Integrated Push Pull Algorithm with Accomplishment Assurance in VANETs

Ajay Guleria
Panjab University, Chandigarh
Narottam Chand
and
Lalit Awasthi
National Institute of Technology, Hamirpur

Vehicular ad hoc networks (VANETs) have become useful in the intelligent transport system having applications such as road safety, traffic management, location dependent querying, advertising, etc. Vehicular networks are becoming popular with the manufacturing of more equipped vehicles and deployment of intelligent transport system by various agencies worldwide. While driving vehicles people want to access different kinds of information from road side units (RSUs). High mobility of vehicles and limited wireless range of RSUs cause them to remain for short period in the range of road side units. In this paper, we address some challenges in scheduling information from a road side unit. Vehicles send requests to road side unit and need to be served in a particular time limit. To address these challenges, we propose a flexible merged scheduling scheme with accomplishment assurance that combines the push and pull strategies probabilistically depending on the number of items present with the server at road side unit and their popularity. The cut-off point that is the separation between the push and pull items is determined in such a way that vehicles are served by the RSU before the deadline. We perform the simulation experiments to demonstrate the performance of our hybrid system.

Keywords: Aerial vehicle, trust, security, data dissemination, active bundles, data filtering

1. INTRODUCTION

Vehicular networks belong to a class of wireless networks that has emerged with the advances in wireless technologies and the automotive industry [Yang et al. 2004; NoW]. Vehicular networks are spontaneously formed between moving vehicles and nearby road side units (RSUs). The RSUs such as IEEE 802.11 access points can store data items which can be periodically broadcasted by RSU or can be pulled explicitly by the moving vehicles. These items can be of local spatial context or RSU can even work as relay of Internet access for vehicles as they are attached to infrastructure network which in turn is connected to Internet. The vehicle infrastructure integration initiative was first launched by U.S. Department of Transportation (USDOT) in 2003 [C2C; ETSI; VII]. Earlier in 1999, Federal Communication Commission (FCC) allocated 75MHz of dedicated short range communication (DSRC) spectrum at 5.9GHz (5.850-5.925GHz) to be used exclusively for vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2R) communication in America. The DSRC spectrum is divided into seven 10 MHz wide channels. IEEE has developed several standards to ensure that vehicle and RSU [DSRC; NoW; VII] can communicate with each other. DSRC is being standardized as IEEE 802.11p. IEEE 1609 is high layer standard on which IEEE 802.11p is based [Biswas et al. 2006; ETSI]. Significant interest shown and considerable efforts made by academia, industry, government and driven by road safety requirements, vehicular networks have become mature enough to be used in daily life. In vehicle-to-infrastructure (V2R)

Authors' Address: Ajay Guleria, Department of Computer Centre, Panjab University, Chandigarh, 160014, INDIA, Email: ag@pu.ac.in
Narottam Chand, Department of Computer Science Engineering, National Institute of Technology, Hamirpur, 177005, INDIA, Email: nar@nith.ac.in
Lalit Awasthi, Department of Computer Science Engineering, National Institute of Technology, Hamirpur, 177005, INDIA, Email: lalit@nith.ac.in

access, the road side unit acts as a gateway for vehicles to access the Internet. These RSUs store local information which is of importance in local geographic situation. Applications catered by RSU can be divided in the following categories:

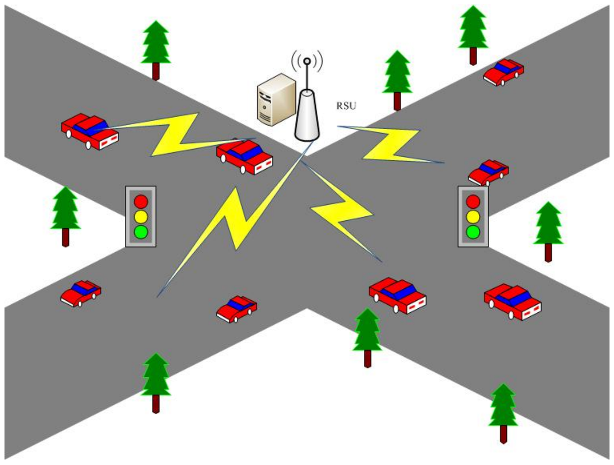
(1) Public Safety: Public safety applications are geared primarily toward avoiding accidents and loss of life. The major characteristics of these applications are that data should be disseminated fast and reliably to a large number of vehicles with a certain level of zone of relevance [Kremer 1991].
(2) Vehicular Traffic Coordination: The objective of the application in this category is to assist the drivers in different ways. The traffic data is stored at the RSUs, providing real time query and notification services to the vehicles. The data can be used to provide traffic conditions and alerts such as road congestion and accidents.
(3) Street Map Downloading: Vehicles driving into new area can download the local street maps from RSUs. These maps can be used by vehicles to select best route available for them towards their destinations.
(4) Comfort Applications and Value-added Advertisements: Vehicle occupants may download some music or videos stored at RSU. Vendors may use RSU for advertising latest products and offers to the vehicles.

In (V2R) communication, vehicles move very fast and only stay in RSU coverage area for a short period of time. So the vehicles are served by the server at RSU before the deadline specified by them at the time of request. Multiple data items belonging to above mentioned applications need to be disseminated by the server at RSU to the vehicles. The items can be periodically broadcasted or pushed by or explicitly pulled by the vehicles based on their access probabilities. The items which are in more demand can be pushed and less popular items can be allowed to be pulled from RSUs. In this paper, we propose a strategy to provide a certain level of accomplishment guarantee by meeting the vehicles deadlines. Instead of strict, sequential push and pull operation, the proposed merged scheduling framework probalistically determines the number of consecutive push and pull operations based on the systems requirement. The contributions of this paper are as follows:

(1) We first propose a basic merged scheduling scheme that combines push and pull depending on the number of items in the server and their popularity. Push and pull operations are not performed in a static and sequential manner, rather one push may be followed by another push or multiple pulls depending on the number of items present with the server and their popularity at that particular time.
(2) Basic merged scheduling algorithm is improved to make it adaptive to the system behavior and provide a certain level of performance guarantee by meeting the vehicles deadlines.
(3) We study the overall access time of the merged scheduling algorithm by varying the degree of popularity of the items from very similar to skew based on distribution parameter θ.
(4) We also study how the service delivery ratio changes with the value of skew coefficient θ.

To the best of our knowledge, our work is the first of its kind to consider a merged scheduling scheme based on popularity of the items and performance guarantee based on deadline of the requests in VANETs.

The rest of the paper is organized as follows. Section 2 presents the system model and necessary preliminaries. Section 3 describes various proposed scheduling schemes for vehicular networks. Section 4, evaluates the performance of proposed schemes. Section 5 describes related work. Finally, we summarize and conclude the paper in Section 6.



Scenario for vehcile-to-infrastructure communication.

2. SYSTEM MODEL AND PRELIMINARIES

2.1 System Model

As shown in Figure 1, we have assumed a single RSU server, large number of vehicles who seek data from RSU when they move into communication range of RSU. The RSU maintains a database consisting of N distinct items which it can provide to the vehicles. The RSU server maintains a service cycle, which is preemptive. When one vehicle enters in the range of RSU, it listens to the wireless channel. All the vehicles are assumed to be equipped with GPS technology so that they can compute service deadlines of their requests based on speed of the vehicle and coverage area of RSU. Each vehicle is characterized by a request which consists of vehicle id, item id, and deadline. All the requests are queued at the RSU server upon arrival. Deadline in the requests requires the items to be served within a precise timeframe, beyond which the service becomes useless. Scheduling of information in vehicular networks impose special requirements on the scheduling algorithms. We assume the following:

(1) Certain items are in more demand than others.

(2) Requests lifetime is very short as the vehicles are moving and RSU has coverage limitations.

(3) Requests whose deadline cannot be met are to be dropped.

3. PROPOSED SCHEDULING ALGORITHMS

Data dissemination in vehicular networks can be divided into two categories: (1) push-based data broadcasting, and (2) pull-based data dissemination. In push-based systems, the server periodically broadcasts a set of data items to all vehicles without any vehicle intervention. The vehicles just listen to the downlink channel to obtain required data items. The server broadcasts next data items based on certain scheduling policy without vehicle intervention [Acharya et al. 1997; Gandhi et al. 2004]. Indeed, this saves bandwidth in the uplink wireless channels, but it suffers from wasting resources in downlink wireless channels by repeatedly transmitting the less popular items. In pull-based systems, a vehicle uses the uplink channel to send an explicit request for a particular data item to the server at RSU. The server, in turn, transmits the item to the vehicle. Requests from vehicles get accumulated at RSU and are served by the server in a unicast fashion one by one based on certain scheduling policy. Pull-based scheduling becomes inefficient when the system is highly loaded and the data items are very popular. A merged approach that uses the flavours of both push-based and pull-based scheduling algorithms in vehicular networks appears to be more attractive. The key idea of merged approach is to exploit intelligently both push and pull scheduling. In hybrid scheduling the items to be disseminated by RSU are divided

into two sets: (1) popular, and (2) non-popular. The popular items are broadcasted using push-based scheduling strategy, and less popular items are transmitted using on-demand pull-based communication strategy. A suitable balance between push and pull schemes is of utmost importance.
In this section, we describe our proposed strategies for VANETs which are based on works of described in [Hameed and Vaidya 1999; Saxena et al. 2004] for any conventional wireless systems .

3.1 Static Merged Scheduling (SMS)

All data items are stored in a database at RSU server. The RSU server uses both push and pull mechanisms for information dissemination. The entire set of items is divided into two disjoint sets, the access set and the request set. The access set contains the hot data items that are very popular among vehicles and hence would be disseminated based on push scheduling algorithm. The request set contains cold data items, which are not so popular, and hence would be disseminated on-demand based on pull scheduling algorithm. At the start, the push set will be empty so the system starts as a pure pull-based scheduler. Based on the requests received for each item during a certain interval of time, it dynamically moves to a merged system with the data items separated into the push set and the pull set. The RSU server at regular intervals monitors the data access probabilities of the items and the arrival rate of the requests. A cut-off point is fixed between push and pull sets, and the RSU server continuously alternates push and pull phase. After every push and every pull operation, the RSU server accepts the set of requests arriving into the system. The RSU server uses a broadcast program for frequently accessed data items where items are periodically broadcasted by using Packet Fair Scheduling *(PFS)* algorithm [Hameed and Vaidya 1999]. The RSU server uses pull mechanism for serving on-demand data requests from vehicles. At any instance of time, the server either broadcasts an item or replies as a response to queued items. *D*S/N scheduling scheme* [Zhang et al. 2007; 2010] is used to provide items on demand. The RSU server stores items in sorted array N by nonincreasing access probabilities $P_1 \geqslant \ldots \geqslant P_N$. The access probability gives a measure of an item's popularity among the vehicles. Similar to [Saxena et al. 2004], the access probability P_i of an item i is governed by the Zipfs distribution and depends on the access skew coefficient θ. Value of θ varies in the range [0, 1]. When θ is small close to 0, P_i is well balanced, but as θ reaches close to 1, the popularity of the data items dips. Every item has a different size randomly distributed between 1 and L, where L is the maximum size. The RSU server pushes M items and vehicles on demand pull the rest $(N - M)$ items. So, the total probabilities of items being served in push set and pull set are respectively given by $\sum_{i=1}^{M} P_i$ and $\sum_{i=M+1}^{N} P_i = (1 - \sum_{i=1}^{M} P_i)$. here P_i denotes the access probability of item i. Figure 2 shows the pseudo code of static merged scheduling (SMS).

```
 1: while true do
 2:     select an item using Packet Fair Scheduling and broadcast it;
 3:     acknowledge new requests from vehicles;
 4:     overlook the requests which are currently in push set;
 5:     if  pull queue is not empty then
 6:         employ $D*S/N$ scheduling to pull out an item from pull queue;
 7:         clear pending requests for that item;
 8:         pull that particular item;
 9:         acknowledge new requests from vehicles;
10:         overlook the requests which are currently in push set;
11:         include the requests for items in the pull queue;
12:     end if
13: end while
```

SMS algorithm.

3.2 Dynamic Merged Scheduling (DMS)

Straight sequential combination of push and pull fails to explore the systems current condition. In reality, it is a better idea to perform more than one push operation followed by multiple pull operations, depending on the number of items currently present in the system. So the push and pull operations are merged in a dynamic fashion. Our proposed dynamic merged scheme adopts this strategy based on the number of items present and their popularity [Saxena et al. 2004].

In this model also, we have assumed a single RSU server, multiple vehicles, and a database consisting of N distinct items, of which M items are pushed and the remaining $(N-M)$ items are pulled. The access probability P_i of an item i is governed by the Zipfs distribution and depends on the access skew coefficient θ. For the dynamic integrated push-pull system the value of θ is changed dynamically, which results dynamic variation of P_i and the size of the push and pull queues. *PFS* [Hameed and Vaidya 1999] and *D*S/N scheduling scheme* [Aksoy and Franklin 1999; Zhang et al. 2007; 2010] are used for selecting the item to be pushed and pulled, respectively. After every push or pull operation, the next push or pull operation is probabilistically determined using the following equation [Saxena et al. 2005; Saxena and Pinotti 2004; Saxena et al. 2004]:

$$\eta_1 = P_r[push|push] = \frac{M}{N}\sum_{i=1}^{M} P_i \tag{1}$$

$$\eta_2 = P_r[pull|push] = 1 - \eta_1 \tag{2}$$

$$\eta_3 = P_r[pull|pull] = \frac{(N-M)}{N}\sum_{i=M+1}^{N} P_i \tag{3}$$

$$\eta_4 = P_r[push|pull] = 1 - \eta_3 \tag{4}$$

P_{r1} and P_{r2} are predefined thresholds which govern the dynamism of the algorithm. At the end of every push operation, the system checks the value of η_1. If $\eta_1 \geqslant P_{r1}$, the system goes for another push, otherwise it switches to the pull mode. Similarly, at the end of every pull operation, it computes the value of η_3. If $\eta_3 \geqslant P_{r2}$ then the system performs another pull operation, otherwise it switches to the push mode. At the RSU server end, the system begins as a pure pull-based scheduler. If the request is for a push item, the server simply ignores the request as the item will be sooner pushed according to the PFS algorithm. However, if the request is for a pull item, the server inserts it into the pull queue with the associated arrival time and updates DSN value. The pseudo code of dynamic merged scheduling is shown in Figure 3. Throughout the system, the arrival rate of requests from the vehicles in RSU area is assumed to obey the Poisson distribution with mean λ_1. The RSU server serves the vehicles by broadcasting or on- demand serving the items. The service times of push and pull systems are exponentially distributed with mean μ_1 and μ_2, respectively. Table I lists the symbols with description used in analysis. The RSU server pushes M items and vehicles on demand pull the rest $(N-M)$

Table I: Symbols used during analysis.

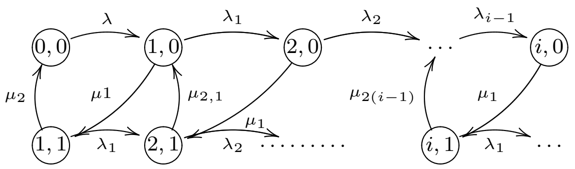
Symbol	Description
N	Number of items
M	Size of push set
i	Index of item
P_i	Access probability
λ	Arrival rate
L_i	Length of item i
μ_1	Service rate of push queue
μ_2	Srvice rate of pull queue
C	Maximum number of vehicles

items. So, the total probabilities of items being served in push set and pull set are respectively

```
 1: while true do
 2:   select an item using Packet Fair Scheduling and push it;
 3:   acknowledge new requests from vehicles;
 4:   overlook the requests which are currently in push set;
 5:   include the requests for items in the pull queue;
 6:   calculate probabilities of $\eta_1$ and $\eta_2$
 7:   if ($P_{r1} \leqslant \eta_1$) go to step 2
 8:   otherwise go to step 9
 9:   if pull queue is not empty then
10:     employ $D*S/N$ scheduling to pull out an item from pull queue;
11:     clear pending requests for that item;
12:     pull that particular item;
13:     acknowledge new requests from vehicles;
14:     overlook the requests which are currently in push set;
15:     include the requests for items in the pull queue;
16:   end if
17:   compute probabilities of $\eta_3$ and $\eta_4$
18:   if ($P_{r2} \leqslant \eta_3$) go to step 9 otherwise go to step 2;
19: end while
```

DMS algorithm.

given by $\sum_{i=1}^{M} P_i$ and $\sum_{i=M+1}^{N} P_i = (1 - \sum_{i=1}^{M} P_i)$. P_i here dontes the access probability of item i. Assumption is that P_i follows the Zipfs distribution with access skew coefficient θ, such that $P_i = \frac{(\frac{1}{i})^\theta}{\sum_{j=1}^{n}(\frac{1}{j})^\theta}$. After every push, the RSU server performs another push with probability η_1 and a pull with probability η_2. Similarly, after every pull, it performs another pull with probability η_3 and a push with probability η_4. Figure 4 illustrates the underlying birth and death process [Gross and Harris 1992] of dynamic merged system, in which the arrival rate in the pull system is given by $\lambda = (1 - \sum_{i}^{M} P_i)\lambda_i$. Any state of the overall system is represented by the tuple (i, j),



Performace modeling of merged scheduling.

in which i represents the number of items in the pull system. On the other hand, j is a binary variable, with $j = 0 (or 1)$ respectively, representing whether the push system (or pull system) is currently being served by the RSU server [Saxena et al. 2004].

The system moves from state *(i, j)* to state *(i +1, j)* when a data item arrives in the pull system; $\forall_i$, such that $0 \leqslant i < \infty$ and $\forall_j \in [0, 1]$. The service of items by RSU server results in different actions. The service of an item in the push queue results in transition of the system from state *(i, j = 0)* to state *(i, j = 1)*, with probability η_2, $\forall_i$ such that $0 \leqslant i < \infty$. With probability η_1, the push service makes the system to remain in the same state. The service of an item in the pull results in transition of the system from state *(i, j = 1)* to the state *(i -1, j = 0)*, with probability η_4 and state *(i - 1, j = 1)* with probability η_3, $\forall_i$, such that $1 \leqslant \infty$. The state of the system at *(i = 0, j = 0)* represents that the pull queue is empty and any subsequent service of the elements of push system leaves the system in the same *(i= 0, j= 0)* state. The state *(i = 0, j = 1)* is not valid [Saxena et al. 2004]. To get the steady state, using the flow balance conditions by Chapman-Kolmogrov's equation [Gross and Harris 1992] and using findings of Navrati [Saxena et al. 2004], we have the following three equations representing the systems behavior:

$$p(i,0) = \frac{p(i-1,0)\lambda + p(i+1,1)\eta_4\mu_2}{(\lambda + \eta_2\mu_1)} \tag{5}$$

$$p(i,1) = \frac{p(i,0)\eta_2\mu_1 + p(i-1,1)\lambda}{(\lambda + \eta_3\mu_2 + \eta_4\mu_4)} \tag{6}$$

$$p(0,0)\lambda = p(1,1)\mu_2 \tag{7}$$

Where, *p(i, j)* represents the probability of state *(i, j)*. The first two Equations i.e (5) & (6) represent the overall behavior of the system for push and the pull system, the Equation (7) represents the initial condition of the system. The most efficient way to solve the above equations is using z transforms [Gross and Harris 1992; Saxena et al. 2004]. Performing z transforms of Equation (5) and Equation (6) and using the initial condition, we get a pair of transformed equations:

$$\begin{aligned} P_2(z)\eta_4\mu_2 = z[P_1(z) - p(0,0)](\lambda + \eta_2\mu_1) \\ - z^2\lambda P_1(z) + p(1,1)\eta_4\mu_2 \end{aligned} \tag{8}$$

$$P_2(z) = \frac{\eta_2\mu_1[P_1(z) - p(0,0)]}{(\lambda + \eta_3\mu_2 + \eta_4\mu_2 - z\lambda)} \tag{9}$$

The occupancy of pull states is the total traffic of pull queue and is given by: $P_2(1) = \sum_{i=1}^{C} p(i,1) = \rho$. The occupancy of the push states (upper chain) is similarly given by $P_1(1) = \sum_{i=1}^{C} p(i,0) = (1-\rho)$. Using these two relations in Equation (8), we can obtain the initial probability, p(0, 0). The initial probability of the system, i.e., probability of an empty pull queue is given by the following equation:

$$p(0,0) = \frac{\mu_1(\eta_2 - \eta_2\rho - \rho\eta_4\mu_2)}{\lambda + \eta_2\mu_1 - \eta_4\lambda} \tag{10}$$

Expected measure of system performance, such as the average number of elements in the system and mean waiting time is to differentiate the z transformed equation (i.e Equation (8)), and capture the values of the z transformed variable at z=1.

$$\begin{aligned} \eta_4\mu_2\frac{dP_2(z)}{dz}|_{z=1} &= \eta_2\mu_1\frac{dP_1(z)}{dz}|_{z=1} \\ &+ (1-\rho)(\eta_2\mu_1\lambda)p(0,0)(\lambda + \eta_2\mu_1) \\ E[\ell^q_{pull}] &= \frac{dP_2(z)}{dz}|_{z=1} \end{aligned} \tag{11}$$

Where $\frac{dP_1(z)}{dz}|_{z=1}$ gives the number of elements in push system in Packet Fair Scheduling. Once, we have the expected number of items in the pull system from Equation (11), using Littles formula [Gross and Harris 1992; Saxena et al. 2004], we can get the vaules of mean waiting time of the system ($E[W_{pull}]$), and expected number of items ($E[\ell^q_{pull}]$) in the pull queue as:

$$E[W^q_{pull}] = E[W_{pull}] - \frac{1}{\mu_2} = \frac{E[\ell_{pull}]}{\lambda} - \frac{1}{\mu_2} \tag{12}$$

If M represents the number of items in the push system, then the expected cycle time of the push system is given by:$\sum_{i=1}^{M} \frac{S_iP_i}{(1-\rho)\mu_1}$. Hence, the expected access time ($E[T_a]$) of proposed merged system is given by:

$$E[T_a] = \sum_{i=1}^{M} s_iP_i + E[W^q_{pull}]\sum_{i=M+1}^{N} P_i \tag{13}$$

The above expression provides an assessment of the average behavior of our merged system.

3.3 Accomplishment Assurance with Static Merged Scheduling (AASMS)

Accomplishment assurance is required in vehicular networks where requests from vehicles need to be responded within a precise timeframe of service. For all N possible values of the cutoff point, the RSU server computes the expected merged waiting time $E[T_w]$. From now on, let $E[T_w]$ denotes such expected merged waiting time. These N expected waiting times are stored, one at a time along with the index of the cutoff point that generates it sequentially in a vector V. That is, for all $1 \leqslant i \leqslant N, V[i,1] = E[T_w(i)]$ and V[i, 2] = i. V is maintained and sorted with respect to the first component, i.e., the expected merged waiting time. Using this structure, the RSU server can extract the first element V[0] of this vector in a single access, which will indicate in correspondence at which value M_0 of the cutoff point, the minimum expected merged waiting time $V[0,1] = E[T_w(N0)]$ is achieved. The RSU server broadcasts V[0] from time to time, thereby informing the vehicles of the best performance it can provide. The server continuously broadcasts the basic merged scheduling that corresponds to the cutoff point M in use. On the other side, when a vehicle sends a request for any item j, it also specifies an expectation $\Delta(j)$ of its possible waiting time for item j. Indeed, $\Delta(j)$ reflects deadline of request based on speed of the vehicle and range of RSU. To accept a vehicle's request for item j with expectation $\Delta(j)$, the RSU server estimates the expected waiting time for j at that instant using the values stored in vector V and the knowledge of the current cutoff point M in use for the hybrid scheduling algorithm broadcasted by the server. If the expected hybrid waiting time provided by the system is smaller or equal to the expectation time of the vehicles request, then the certain level of accomplishment assurance to the vehicle is guaranteed. Otherwise, the RSU server checks whether the item j belongs to its current push set, that is if $j \leqslant M$. If this is true, it compares the vehicle request deadline with the expected waiting time guaranteed by the packet fair scheduling queuing for the push part of the system, say $E[T_{PFS}(j)]$. Such a value is known and is proportional to the space S_j between two instances of j in the packet fair queuing scheduling [Saxena et al. 2004], and it is different from the overall expected waiting time of the server although it depends on the cutoff point in use. If the expectation of the vehicle $2E[T_{PFS}(j)]$ is smaller than or equal to $\Delta(j)$, the request can still be accepted and served. The $E[T_{PFS}(j)]$ is doubled to make allowance for the fact that the system pulls one item between two consecutive pushed items. Otherwise, the request can be accepted only if the cutoff point is updated. RSU server will perform a binary search on the vector V to look for a cutoff point value whose corresponding expected merged waiting time is the largest value smaller than or equal to $\Delta(j)$. Such a value always exists if $V[0,1] \leqslant \Delta(j)$. Then, the cutoff point is updated accordingly and the scheduling is reinitialized. Figure 5 shows a pseudo code for this algorithm.

```
1: for i = 1 to N do
2: calculate the mean waiting time $E[T_w(i)]$;
3: sort all the values of $E[T_w(1)]$, . . . , $E[T_w(N)]$ and store them in increasing order in a vector V
4: broadcast to the vehicles the minimum affordable waiting time V[0];
5: accept the vehicle's request for any item j with expected waiting-time $\Delta(j) \geq E[T_w(M)]$, where M = current
   cutoff;
6: if (condition at step 5 is not verified and $j \leqslant M$) accept the vehicle's request for any item j with expected
   waiting time $\Delta(j) \geq 2E[T_{PFS}(j)]$, where $E[T_{PFS}(j)]$ is the expected waiting time guaranteed by packet fair
   scheduling;
7: if (both conditions at lines 5 and 6 are not verified) and $\Delta(j)$ is larger than expected waiting time stored in
   V[0,1]) then
8: get the largest value of $E[T_w(j)] \leqslant \Delta(j)$ from V
9: adjust the cutoff point and restart new merged scheduling;
10: Otherwise reject the request.
```

AASMS algorithm.

4. EVALUATION

In order to evaluate the proposed scheduling algorithms we have used simulator based on sumo [MOVE] for traffic simulation and ns-2 [NS2] for network simulation. We have compared the static merged scheduling with dynamic merged scheduling.

4.1 Performance Metrics

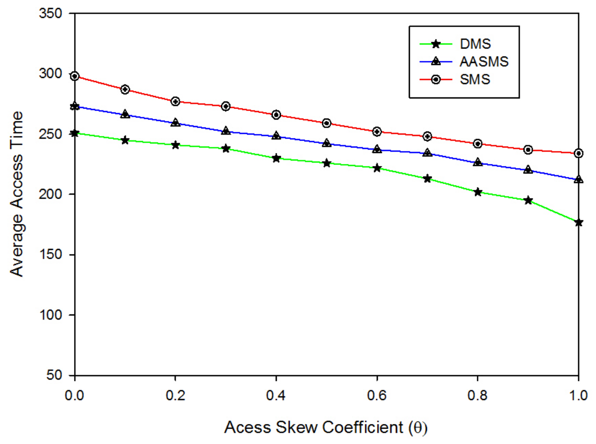
The prime goal of all merged scheduling schemes is to reduce the overall expected access time. In case of vehicular networks a performance guarantee is required to be delivered i.e. requests from vehicles need to be responded within a precise timeframe, thereby ensuring a good delivery ratio with minimal average access time. We use the following metrics for performance evaluation.

(1) Average access time: Definitive goal of the proposed hybrid algorithm is to reduce the expected access time.

(2) Service ratio: Better scheduling algorithm should serve as many requests as possible. Delivery ratio or service ratio is defined as the ratio of the number of requests served before deadlines to the total number of arriving requests from vehicles to server at RSU.

4.2 Experimental Setup

The experiment is based on a 200x200m^2 square street intersection of one horizontal road and one vertical road where each two-way road has two lanes. One RSU server is placed at the center of the junction. To generate the vehicle traffic, we randomly deploy 10 vehicles in each lane, which makes a total of 40 vehicles. All vehicles move towards either end of the road. They are moving forward and backward during the simulation to have 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 serving area, its request not serviced will be dropped. Each vehicle sends requests with a probability $p, (0 < p \leqslant 1)$. Value of p closer to 1 generates a heavy workload for RSU Server. The arrival rates of requests from vehciles and service rates of these requests by the RSU server follows Poisson distribution. The arrival rate (λ) is varied between 5 and 20. The RSU server in the experiment serves N=10 items. In order to keep the access probabilities of items from very similiar to very skewed, θ is varied between 0 and 1. Most of the simulation parameters with their corresponding values are listed in Table II.



Effect of θ on average access time.

Table II: Simulation parameters.

Parameter	Value
Simulation time	1000s
RSU coverage area	200m
Vehicle speed	20m/s
Transmission rate	11 Mbps
Item size	4Mb, max
Item set size (N)	10
Skew coefficient, θ	0.0-1.0
Arrival rate, λ	10, 20

4.3 Results

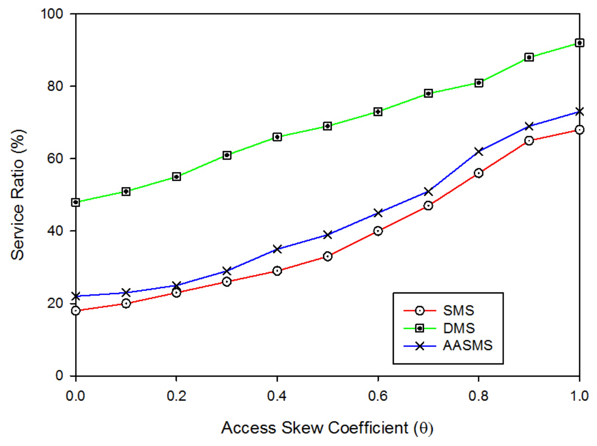
4.3.1 *Effect of Access Pattern.* When θ is zero or close to zero, the access pattern is uniformly distributed, and different items have similar popularity. As its value increases the access pattern becomes more skewed. The simulation experiments are evaluated for N=10 items. The average value of arrival rate λ is taken 10. In order to keep access probabilities of the items from similar to much skewed, θ is dynamically varied from 0.0 to 1.0. Figure 6 shows the variation of average access time with different values of θ. The average access time for DMS scheme is lower than AASMS and SMS schemes. Figure 7 demonstrates service ratio of all three schemes against different values of θ for N=10 and λ=10. Note that, dynamic merged scheduling (DMS) achieves better service ratio than static merged scheduling (SMS). For low values of θ, dynamic merged scheduling (DMS) performs much better than static merged scheduling (SMS). For higher values of θ, all schemes gain slightly in terms of service ratio. Accomplishment assurance static merged scheduling (AASMS) performs slightly better than static merged scheduling (SMS).

4.3.2 *Effect of Arrival Rate.* The effect of the vehicles request arrival rate on the scheduling performance for the three scheduling schemes discussed in this paper is shown in Figure 8. Simulation experiments are performed for a total number of N=10 items. Arrival rate is varied between 5 to 20. The value of skew coefficient θ is taken as 0.6 where items access probabilities is well balanced. Figure 8 demonstrate the effect of arrival rate on various proposed hybrid scheduling schemes.

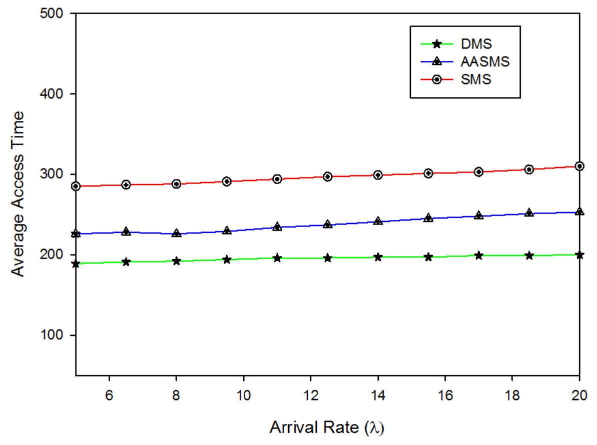
4.3.3 *Conclusion.* Results shows that dynamic merged scheduling scheme (DMS) has less average access time as compared to static merged scheduling (SMS) with different arrival rates. The variation of expected access time with different arrival rates is pretty low in case of all schemes. Figure 9 shows that service ratio decreases as the arrival rate increases in case of all schemes. Higher service ratio is achieved in case of accomplishment assurance static merged scheduling scheme (AASMS).

5. RELATED WORK

Major research challenges in VANETs are introduced in [Sichitiu and Kihl 2008; Santos et al. 2006; Durresi et al. 2006; Xu et al. 2006]. In [Sichitiu and Kihl 2008], authors presented several major classes of applications and types of services which can exist in VANETs and common performance evaluation techniques for these networks. They have critically reviewed various research works such as taxonomy of applications, communication technologies, MAC layer protocols, routing protocols, etc. in VANETs. According to the authors, data scheduling in VANETs is an important issue and needs to be addressed for efficient disseminations of information. During our research we have made an attempt to address the problem of data scheduling by employing hybrid (push and pull) method. High mobility of vehicles is main challenge in VANETs which leads to short deadline to access data from RSU and causes highly dynamic topology. In case of vehicle to roadside data access there is more than one vehicle under coverage of one RSU. So multiple vehicles can send data upload/download request to RSU. Because deadlines are short therefore, RSU needs to process these requests efficiently in terms of time. Many broadcasting

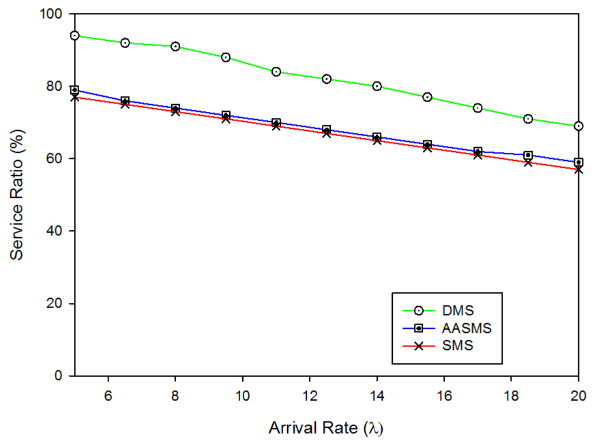


Effect of θ on service ratio.



Effect of λ on average access time.

algorithm have been proposed to reduce the waiting time [Gandhi et al. 2004; Yang et al. 2004; Yee and Navathe 2003]. In [Acharya et al. 1997] and [Hameed and Vaidya 1999] addressed the broadcast scheduling problem in heterogeneous environments, where data items have different sizes. All these schemes either used push scheduling mechanism for broadcasting items or used pull mechanism for data dissemination. In [Xu et al. 2006] proposed online scheduling algorithm for time critical on-demand data broadcast but they assumed that data can only be updated by server i.e. vehicle can only request download, it does not allow vehicles to update urgent data. [Jiang and Vaidya 1999; Daniel et al.] proposed periodic push based broadcast, which is not well suited to VANET applications. In [Zhang et al. 2007; 2010], first proposed D*S algorithm, further optimized downloading by using D*S/N and optimized uploading by using D*S/R while maintaining different queues for download and upload requests. The algorithm assigns different bandwidth to these queues and serves upload requests on basis of service rate of data items in past. None of these earlier data access schemes considered the optimization of service queue and incoming request analysis to remove the various sources of time wastage in data access scheme for VANETs. In contrast we have provided algorithms for analysis of incoming data and to make service queue size dynamic to increase the deadline by reducing the time wastage. The simulation



Effect of λ on service ratio.

results show that the proposed algorithms significantly improve the service ratio. Basic hybrid scheduling with heterogeneous items in wireless networks were studied by [Saxena et al. 2005; Saxena and Pinotti 2004]. Based on popularity, they divided the requests into two sets. Based on a cut-off point between these sets, items were broadcasted in push phase and on-demand disseminated in pull phase. Instead of sequentially following a strict push pull overlapping they proposed a dynamic scheme for hybrid information scheduling [Saxena et al. 2004]. Their work was for generic asymmetric wireless environments. These schemes did not consider the deadline and high mobility of the vehicular ad hoc networks. Based on the work of [Saxena et al. 2005; Saxena and Pinotti 2004; Zhang et al. 2010], we proposed our variants of merged algorithms for VANETs.

6. CONCLUSIONS

In this paper, we have addressed some challenges in scheduling information in VANETs. We have proposed a new framework for merged scheduling in VANET, for disseminating information from a RSU server. The framework is initially designed for statically merging the push and pull schemes to provide data to the moving vehicles from a RSU. The cut-off point used to separate push and pull system is determined such that overall mean access time is minimized. We have then improved our merged scheduling framework to make it adaptive to the systems behavior. Instead of strict, sequential push and pull operations, the hybrid scheduling framework now probabilistically determines the number of consecutive push and pull operations based on the system requirements. Subsequently, we propose a strategy to provide a certain level of performance guarantee by meeting the vehicles deadlines. It gives certain level of accomplishment assurance for moving vehicles in a RSU area. Simulation results demonstrate that hybrid technology employing both push and pull performs better than individual push and pull technologies.

REFERENCES

Acharya, S., Franklin, M., and Zdonik, S. 1997. Balancing push and pull for data broadcast. In *Proceedings of the 1997 ACM SIGMOD international conference on Management of data*. SIGMOD '97. ACM, New York, NY, USA, 183–194.

Acharya, S. and Muthukrishnan, S. 1998. Scheduling on-demand broadcasts: new metrics and algorithms. In *Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*. MobiCom '98. ACM, New York, NY, USA, 43–54.

Aksoy, D. and Franklin, M. 1999. Rw: a scheduling approach for large-scale on-demand data broadcast. *IEEE/ACM Trans. Netw. 7,* 6 (Dec.), 846–860.

Biswas, S., Tatchikou, R., and Dion, F. 2006. Vehicle-to-vehicle wireless communication protocols for enhancing highway traffic safety. *IEEE Communication Magazine 44,* 1, 74–82.

C2C. Car 2 car communication consortium. `http://www.car-to-car.org`.

Daniel, K., Wong, Tepe, K., Chen, W., and Gerla, M. Inter-vehicular communications. In *IEEE Wireless Communications*. Vol. 13. 6–7.

DSRC. Dedicated short-range communications. `http://www.leearmstrong.com/DSRC/`.

Durresi, M., Durresi, A., and Barolli, L. 2006. Adaptive inter vehicle communications. *International Journal of Wireless Information Networks 13*, 151–160. 10.1007/s10776-006-0030-5.

ETSI. The european telecommunications standards institute. `http://www.etsi.org`.

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

Gross, D. and Harris, C., Eds. 1992. *Fundamental of Queuing Theory.* John Wiley Sons Inc.

Hameed, S. and Vaidya, N. H. 1999. Efficient algorithms for scheduling data broadcast. *Wireless Networks 5*, 183–193.

Jiang, S. and Vaidya, N. H. 1999. Scheduling data broadcast to impatient users. In *Proceedings of the 1st ACM international workshop on Data engineering for wireless and mobile access*. MobiDe '99. ACM, New York, NY, USA, 52–59.

Kremer, W. 1991. Realistic simulation of a broadcast protocol for an inter vehicle communication system. In *Proc. of the 41st IEEE Vehicular Technology Conference*. St. Louis, MO, 624–629.

MOVE. Mobility model generator for vehicular networks. `http://lens.csie.ncku.edu.tw/`.

NoW. Network-on-wheels. `http://www.network-on-wheels.de`.

NS2. The network simulator ns-2. `http://www.isi.edu/nsmam/ns/`.

Santos, R., Edwards, A., and Alvarez, O. 2006. Towards an inter-vehicle communication algorithm. In *Electrical and Electronics Engineering, 2006 3rd International Conference on.* 1 –4.

Saxena, N., Basu, K., and Das, S. 2004. Design and performance analysis of a dynamic hybrid scheduling for symmetric environment. *IEEE Intl. Workshop on Mobile Adhoc Networks*.

Saxena, N., Basu, K., and Pinotti, S. K. D. M. C. 2005. A prioritized hybrid scheduling for two different classes of clients in asymmetric wireless networks. In *24th IEEE International Performance Computing and Communications Conference (IPCCC)*. St. Louis, MO.

Saxena, N. and Pinotti, M. 2004. Performance guarantee in a new hybrid push-pull scheduling algorithm. In *Third International Workshop on Wireless Information Systems (WIS),*.

Sichitiu, M. and Kihl, M. 2008. Inter-vehicle communication systems: A survey. *IEEE Comm. Surveys and Tutorials 10,* 2, 88–105.

VII. The vehicle infrastructure integration. `http://www.its.dot.gov/vii`.

Xu, J., Tang, X., and Lee, W.-C. 2006. Time-critical on-demand data broadcast: algorithms, analysis, and performance evaluation. *Parallel and Distributed Systems, IEEE Transactions on 17,* 1 (jan.), 3 – 14.

Yang, X., Zhao, F., and Vaidya, N. 2004. A vehicle-to-vehicle communication protocol for cooperative collision warning. In *Proc.of First Annual International Conference on Moble and Ubiquitios Systems*. Boston,MA, 114–123.

Yee, W. G. and Navathe, S. B. 2003. Efficient data access to multi-channel broadcast programs. In *Proceedings of the twelfth international conference on Information and knowledge management.* CIKM '03. ACM, New York, NY, USA, 153–160.

Zhang, Y., Zhao, J., and Cao, G. 2007. On scheduling vehicle-roadside data access. In *Proceedings of the fourth ACM international workshop on Vehicular ad hoc networks*. VANET '07. ACM, New York, NY, USA, 9–18.

Zhang, Y., Zhao, J., and Cao, G. 2010. Roadcast: a popularity aware content sharing scheme in vanets. *SIGMOBILE Mob. Comput. Commun. Rev. 13,* 4 (Mar.), 1–14.

Ajay Guleria is a Sr. System Manager with Panjab University Chandigarh and a research scholar at the Department of Computer Science and Engineering at the National Institute of Technology, Hamirpur. His research interests pertain to issues of data dissemination in VANET's. He is working under the joint supervision of Professor Narottam Chand and Professor Lalit Awasthi at NIT Hamirpur.

Dr. Narottam C. Kaushal received his Ph.D. degree from IIT Roorkee in Computer Science and Engineering. Previously he received M.Tech. and B.Tech. degrees in Computer Science and Engineering from IIT Delhi and NIT Hamirpur, respectively.
Presently he is working as Associate Professor, Department of Computer Science and Engineering, NIT Hamirpur. He has served as Head, Department of Computer Science & Engineering, during Feb 2008 to Jan 2011 and Head, Institute Computer Centre, during Feb 2008 to July 2009.
His current research areas of interest include mobile computing, mobile ad hoc networks and wireless sensor networks. He has published more than 120 research papers in International/National journals & conferences and guiding six PhDs in these areas. He is member of IEEE, ISTE, CSI, International Association of Engineers and Internet Society.

Lalit Kumar Awasthi is a Professor at the Department of Computer Science & Engineering, National Institute of Technology, Hamirpur, India since 2003. He has completed his M.Tech. at Computer Science and engineering Department, Indian Institute of Technology, Delhi in 1993 and his Ph.D. from Indian Institute of Technology, Roorkee, India. His current areas of interest include mobile computing, cloud computing, mobile e-commerce, fault tolerance in distributed systems and P2P networks. He has over 22 years of teaching experience and has been teaching B.Tech and M.Tech courses, besides supervising ten PhD students currently. He received Best Faculty Award in CSE, NIT Hamirpur, Himachal, India for the year 2007-08. He is "Fellow Member" of The Institute of Engineers (India), Kolkata, India and Life Member of Computer Society of India. He is member of editorial teams of several Journals and is associated with many conferences. He has ninety publications to his credit in international and national journals and conferences of repute. He has been reviewer of journals such as IEEE Transactions, Elsevier Journal on AD-HOC Networks and served on the technical programme committees of many International Conferences.