On Scheduling Vehicle-Roadside Data Access

Yang Zhang
Department of Computer Science & Engineering
The Pennsylvania State University
yangzhan@cse.psu.edu

Jing Zhao
Department of Computer Science & Engineering
The Pennsylvania State University
jizhao@cse.psu.edu

Guohong Cao
Department of Computer Science & Engineering
The Pennsylvania State University
gcao@cse.psu.edu

ABSTRACT

As vehicular networks become popular, more and more people want to access data from their vehicles. When many vehicles want to access data through a roadside unit, data scheduling becomes an important issue. In this paper, we identify some challenges in vehicle-roadside data access. As vehicles move pretty fast, the requests should be served quickly. Also, vehicles may upload data to the roadside unit, and hence the download and upload requests compete for the same bandwidth. To address these challenges, we propose several scheduling schemes. We first propose a basic scheduling scheme called $\mathcal{D} * \mathcal{S}$ to consider both service deadline and data size. We then enhance it by using a single broadcast to serve multiple requests. Finally, we identify the effects of upload requests on data quality, and propose a Two-Step scheduling scheme to provide a balance between serving download and update requests. Simulation results show that the Two-Step scheduling scheme outperforms other scheduling schemes.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Wireless communication

General Terms

Algorithms, Management, Performance

Keywords

RoadSide Unit(RSU), Upload/Download, Data Quality, Scheduling

1. INTRODUCTION

Recently, vehicle-roadside data access has received considerable attention [11, 5, 8, 7]. With RoadSide Unit (RSU) such as 802.11 access point, vehicles can access data stored in the RSU or even access the Internet through these RSUs.

VANET'07, September 10, 2007, Montréal, Québec, Canada.

From 2003, the US Department of Transportation has invested millions of dollars [10] to integrate vehicles and RSUs and to make the current transportation system more intelligent. Chrysler-Daimler has introduced the "InfoFuel" system to provide Mercedes drivers the ability to access wireless data through roadside hotspots [1]. Also, FCC dedicates the 5.9 GHz frequency specifically allocated to vehicle-vehicle and vehicle-roadside communications and enacts the Dedicated Short Range Communications (DSRC) standard in 2004. Later, IEEE develops several standards to ensure that vehicles and roadside infrastructures can communicate with each other. All these efforts in academy, government, and industry make vehicle-roadside data access mature enough to be used in our daily life.

In vehicle-roadside data access, the RSU can act as a router for vehicles to access the Internet. Although this can bring many benefits to the drivers, the deployment cost and maintenance cost are very high. As another option, RSU can also be just used as a buffer point (or data island) between vehicles. In this paper, we focus on the latter paradigm due to its low cost and easy deployment. In this paradigm, all data on the RSUs are uploaded or downloaded by vehicles. For example, some data, especially those with spacial/temporal constraints, only need to be stored and used locally. The following applications also belong to this case where the data are buffered at the RSUs, and will not be sent to the Internet.

1. Value-added Advertisement: Store owners may want to advertise their sale or activity information in nearby area. Without Internet connection [20], they can ask the running vehicles to carry and upload the advertisement information to nearby RSUs. At the same time, other vehicles driving around can download these advertisements and visit the stores.

2. Real-Time Traffic: Vehicles can report real-time traffic observations to RSUs. The traffic data are stored at RSUs, providing real-time query and notification services to other vehicles. The data can be used to provide traffic conditions and alerts such as road congestion and accidents.

3. Digital Map Downloading: Due to the storage limitations of memory card and frequent road construction, it is impossible for vehicles to install all the most up-to-date digital maps before traveling. Hence, vehicles driving to a new area may hope to update map data locally for travel guidance.

Different from traditional data access system in which users can always wait for the service from the data server, vehicles are moving and they only stay in the RSU area for a short period of time. As a result, there is always a time constraint associated with each request. Meanwhile, to make the best use of the RSU and to share the information to as many vehicles as possible, RSUs are often set at the roadway intersections or areas with high traffic. In these areas, download (query) requests retrieve data from the RSU, and upload (update) requests upload data to the RSU. Both download and upload requests compete for the same limited bandwidth. As the number of users increases, which request to serve at which time will be critical to the system performance. Hence, it is important to design an efficient scheduling algorithm for vehicle-roadside data access. In this paper, we design efficient solutions for scheduling vehicle-roadside data access. Our contributions are as follows.

1. We first propose a basic low complexity scheduling scheme called $\mathcal{D} * \mathcal{S}$ which considers both *data size* and *request deadline*.

2. We improve the performance of the basic scheduling algorithm by using *broadcasting* techniques to serve more requests.

3. We study the tradeoffs between *service ratio* and *data quality*, and propose a *Two-Step* scheme to address the tradeoffs between uploads and downloads;

4. We conduct extensive simulations to study the performance of the proposed scheduling schemes.

To the best of our knowledge, our work is the first to consider time constraint and data quality for scheduling vehicle-roadside data access.

The rest of this paper is organized as follows: Section 2 presents the background and necessary preliminaries. Section 3 describes the limitation of three naive schemes and presents a new scheduling scheme called $\mathcal{D} * \mathcal{S}$. In section 4, we study how to optimize scheduling by broadcasting. An improved scheme called $\mathcal{D} * \mathcal{S}/\mathcal{N}$ is proposed. We discuss the impact of data staleness and propose a two-step scheduling scheme in Section 5. Section 6 evaluates the performance of the proposed schemes. Finally, we summarize the related work in Section 7 and conclude the paper in Section 8.

2. BACKGROUND AND PRELIMINARIES

2.1 System Model

As shown in Figure 1, a large number of vehicles retrieve (or upload) their data from (or to) the RSU when they are in the communication range. The RSU (server) maintains a service cycle, which is non-preemptive; i.e., one service can not be interrupted until it finishes. When one vehicle enters the RSU area, it listens to the wireless channel. All vehicles can send requests to the RSU if they want to access the data. Each request is characterized by a 4-tuple: <*v-id*, *d-id*, *op*, *deadline*>, where *v-id* is the identifier of the vehicle, *d-id* is the identifier of the requested data item, *op* is the operation that the vehicle wants to do (upload or download), and *deadline* is the critical time constraint of the request, beyond which the service becomes useless. All requests are queued at the RSU server upon arrival. Based on the scheduling algorithm, the server serves one request and removes it from the request queue.

Different from traditional scheduling services, data access in vehicular networks has two unique features: 1) The arrival request is only active for a short period of time due to vehicle moving and coverage limitations of RSUs. When vehicles move out of the RSU area, the requests not served have to be dropped; and 2) Data items can be downloaded and uploaded from the RSU server. The download and update requests compete for the service bandwidth.

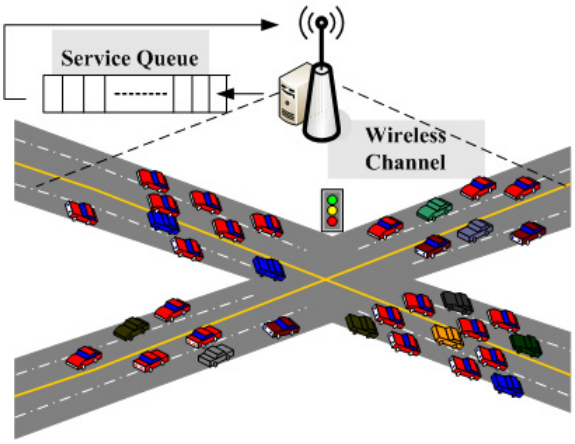


Figure 1: The Architecture of Vehicle-Roadside Service Scheduling

We assume that each vehicle knows the service deadline of its request. This is reasonable because when a vehicle with GPS device enters the coverage area of a RSU, it can estimate its leaving time based on the knowledge of its driving velocity and its geographic position[1].

2.2 Performance Metrics

In most previous work, the metrics for scheduling algorithms are responsiveness (e.g., average/worst-case waiting time [14, 4, 6]) or fairness (e.g., stretch [3, 17]). In most of these works, requests do not have time constraints and the data on the server are not updated or updated only by the server. However, in the vehicle-roadside data access scenario, requests not served within a time limit will be dropped as the vehicles move out of the RSU area. Since update requests compete bandwidth with other download requests, some data may become stale after an update is missed, degrading the service quality. Therefore, compared with responsiveness and fairness, providing fresh data to more vehicles is more important and we use the following metrics for scheduling vehicle-roadside data access.

1) ***Service Ratio***: Service ratio is defined as the ratio of the number of requests served before the service deadline to the total number of arriving requests. A good scheduling scheme should serve as many requests as possible.

2) ***Data Quality***: Data will become stale if a vehicle has the new version of the data but fails to upload it before the

[1]After a vehicle establishes the connectivity with one RSU, it can get the geographic information and radio range of the RSU through beacon messages. With its own driving velocity and position information, the vehicle can estimate its living time, which is its service deadline.

vehicle moves out of the RSU range. The staleness of the data will degrade the data quality for the download service. In this paper, we use the percentage of fresh data access to represent the data quality of the system. Therefore, a good scheduling scheme should update data in time and try to avoid data staleness.

It is difficult to achieve both high service ratio and good data quality. Giving more bandwidth to download requests can have a higher download service ratio, but a higher update drop ratio and hence low data quality. If update requests get more bandwidth, the service ratio decreases. There is always a tradeoff between high service ratio and good data quality. In the following sections, we first focus on improving the service ratio, and then design scheduling algorithms considering both service ratio and data quality.

3. THE BASIC SCHEDULING SCHEMES

The primary goal of a scheduling scheme is to serve as many requests as possible. We identify two parameters that can be used for scheduling vehicle-roadside data access:

- *DataSize*: If the vehicles can communicate with the RSU at the same data transmission rate, the data size can decide how long the service will last.
- *Deadline*: If a request can not be served before its deadline, it has to be dropped. Thus, the request with an earlier deadline is more urgent than the request with a later deadline.

3.1 Three Naive Schemes

There are three naive Schemes:

- First Come First Serve (FCFS): the request with the earliest arrival time will be served first.
- First Deadline First (FDF): the request with the most urgency will be served first.
- Smallest DataSize First (SDF): the data with a small size will be served first.

Figure 2 compares the service ratios under these three naive scheduling schemes. The experiment is conducted using the same simulation environment described in Section 6. The inter-arrival time of the requests is determined by the percentage of vehicles that will issue service requests, which is varied along the x axis.

As shown in the figure, when the request arrival rate is low, FDF outperforms FCFS and SDF. This is because when the workload is low, the deadline factor has more impact on the performance. After the urgent requests are served, other pending requests can still have the opportunity to get services. However, when the request arrival rate increases, the service ratio of FDF drops quickly while SDF performs relatively better. Since the system can always find short requests for service, SDF can still keep a higher service ratio. FCFS does not take any deadline or data size factors into account when making scheduling decision, it has the worst performance.

Clearly, FDF and SDF can only achieve good performance for certain workloads only. This motivates us to integrate the deadline and data size to improve the performance of scheduling.

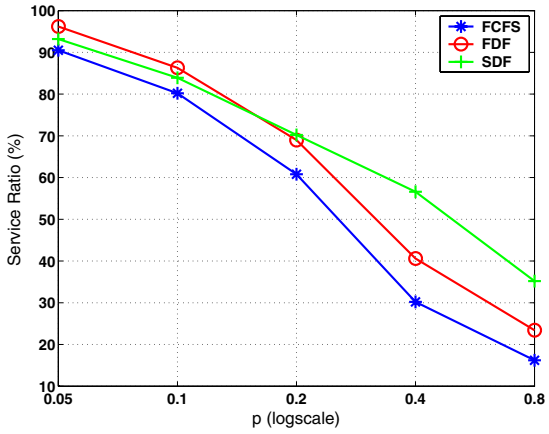


Figure 2: Service Ratio for FCFS, FDF and SDF Scheme

3.2 The $\mathcal{D} * \mathcal{S}$ Scheduling

FCFS does not consider data size and request deadline. FDF gives the highest priority to the most urgent requests while neglects the service time spent on those data items. SDF takes the data size into account but ignores the request urgency. As a result, none of them can provide a good scheduling. Inspired by [4], we propose a new scheduling scheme, called $\mathcal{D} * \mathcal{S}$ to consider both data size and deadline when scheduling vehicle-roadside data access. Intuitively,

- Given two requests with the same deadline, the one asking for a small size data should be served first.
- Given two requests asking for data with same size, the one with earlier deadline should be served first.

Motivated by the above observations, each request is given a service value based on its deadline and data size, called DS_value, as its service priority weight.

$$DS\_value = (Deadline - CurrentClock) * DataSize$$

Here, product is used to connect the deadline and data size factors because these two factors have different measurement scales and/or units. With product, different metrologies will not bring any negative effect on the comparison of two DS_values.

At each scheduling time, the $\mathcal{D} * \mathcal{S}$ scheme always serves the requests with the minimum DS_value.

3.3 The Implementation of the $\mathcal{D} * \mathcal{S}$ Scheme

A straightforward implementation of the $\mathcal{D} * \mathcal{S}$ scheme is to compute the DS_values of all requests, and then select the smallest one at the decision tick. This implementation has a computation complexity of $\mathcal{O}(m)$, where m is the number of pending requests. Here, we propose a different data structure to reduce the computation complexity. The data structure uses two sorted lists to store the requests. One list called D-List (Deadline-list) is used to record the deadline (D value) of each request. The other list called S-List (dataSize-list) is used to record the size (S value) of the data item that is asked by the request. D-List is ordered by the increasing deadline of each request and S-List is sorted in ascending order of the data size. As Figure 3 shows, the searching procedure starts by examining the entry at the

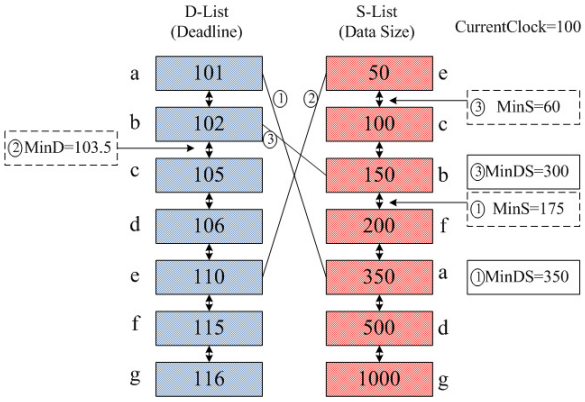


Figure 3: Search Space Pruning Structure

top of D-List. With an index we can easily find the corresponding size entry in S-List and calculate its DS_value. The MinDS is set to the DS_value of the first request in D-List. At the same time, MinS can be calculated using D' which is the deadline value of the next entry in the D-List, and the current clock time. Since the D-List is ordered increasingly, it is known that for any unexamined entry to have a DS_value less than MinDS, it must have an S value satisfying the inequality

$$S < \frac{MinDS}{D' - CurrentClock}$$

Then, we examine the entry at the top of S-List and calculate its DS_value with its S value and its corresponding D value in D-List. The MinDS value need to be updated with the current DS_value if the current DS_value is less than MinDS. Similarly, since the S-List is sorted in ascending order, an unexamined request has a DS_value less than MinDS only if its deadline value satisfies

$$D < \frac{MinDS}{S'} + CurrentClock$$

where S' is the data size value of the next entry in the S-List.

The search process keeps alternating between the D-List and S-List, updating the MinDS value when an entry with a DS_value less than MinDS is encountered and pruning the search space. The process stops when the checked entry goes across MinD or MinS, or when the search reaches halfway of both lists. At this point, MinDS is known to be the minimum DS_value for all requests and that recorded request can be served. Clearly, with this pruning technique, the search space can diminish quickly and the computation complexity can be reduced.

Figure 3 shows a simple example using these two lists. Suppose that the current clock is 100. First, the top entry (request a) in D-List is examined and the MinDS is set as 350 (= (101-100)*350). With this MinDS, we can calculate MinS=175 (=350/ (102-100)). Next, we check the entry of request e (the top of the S-List). The DS_value of request e is 500 (= (110-100)*50), which is larger than the current MinDS, so MinDS does not need to be updated. We can also get the value of MinS as 103.5 (=350/100+100). Then the second entry of D-List is checked. Its DS_value is 300 (= (102-100)*150). Since it is less than the current MinDS, MinDS should be updated to 300, and MinS is set to 60 (=300/ (105-100)). Next, we go to the second entry (request c) of S-List. Because its size is larger than the current MinS value, the search process can stop here since the unexamined requests (with size $\geq$ 100 and deadline $\geq$ 103.5) do not have a DS_value less than MinDS=300. In this example, we only need to check three index entries to find the most suitable request for service. The overhead to maintain this data structure is very low. Once an entry is added to the list, it is not moved until the corresponding request is served or dropped.

4. DOWNLOAD OPTIMIZATION: BROADCASTING

4.1 The Basic Idea

The $\mathcal{D} * \mathcal{S}$ scheduling scheme considers both request deadline and data size, and it serves one request at one time. Observe that some vehicles may ask for the same data and the wireless communication has the broadcast capability. If we can delay some requested data and broadcast it before the deadlines, several requests may be served through a single broadcast. For example, several vehicles at an intersection want to check the same traffic information. One broadcast can serve all these requests. With this optimization, the scheduling performance can be improved.

To improve the broadcast efficiency, the data with more pending requests should be served first. We add one more parameter to the $\mathcal{D} * \mathcal{S}$ scheme, i.e., the number of pending requests for the same data (N). We call the new scheme $\mathcal{D} * \mathcal{S}/\mathcal{N}$. In this new scheme, the service value, DSN_value, is calculated as

$$DSN\_value = (Deadline - CurrentClock) * DataSize/Number$$

4.2 The Implementation of $\mathcal{D} * \mathcal{S}/\mathcal{N}$

The $\mathcal{D} * \mathcal{S}/\mathcal{N}$ scheme can be implemented using a similar dual-list data structure as in $\mathcal{D} * \mathcal{S}$ to reduce the computation complexity. The difference is that each entry records the information for each request group for the same data item instead of individual requests in $\mathcal{D} * \mathcal{S}$. Because there are three parameters (deadline, data size and pending requests (N)) in consideration, we need to combine the S value and N value in advance to form a single S/N list. Since the S/N value of the corresponding data item can be updated when a new request comes, this change does not bring much maintenance overhead. At each scheduling decision tick, the same pruning process is executed alternatively between D-List and S/N-List until the request with minimal DSN_value is found.

4.3 The Selection of Representative Deadline

With the broadcast optimization, several requests can be served simultaneously in a single broadcast, which leads to more efficient use of shared bandwidth and a higher service ratio. However, when calculating their DSN_value, we need to assign each pending request group a single deadline to estimate the urgency of the whole group. If there are more than one request waiting for the service, we can use the earliest, the median or mean deadline of the group to represent the group urgency. The earliest deadline reflects the urgency of satisfying all requests in the group and the

mean and/or median deadline reflects the average urgency of the requests group. With the same setting in Section 6, we compare the performance under different representative deadlines. The simulation results (Figure 4) show that selecting different representative deadline does not have too much impact on the scheduling performance. The earliest, median, and mean deadlines lead to similar scheduling performance. Therefore, in our later simulation settings, we choose the earliest deadline to represent the urgency of all pending requests in the same group.

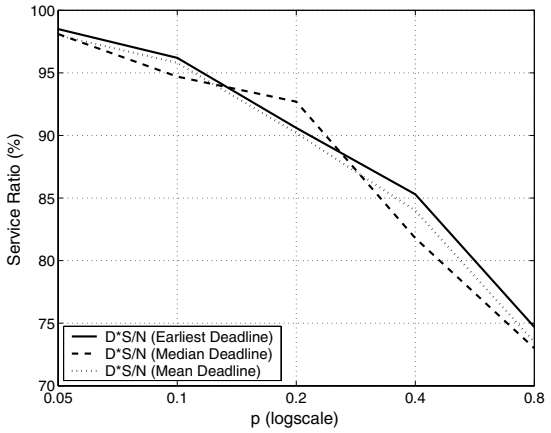


Figure 4: The Service Ratio of $\mathcal{D} * \mathcal{S}/\mathcal{N}$ with Earliest, Median, and Mean Deadline

5. UPLOAD OPTIMIZATION: TWO-STEP SCHEDULING

$\mathcal{D} * \mathcal{S}/\mathcal{N}$ can improve the service ratio, but it sacrifices the service opportunity of the upload (update) requests. For upload request, it is not necessary to maintain several update requests for one data item since only the last update is useful. As a result, the N value of upload request is always 1 in the $\mathcal{D} * \mathcal{S}/\mathcal{N}$ scheme, and hence it is not fair for update requests to compete for the service bandwidth. Clearly, more update requests have to be dropped and the data quality for downloading degrades as lots of later arriving download requests will get the stale data. A possible improvement is to incorporate a weight value to update requests to help them get a higher priority in scheduling. However, it is quite difficult to say how much weight should be given to the update requests since in a dynamic system the degradation of service quality by staleness and service ratio are incomparable. Therefore, it is impossible to use one single queue for both update and download [12]. Next, we propose a new scheduling scheme with two separate queues and a Two-Step scheduling approach to achieve the balance of data quality and service ratio.

5.1 The Basic Idea

We use two priority queues: one for the update requests and the other for the download requests. The RSU server provides two queues with different bandwidth (i.e., service probability). The benefit of using two separate priority queues is that we only need to compare the download queue and update queue instead of individual update and download requests. The scheduling goes through two steps: the first step chooses the service queue and the second step chooses the most suitable service request. Because of their specific concerns, update and download queues have their own priority scheduling schemes, which makes the scheduling more flexible.

5.2 Step I: Update Queue or Download Queue

Here we give a new definition, $Service\_Profit$, as the sum of the profit gained from update and download requests. Suppose the download requests share ρ ($0 \leq \rho \leq 1$) of the bandwidth and the update requests share the rest: 1-ρ. We need to set ρ to the best value to achieve the maximum $Service\_Profit$. To do that, we need to find the relationship between bandwidth allocation and the $Service\_Profit$ that the system can have.

The download requests can be served with fresh data or stale data, and hence the profits they bring to the system are different. Formally, $Service\_Profit$ can be presented as

$$\begin{aligned} Service\_Profit &= Update\_Profit \\ &\quad + FreshDownload\_Profit \\ &\quad + StaleDownload\_Profit \end{aligned}$$

We assume one update request can contribute the same profit as one download request with fresh data. If one update request is dropped, all the following download requests on that data item can only get the stale data and their service qualities degrade with a coefficient, say α. The service degrades until the data item is updated by the next update.

We use r_u to denote the service rate of update requests. Then the update profit rate depends on its service rate (r_u) and the bandwidth allocated to it ($1 - \rho$). After a time period t, the update profit can be approximated as:

$$Update\_Profit \simeq r_u \cdot (1 - \rho) \cdot t$$

Similarly, we use r_d to denote the service rate of the download requests. The download profit relies on its service rate (r_d), the bandwidth allocation, and the quality for each download. Note that the data quality is related to uploads. The more bandwidth allocated to the update queue, the more requests will be served with fresh data. Therefore, the download profit can be approximated by:

$$FreshDownload\_Profit \simeq r_d \cdot \rho \cdot (1 - \rho) \cdot t$$

and

$$StaleDownload\_Profit \simeq r_d \cdot \rho^2 \cdot \alpha \cdot t$$

Then the total profit can be given by

$$\begin{aligned} Service\_Profit &\simeq r_u \cdot (1 - \rho) \cdot t + r_d \cdot \rho \cdot (1 - \rho) \cdot t \\ &\quad + r_d \cdot \rho^2 \cdot \alpha \cdot t \\ &= r_d \cdot (\alpha - 1) \cdot t\rho^2 + (r_d - r_u) \cdot t \cdot \rho \\ &\quad + f_u \cdot t \qquad (0 \leq \rho \leq 1). \end{aligned}$$

We can calculate the optimal ρ to maximize the $Service\_Profit$ by solving the quadratic function with the lineal constraint on ρ. The optimal solution is:

$$\rho = \begin{cases} min(\frac{r_d - r_u}{2(1-\alpha)r_d}, 1) & (0 \leq \alpha < 1) \wedge (r_u < r_d) \\ 0 & r_u \geq r_d \\ 1 & (a = 1) \wedge (r_u < r_d) \end{cases}$$

When $\alpha = 1$, which means that the stale data do not have any negative impact on the service quality, the bandwidth

allocation totally depends on the service rate of update and download requests. Either update or download request that has a higher service rate can take the whole bandwidth.

Since the value of ρ depends on the service rate of update and download requests, which are related to the workload, it should be able to adjust adaptively with the workload. When an accident happens, there will be more update requests. Also, the request workload at daytime and night should be different. The workload is examined with a time period τ, which is referred to as the *adaptation window*. At the beginning of each τ, ρ is re-calculated. To learn about the gradual change in workload over a period of time by utilizing some history information, we record the information of two time windows and get:

$$\rho_{new} = \begin{cases} min(\frac{r_{d,k-1} - r_{u,k-1}}{2(1-\alpha)r_{d,k-1}}, 1) & (0 \leq \alpha < 1) \wedge (r_{u,k-1} < r_{d,k-1}) \\ 0 & r_{u,k-1} \geq r_{d,k-1} \\ 1 & (a = 1) \wedge (r_{u,k-1} < r_{d,k-1}) \end{cases}$$

$$\rho_k = (1 - \beta) * \rho_{k-1} + \beta * \rho_{new}$$

where $r_{d,k-1}$ and $r_{u,k-1}$ are the service rates of the previous time period and β is a parameter to measure the importance of the most recent value in comparison with the old value.

5.3 Step II: $\mathcal{D} * \mathcal{S}/\mathcal{N}$ and $\mathcal{D} * \mathcal{S}/\mathcal{R}$

As discussed in Section 4, entries in the download queue can be sorted based on their priority values calculated by the $\mathcal{D} * \mathcal{S}/\mathcal{N}$ scheme. In the update queue, we calculate the priority values with another scheme, called $\mathcal{D} * \mathcal{S}/\mathcal{R}$, where R is the service rate of the download requests in the download queue of one data item. The basic idea of $\mathcal{D} * \mathcal{S}/\mathcal{R}$ is that we use the service rate of download requests in the download queue to optimize the scheduling in the update queue. For example, given two update requests with the same DS_value, the request that updates hot data should have a higher service priority since when the data item is updated, more download requests can get the fresh data thus improving the system profit. Therefore, we add the service rate, denoted by R, as a weight factor to the priority calculation, that is:

$$DSR\_value = (Deadline - CurrentClock) * DataSize/R$$

The pruning process in $\mathcal{D} * \mathcal{S}/\mathcal{R}$ can also be optimized by using the dual-list data structure, similar to that in $\mathcal{D} * \mathcal{S}/\mathcal{N}$. The S/R-List is updated at the beginning of each adaptation time window and the R value of each entry can be updated similar to ρ, that is

$$R_k = (1 - \beta) * R_{k-1} + \beta * R_{new}$$

where R_{new} is the number of served download requests of one data item in the last adaptation window.

5.4 The Implementation of the Two-Step Scheduling

According to previous discussions, the scheduling is based on two separate priority queues. The bandwidth allocation for the update queue and the download queue depends on the periodically evaluated ρ. At each decision tick, the scheduler decides what queue (update or download) can get the service. After that, it serves a request in the specific queue. If the update queue is chosen, the $\mathcal{D} * \mathcal{S}/\mathcal{R}$ scheme is used to pick the next request to serve. If the bandwidth is assigned to the download queue, the $\mathcal{D} * \mathcal{S}/\mathcal{N}$ scheme is used. If one queue is empty, its service opportunity will be given to the other queue immediately. Figure 5 describes the flow chart of the Two-Step scheduling.

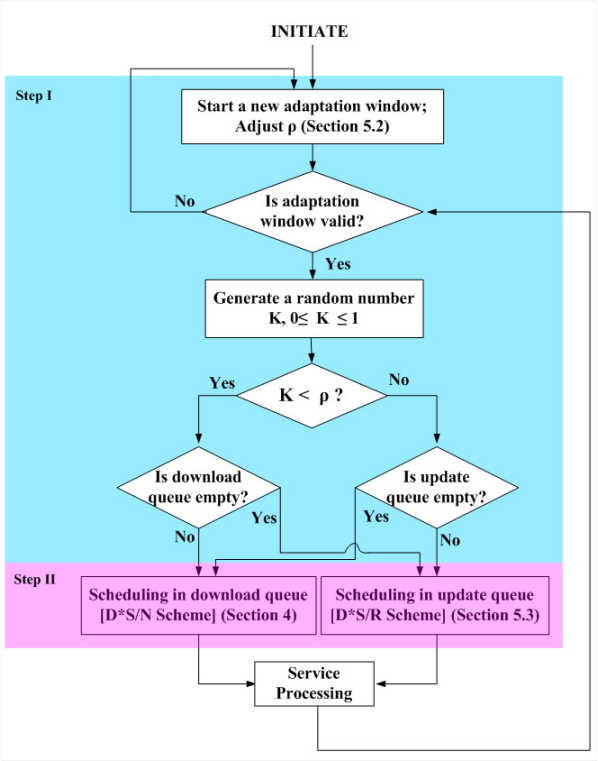


Figure 5: Flow Chart of the Two-Step Scheduling

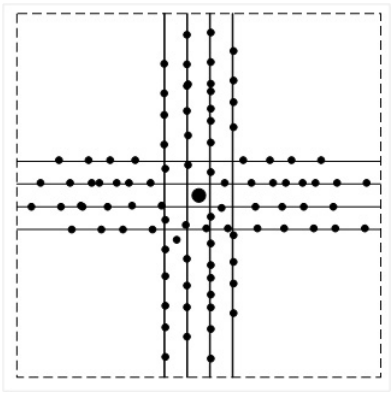
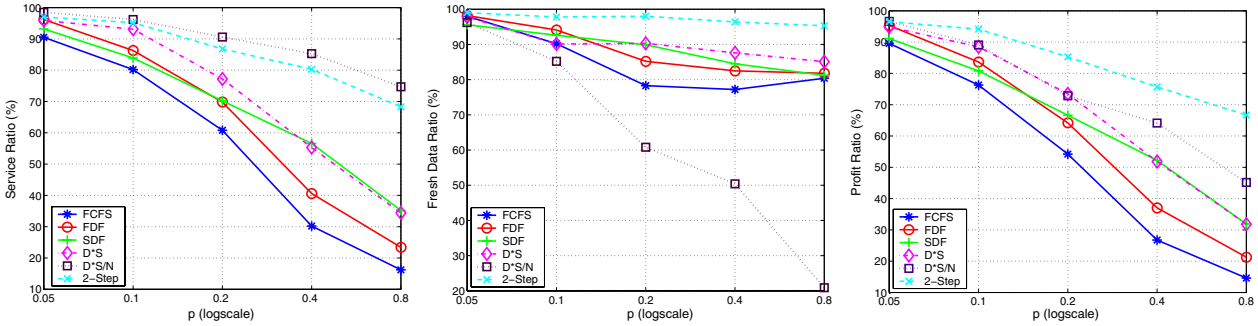


Figure 6: The Simulation Scenario Layout

6. PERFORMANCE EVALUATIONS

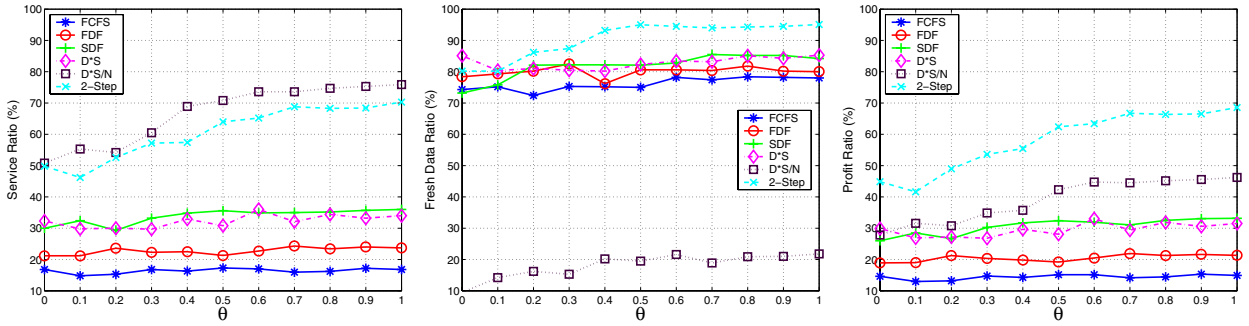
6.1 Experimental Setup

We developed an ns-2 [2] based simulator to evaluate the proposed scheduling schemes. The experiment is based on a 400m*400m square street scenario. One RSU server is put at the center of the area. It is also the intersection of



(a) Service Ratio (b) Fresh Data Ratio (c) System Profit ($\alpha = 0.5$)

Figure 7: The Effect of Workload



(a) Service Ratio (b) Fresh Data Ratio (c) System Profit ($\alpha = 0.5$)

Figure 8: The Effect of $Zipf$ Distribution Parameter θ

Table 1: Simulation Setup

Parameter	Value
Simulation Time	900s
Transmission Rate	$5Mbit/s = 625Kbyte/s$[11]
Vehicle Velocity	15m/s
Wireless Coverage	200 m
Data size	50K $\sim$ 5M, average 2.5M
Vehicle-Vehicle Space	20m
Data set size	25
$Zipf$ Parameter θ	0.8
Update Percentage	10%
Adaptation Window	40s

one horizontal road and one vertical road, where each two-way road has four lanes (Figure 6). To simulate the vehicle traffic, we randomly deploy 40 vehicles in each lane initially, i.e., a total of 160 vehicles. All vehicles move towards either end of the road. They are moving forth and back during the simulation to mimic the continuous traffic flow in the intersection area. When one vehicle reaches the end of the road, which means the vehicle will move out of the RSU area, its request not serviced will be dropped. Each vehicle issues service requests with a probability p, $(0 < p \leq 1)$. A larger p is used to simulate a heavy service workload and a smaller p is for low workload. When one vehicle is served or reaching the end of the road, it waits some time to issue a new request. The inter-arrival time of each request follows an exponential distribution with a mean of λ. Its density function is:

$$f(t) = \lambda e^{-\lambda t}$$

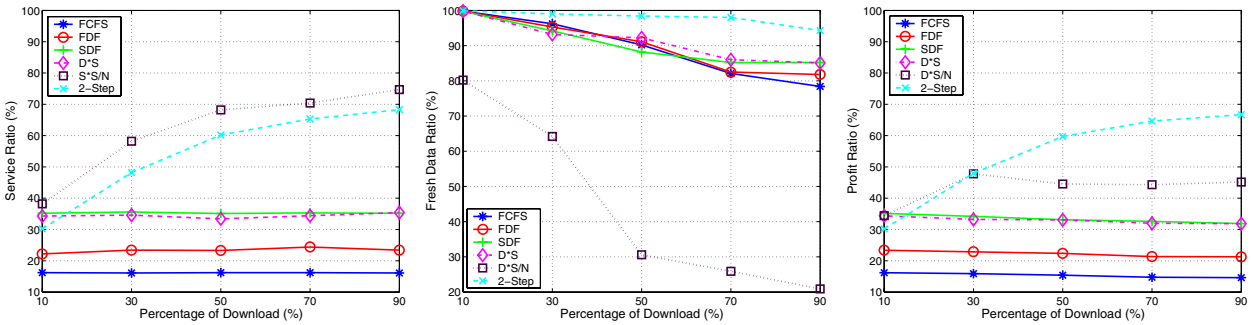
Similar to [19], the access pattern of each data item follows $Zipf$ distribution. In the $Zipf$ distribution, the access probability of the i^{th} data item is represented as follows:

$$P_i = \frac{1}{i^{\theta} \sum_{j=1}^{n} \frac{1}{j^{\theta}}}$$

where $0 \leq \theta \leq 1$, n is the database size. When $\theta = 1$, it is the strict $Zipf$ distribution. When θ=0, it becomes the uniform distribution. The data item size randomly distributes between s_{min} and s_{max}. Most of the system parameters and their default values are listed in Table 1.

6.2 The Effect of Workload

Figure 7 shows the effect of the request arrival rate to the scheduling performance for the six schemes discussed in this paper. As shown in Figure 7 (a), more requests have to be dropped as the request arrival rate increases. Since FCFS, FDF, SDF and $\mathcal{D} * \mathcal{S}$ serve each request individually, their service ratios decrease very quickly with the increasing of workload. $\mathcal{D} * \mathcal{S}/\mathcal{N}$ and the Two-Step scheme use broadcast to optimize the service. With this technique, they can achieve much higher service ratio than the other four schemes because several download requests for the same data item can be served simultaneously using a single broadcast. Since $\mathcal{D} * \mathcal{S}/\mathcal{N}$ scheduling is not fair to update requests, the data quality cannot be guaranteed. As Figure 7 (b) shows, when 80% of vehicles issue requests for service



(a) Service Ratio (b) Fresh Data Ratio (c) System Profit ($\alpha = 0.5$)

Figure 9: The Effect of Download/Update Ratio

(heavy workload), only about 20% of all served requests can get the up-to-date data, which is much lower than other schemes. Figure 7(c) compares the total $Service\_Profit$ ratios of different schemes. Here, we set $\alpha = 0.5$, which means if one download request is served with stale data, it only contributes 50% profits compared with other requests that are served with fresh data. As can be seen, the Two-Step scheme has a much higher service profit ratio than other schemes.

6.3 The Effect of Access Pattern

6.3.1 θ

Figure 8 shows the performance as a function of the access skew parameter θ. In $Zipf$ distribution, when $\theta = 0$, the access pattern is uniformly distributed, and different data items have similar popularity. As θ increases, the access pattern becomes more skewed. Since FCFS, FDF, SDF and $\mathcal{D} * \mathcal{S}$ make the scheduling decision based on individual request, the change of θ does not have too much impact on their performance. For $\mathcal{D} * \mathcal{S}/\mathcal{N}$ and the Two-Step scheme that use broadcast optimization, they can benefit from the skewness of the data access pattern with the increase of θ. When the data access pattern becomes asymmetrical, popularity becomes the major performance factor. The popular data can have a high service priority weight in $\mathcal{D} * \mathcal{S}/\mathcal{N}$ and Two-Step scheme, which helps improve the service ratio.

6.3.2 Download/Update Ratio

Figure 9 compares the performance of different schemes when the download/upload ratio changes. When more download requests come, the service ratio of $\mathcal{D} * \mathcal{S}/\mathcal{N}$ scheme and Two-Step scheme increases quickly. This performance improvement comes from the benefit of download broadcasting. Because $\mathcal{D} * \mathcal{S}/\mathcal{N}$ prefers service ratio rather than data quality, as download rate increases, its fresh data ratio decreases. This is consistent with our previous discussion. The Two-Step scheme can achieve relatively good performance on both service ratio and data quality. Therefore, it has the highest profit ratio.

6.4 Adaptivity to Traffic Condition

To study the adaptivity of the Two-Step scheme. We divide the experiment period evenly into four intervals. The first interval and the second interval have low workloads ($p = 0.05$) while the third and fourth have heavy workloads ($p = 0.8$). At the same time, the download ratios of the first and third interval are low (10% download) while the second and fourth intervals have high download ratios (90% download). We create a sudden change at the start of each interval. This experiment is to show how the Two-Step scheme can react to the changes to different scenarios and adjust ρ accordingly.

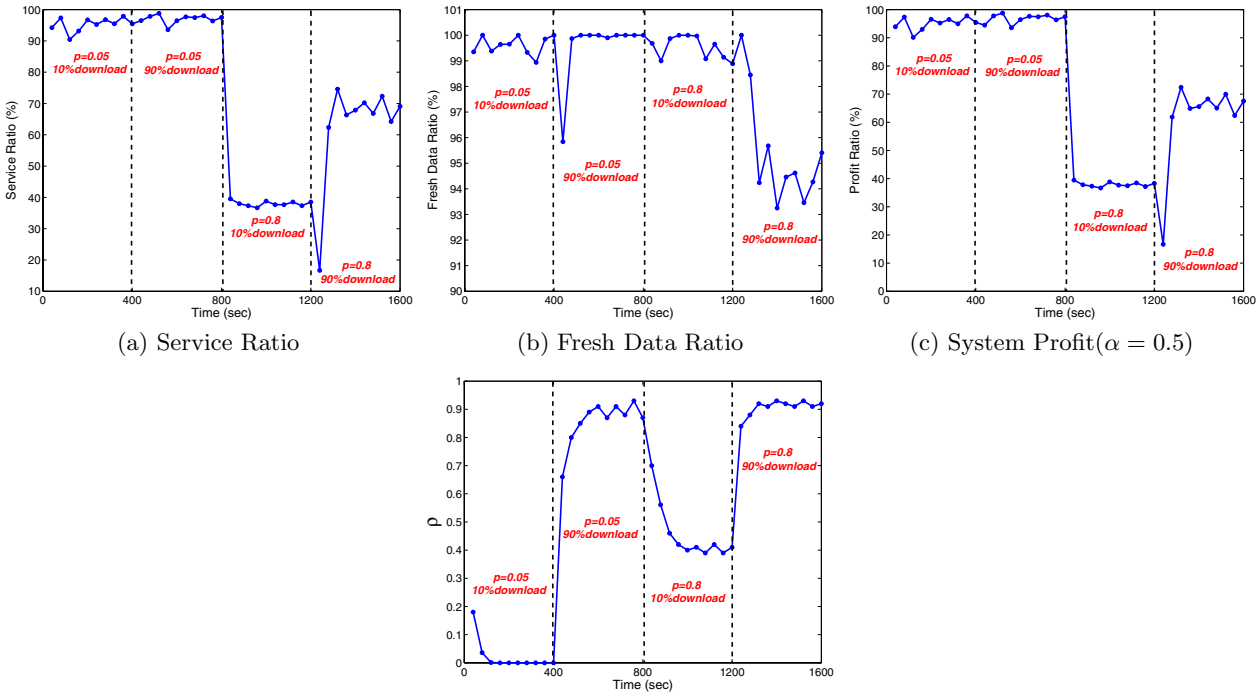
As expected, the Two-Step scheme can achieve good performance in almost all scenarios. Here, note that in the third time interval with high workload and low download ratio, the service ratio is relatively low compared with that in other intervals(Figure 10(a)). This is because in this service scenario, most arrivals are update requests. They need to be serviced one by one and hence the advantage of broadcasting cannot be well utilized. Also, although there may be sudden performance drops at the start of a switch interval, the Two-Step scheme adjusts ρ quickly and keeps a high performance. Figure 10(d) shows the value of ρ over time. ρ is the probability that download requests have higher priority than upload requests. As shown in Figure 10(d), ρ is small when the download rate is low, and ρ adapts quickly when the download rate becomes high.

7. RELATED WORK

In this paper, we have studied scheduling issues for vehicle-roadside data access. Although there are many works discussing how to benefit from vehicle-roadside data access [11, 5, 8, 7], they do not deal with the application-layer scheduling issue for data access.

There are a large amount of work related to CPU and job scheduling. Wong studied several scheduling algorithms such as first-come-first-serve (FCFS), longest wait time (LWT), most requests first (MRF) in the broadcasting environments [16]. Later, many broadcast scheduling algorithms have been proposed to reduce the waiting time and energy consumption [14, 15, 6, 4]. Acharya and Muthukrishnan [3] introduced a new performance metric called stretch for variable-size data items and investigated several scheduling algorithms. The work by [12] introduces the tradeoff between respond time and data quality, but it is based on point-to-point communication and does not take advantage of broadcasting. All these works mainly focus on responsiveness such as average/worst-case waiting time or fairness without considering the time constraints of the user requests. However, in vehicular networks, time constraint of the request has to be considered.

[9, 13] and [18] studied the scheduling problem in real-time



(a) Service Ratio (b) Fresh Data Ratio (c) System Profit($\alpha = 0.5$)

(d) ρ

Figure 10: The Study of the Adaptivity to Traffic Condition

broadcasting environment and took time constraint into account. However, they ignored the data update issue. They assume that data are read only or can only be updated by the server. Hence, they only try to improve the service ratio for download broadcasting. In contrast, our vehicle-roadside data access model is different as both update and download compete for the same bandwidth. Also, missing the update degrades the data quality.

8. CONCLUSIONS

In the paper, we addressed some challenges in vehicle-roadside data access. We proposed a basic scheduling scheme called $\mathcal{D} * \mathcal{S}$ to consider both service deadline and data size when making scheduling decisions. An efficient search space pruning technique is presented to reduce the computation complexity for making scheduling decisions. To make use of the wireless broadcasting, we proposed a new scheduling scheme called $\mathcal{D} * \mathcal{S}/\mathcal{N}$ to serve multiple requests with a single broadcast. We also identified the effects of upload requests on data quality, and proposed a Two-Step scheduling scheme to provide a balance between serving download and update requests. Simulation results show that the Two-Step scheduling scheme outperforms other scheduling schemes. Further, the Two-Step scheduling scheme is adaptive to different workload scenarios.

To the best of our knowledge, this is the first paper to address scheduling issues in vehicular ad hoc networks. The unique challenges in vehicular networks such as time constraints and upload updates will motivate further research in this area.

9. ACKNOWLEDGMENTS

This work was supported in part by National Science Foundation (CNS-0092770, CNS-0519460, CNS-0721479).

10. REFERENCES

[1] http://grouper.ieee.org/groups/scc32/dsrc/video/.

[2] http://www.isi.edu/nsnam/ns/.

[3] S. Acharya and S. Muthukrishnan. Scheduling on-demand broadcasts: New metrics and algorithms. In *Proceeding of MobiCom 98'*, 1998.

[4] D. Aksoy and M. Franklin. R*W: A scheduling approach for large-scale on-demand data broadcast. *IEEE/ACM Transactions on Networking*, volume 7, 1999.

[5] V. Bychkovsky, B. Hull, et al. A measurement study of vehicular internet access using in situ wi-fi networks. In *Proceedings of the 12th annual international conference on Mobile computing and networking(MobiCom '06)*, pages 50–61, 2006.

[6] R. Gandhi, S. Khuller, Y. Kim, and Y. Wan. Algorithms for minimizing response time in broadcast scheduling. *Algorithmica*, 38(4):597–608, 2004.

[7] D. Hadaller, S. Keshav, T. brecht, et.al. Vehicular opportunistic communication under the microscope. In *Proceedings of The 5th International Conference on Mobile Systems, Applications, and Services(MobiSys '07)*, 2007.

[8] B. Hull, V. Bychkovsky, Y. Zhang, et.al. Cartel: a distributed mobile sensor computing system. In *Proceedings of the 4th international conference on Embedded networked sensor systems(SenSys '06)*, pages 125–138, 2006.

[9] S. Jiang and N. Vaidya. Scheduling data broadcast to "impatient" users. In *Proceedings of MobiDE 99*, pages 52–59, 1999.

[10] U. D. of Transportation. http://www.its.dot.gov/vii/.

[11] J. Ott and D. Kutscher. Drive-thru internet: IEEE 802.11b for automobile users. In *Proceedings of Infocom 04'*, 2004.

[12] H. Qu and A. Labrinidis. Preference-aware query and update scheduling in web-databases. In *Proceedings of the 23rd International Conference on Data Engineering(ICDE07)*, 2007.

[13] D. Rajan, A. Sabharwal, and B. Aazhang. Power efficient broadcast scheduling with delay deadlines. In *Proceedings of the First International Conference on Broadband Networks (BROADNETS'04)*, 2004.

[14] C. Su and L. Tassiulas. Broadcast scheduling for information distribution. In *Proceeding of Infocom 97'*, 1997.

[15] N. Vaidya and S. Hameed. Scheduling data broadcast in asymemetric communication environments. In *Wireless Networks*, volume 5, 1999.

[16] J. Wong. Broadcast delivery. In *Proceeding of the IEEE*, pages 1566–1577, 1988.

[17] Y. Wu and G. Cao. Stretch-optimal scheduling for on-demand data broadcasts. In *Proceeding of the 10th International Conference on Computer Communications and Networks(ICCCN 01')*, pages 500–504, 2001.

[18] J. Xu, X. Tang, and W. Lee. Time-critical on-demand data broadcast: Algorithms, analysis, and performance evaluation. *IEEE Transactions on Parallel and Distributed Systems*, 17:3–14, 2006.

[19] L. Yin and G. Cao, "Supporting Cooperative Caching in Ad Hoc Networks," *IEEE Transactions on Mobile Computing*, vol. 5, no. 1, January 2006.

[20] J. Zhao and G. Cao, "VADD: Vehicle-Assisted Data Delivery in Vehicular Ad Hoc Networks," in *IEEE Infocom* 2006.