] C. Lochert, B. Scheuemann, M. Caliskan, and M. Mauve, "The feasibility of information dissemination in vehicular ad-hoc networks," in *Proc. of Wireless on Demand Network Systems and Services (WONS'07)*, 2007, pp. 92-99

[3] V. Bychkovsky, B. Hull, A. Miu, H. Balakrishnan, and S. Madden, "A measurement study of vehicular internet access using in situ Wi-Fi networks," in *Proc. of 12th International Conference on Mobile Computing and Networking (MobiCom '06)*, 2006, pp. 50-61

[4] K. Liu and V. C. S. Lee, "RSU-based real-time data access in dynamic vehicular networks", in *Proc. of 13th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, 2010, pp. 1051-1056

[5] J. Lee and C. M. Kim, "A roadside unit placement scheme for vehicular telematics networks," in *Proc. of AST/UCMA/ISA/CAN*, 2010, pp. 196-202

[6] K. Liu, V. C. S. Lee, and K. Leung, "Data scheduling for multi-item requests in multi-channel on-demand broadcast environments", in *Proc. 7th ACM International Workshop on Data Engineering for Wireless and Mobile Access*, 2008, pp. 45-54

[7] F. Bai, N. Sadagopan, and A. Helmy, "IMPORTANT: a framework to systematically analyze the impact of mobility on performance of routing protocols for ad hoc networks", in *Proc. of IEEE INFOCOM*, 2003, vol.2, pp. 825 – 835

[8] K. Liu and V. C. S. Lee, "Analysis of data scheduling for multi-item requests in multi-channel on-demand broadcast environments," *Technical Report*, *Computer Science Department*, *City University of Hong Kong*, 2008.

[9] CSIM Simulation Library from *Mesquite Software*: http://www.mesquite.com/

[10] G. Zipf, "Human behavior and the principle of least effort," *Addison-Wesley*, 1949

[11] L. Yin and G. Gao, "Supporting cooperative caching in ad hoc networks," *IEEE Transaction on Mobile Computing*, Jan. 2006, vol. 5, issue 1, pp. 77-89

[12] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and *Zipf*-like distributions: evidence and implications," in *Proc. of IEEE INFOCOM*, 1999, vol. 1, pp. 126-134

Yiqing Gui is currently seeking a master's degree in computer science in City University of Hong Kong. He received his BEng degree in the same university in 2011. His research interests include networking, mobile computing and vehicular ad-hoc networks.

Edward Chan received his BSc ad Msc degrees in Electrical Engineering from Stanford University, and his PhD in Computer Science from Sunderland University. He worked in the Silicon Valley for a number of years in the design and implementation of computer networks and real-time control systems before joining City University of Hong Kong where he is now an Associate Professor. His current research interests include performance evaluation of high speed networks, mobile data management, power-aware computing, and network management.