Request Analysis and Dynamic Queuing System for VANETs

Ajay Guleria
Department of Computer Centre
PU Chandigarh, India

Narottam Chand
Department of CSE
NIT Hamirpur, India

Mohinder Kumar
Department of CSE
NIT Hamirpur, India

Lalit Awasthi
Department of CSE
NIT Hamirpur, India

Abstract—**Vehicular Ad hoc Network (VANET) is a kind of mobile ad hoc network using the capabilities of wireless communication for Vehicle-to-Vehicle and Vehicle-to-Roadside communication to provide safety and comfort to vehicles in transportation system. People in vehicles want to access data of their interest from Road Side Unit (RSU). RSU need to schedule these requests in a way to maximize the service ratio. In this paper we have proposed new methods for careful analysis of incoming requests to find whether these requests can be completed within deadline or not and to provide dynamic service queue. Simulation results show that the proposed schemes increase the service ratio significantly.**

Keywords- Road Side Uni; Service Ratio; Propagation Delay; Service Queue.

I. INTRODUCTION

Vehicular ad hoc network (VANET) is emerging as important technology to provide safety and comfort to vehicles in transportation system. It is special type of mobile ad hoc network with highly mobile nodes. Federal Communications Commission has allocated 5.850-5.925 GHz portion of the spectrum for inter-vehicle communication (IVC) and vehicle to roadside communication (VRC) [1, 2]. VANETs are not purely mobile ad hoc network; they have fixed points called Road Side Units (RSUs) to provide services to vehicles. RSU can provide safety, local news and advertisement, music, radio, video, etc. [3, 4, 5, 6]. Applications of VANET can be broadly classified into two categories.

A. *Safety related applications*: accident related alerts, red light warning, etc.

B. *Non-safety related applications:* these applications include downloading audio/video programs, digital map, internet related services, traffic information, weather forecast and other communication applications.

Some of the major challenges for communication in VANETs are high mobility, dynamically changing topology, sparsely located nodes and very short duration of communication. So, serving the requested data items to vehicles before it goes outside the coverage of RSU is very important. This paper makes the following contributions:

We have proposed an algorithm to check whether the incoming request can be completed before its deadline or not. Proposed an algorithm to provide dynamic service queuing to work efficiently under variable density of traffic.

Conducted simulation to evaluate the performance of VANETs when performing operations with both proposed algorithms.

Rest of the paper is organized as follows. Section II discusses related work in this field. Section III describes system model. Section IV proposes algorithms for request analysis and dynamic queuing system.Section V talks about simulation environment and results.Finally Section VI is devoted to concluding remarks.

II. RELATED WORK

Major research challenges in VANETs are introduced in [7, 8 and 9]. 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 algorithm have been proposed to reduce the waiting time [10, 11,12].

In [13] Acharya and Muthukrishnan proposed a data scheduling algorithm called longest total stretched first (LTSF) which is based on a new metric called stretch i.e. service ratio of the response time of a request to its service time. LTSF optimizes stretch and maintains balance between worst case and average case but implementation of LTSF for large system is not practical because server needs to recalculate stretch for every data item with pending request, to find next data to be broadcasted.

In [14] Xu et al. 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 in [15] proposed periodic push based broadcast, which is not well suited to VANET applications.

In [16] Zhang et al. proposed a vehicle platoon aware data access called V-PADA. In this scheme vehicles contribute their part of buffer to replicate data for others in the same platoon and share data with others. When vehicle leaves a platoon it prefetches interested data and transfers its buffered data to other vehicles in advance so that they can still access the data after it leaves.

V-PADA consists of two components: first a vehicle platooning protocol to identify platoon formation and platoon splitting by using stochastic time series analysis, second a data management to guide platoon members to replicate and fetchmost suitable data to achieve high availability and low control overhead.

In [17] Zhang et al. 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 the 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 results show that the proposed algorithms significantly improve the service ratio.

III. STSTEM MODEL

In vehicular ad hoc networks there are two communicating entities i.e. vehicles and roadside unit (RSU). Each vehicle in VANET is equipped with On Board Unit (OBU) which has transceiver, computational power and omnidirectional antenna. RSU has transceiver, antenna, processor, sensors. RSU manages data and provides services to vehicle. Vehicles use services provided by RSU. Communication can be either inter-vehicle or vehicle-to-infrastructure. Fig. 1shows various steps involved in communication between vehicle and RSU.

Operation sequence of VANETs can be summarized in Fig. 2.

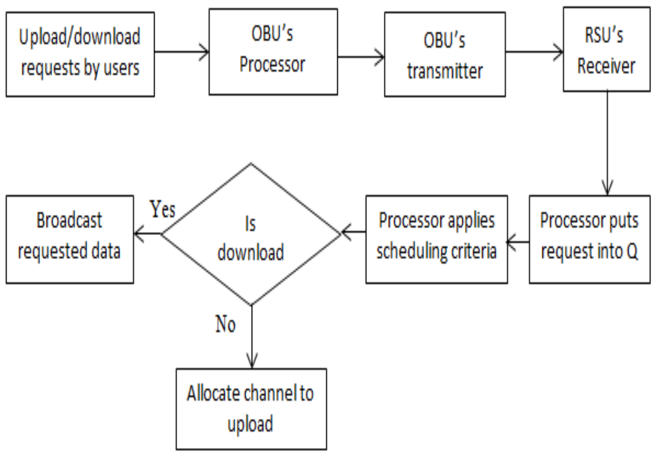
Upload/download requests by users
OBU's Processor
OBU's transmitter
RSU's Receiver
Processor puts request into Q
Processor applies scheduling criteria
Is download
Yes
Broadcast requested data
No
Allocate channel to upload

Figure 1.Various steps in communication between vehicleand RSU.

RSU is generally placed at intersections to increase service ratio and safety. We have also assumed that RSU is placed at intersection and vehicles are synchronized with Global Positioning System(GPS). Further, we have assumed two lanehighways for simplicity. The proposed scheme is applicable to multilane highways without any major change. OBU on vehicle is capable of calculating average speed of vehicle for last few minutes.

A request from vehicle is represented by five tuple as given below:

<VEHICLE_ID, DATA_ID, AVG_SPEED, CURRENT_LOCATION, OP_CODE>

VEHICLE_ID: Unique Identity of Vehicle.

DATA_ID: Unique Identification code of requested data.

AVG_SPEED: Average Speed of vehicle.

CURRENT_LOCATION: Current Location of vehicle.

OP_CODE: Operation Code i.e. either upload or download.

It is assumed that RSU maintains single queue for upload and download requests.RSU is capable of determining CURRENT_LOCATION. Hence it can find the direction of movement of vehicle.

IV. PROPOSED SCHEMES

In this section we have proposed two independent algorithms PROCESS_REQUEST to find whether incoming request can be completed before its deadline or not and SRBAQS to provide dynamic queuing system.

As shown in Fig. 2, our first proposed algorithm PROCESS_REQUEST uses following notations:

LNS: Lane from north to south.

LSN: Lane from south to north.

LWE: Lane from west to east.

LEW: Lane from east to west.

(XN, YN): Location of last coverage point of RSU in north.

(XS, YS): Location of last coverage point of RSU in south.

(XE, YE): Location of last coverage point of RSU in east.

(XW, YW): Location of last coverage point of RSU in west.

(XI, YI): Point of intersection of horizontal and vertical dividers.

(XR, YR): Reference point (discussed below).

(XV, YV): Current location of vehicle.

Size(i): Size of data item requested in current request.

t: transfer rate of data from RSU to vehicle.

ΔT: Average propagation delay of request from vehicle to RSU.

TTRANSFER: Time required for transferring requested data to vehicle.

TLIFE: Connection life time i.e. duration for which vehicle will remain in range of RSU.

Reference point: It is last coverage point in direction of movement of vehicle when vehicle requested the data. Even if vehicle takes a left or right turn, reference point will remain same i.e. change in direction of movement after request has been sent to RSU does not make any effect on the proposed scheme.

PROCESS_REQUEST uses procedure REFERENCE_POINT to find the reference point. REFERENCE_POINT procedure is as below.

```
1.  REFERENCE_POINT($X_V$, $Y_V$)
2.    If ($X_V$, $Y_V$) ϵ $L_{NS}$
3.      then return ($X_S$, $Y_S$)
4.    If ($X_V$, $Y_V$) ϵ $L_{SN}$
5.      then return ($X_N$, $Y_N$)
6.    If ($X_V$, $Y_V$) ϵ $L_{EW}$
7.      then return ($X_W$, $Y_W$)
8.    If ($X_V$, $Y_V$) ϵ $L_{WE}$
9.      then return ($X_E$, $Y_E$)
```

In procedure REFERENCE_POINT lines 2 and 3 return reference point as (XS, YS) if vehicle is in lane from north to south i.e. moving from north to south direction. Lines 4 and 5 return reference point as (XN, YN) if vehicle is in lane from south to north i.e. moving from south to north direction. Lines 6 and 7 return reference point as (XW, YW) if vehicle is in lane from east to west i.e. moving from east to west direction. Similarly lines 8 and 9 return reference point as (XE, YE) if vehicle is in lane from west to east i.e. moving from west to east direction.

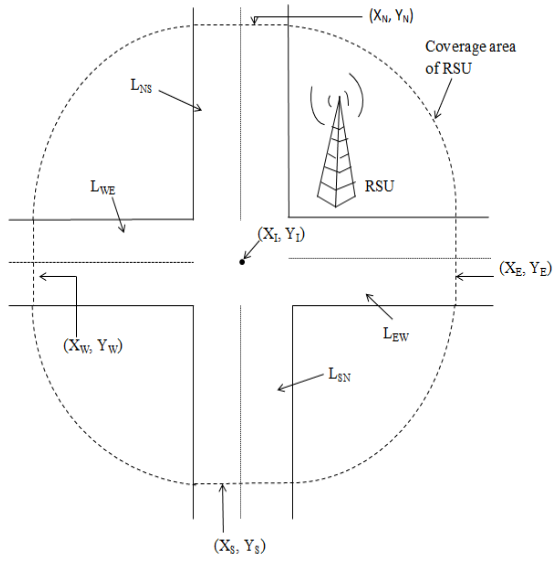

Figure 2. System model and notations.

```
1.  PROCESS_REQUEST(XV, YV, AVG_SPEED)
2.  (XR, YR)=REFERENCE_POINT(XV, YV)
3.  d1 = √((XR-XV)2+(YR-YV)2)
4.  if ((XR==XS)||(XR==XN))
5.  d2 = √((XI-XW )2+(YI-YW)2)+ √((XI-XV)2+(YI-YV)2)
6.    d3 = √((XI-XE )2+(YI-YE)2)+ √((XI-XV)2+(YI-YV)2)
7.    if ((XR==XW)||(XR==XE))
8.    d2 = √((XI-XN )2+(YI-YN)2)+ √((XI-XV)2+(YI-YV)2)
9.    d3 = √((XI-XS)2+(YI-YS)2)+ √((XI-XV)2+(YI-YV)2)
10.   d=min {d1, d2, d3}
11.   di =d-(ΔT * AVG_SPEED)
12.   TLIFE=di/AVG_SPEED
13.   TTRANSFER=Size(i)/t
14.   if (TLIFE >=TTRANSFER)
15.   then put request in Queue
16.   else
17.   reject the request
```

Algorithm PROCESS_REQUEST at RSU determines whether the coming request can be completed within deadline or not. This algorithm takes current location of vehicle i.e. (XV, YV) and AVG_SPEED as input. Other values including Size(i) are known to RSU.

Line 2 calls the procedure REFERENCE_POINT on current location of vehicle to find the reference point. In line 2 (XA, YA) = (XB, YB) implies XA = XB and YA= YB. Line 3 computes distance between current location of vehicle and reference point, it is case when vehicle does not take any turn i.e. distance between (Xv, YV) and reference point computed in step 2 i.e. (XR, YR). Line 4 checks whether vehicle is moving in LNS or LSN i.e. vehicle can take turn to either west or eastside. If result of line 4 is true then line 5 computes total distance to be travelled by vehicle if it takes turn to west and line 6 computes total distance to be travelled by vehicle if it takes turn to east.

Line 7 checks whether vehicle is moving in LWE or LEW i.e. vehicle can take turn to either north or south side. If result of line 7 is true then line 8 computes total distance to be travelled by vehicle if it takes turn to north and line 9 computes total distance to be travelled by vehicle if it takes turn to south.

Line 10 finds the minimum distance which can be travelled by vehicle without going outside the coverage of RSU. Line 11 reduces distance to be travelled by vehicle, by the distance vehicle has travelled till its request reaches the RSU. Line 12 finds the time period for which vehicle will remain under coverage of RSU. Line 13 finds the time to be taken to transfer data to vehicle by dividing the requested data size by transfer rate. Line 14 checks whether data can be transferred to vehicle before it goes out coverage of RSU, if yes, line 15 puts this request in Queue else line 17 rejects the request.

Service Ratio Based Adaptive Queuing System (SRBAQS)

Traffic density varies on road during day and night, on working days and holidays also if there is jam on other routes. If size of service queue is kept static then, there can be following problems:

1. If service queue size at RSU is very large then it will take very long time to start the scheduling process, because RSU will wait for service queue to become full. Even when there is limit on waiting time, it will

affect the service ratio significantly as smaller deadlines are major constraint in VANETs.

2. If service queue size is very small it will start the scheduling process too early. It can reduce the service ratio e.g. let size of service queue is two and first two requests arrived are of very large size. Then it can block subsequent smaller size requests.

SRBAQS algorithm avoids this scenario by making service queue size dynamic depending on the service ratio. It uses following inputs:

λ: Service ratio of past x requests

x: number of requests processed in past

n: Service queue size

s: Number of requests successfully served, initially s=0

SC: Service ratio for current service cycle

λn: New service ratio after completion of current service cycle

SIZE(Q): Number of requests in queue

Success (requesti): Returns true if requesti has been completed successfully

```
1.  SRBAQS ($\lambda$, x, n)
2.  while(true)
3.  while(SIZE(Q)!=n)
4.  if  new request has arrived put request in
    service queue Q
5.  for i$\leftarrow$0 to n do
6.  process requesti according to scheduling
    criteria
7.  if Success(requesti)=true
8.  then  s=s+1
9.  SC=s/n
10. $\lambda$n= ($\lambda$x + SCn)/(x+n)
11. x=x+n
12. If (SC>$\lambda$)
13. then n=n+ n/2
14. else
15. n=n-n/2
16. $\lambda$=$\lambda$n
```

NOTE: loops in SRBAQS are nested according to indentation.

Line 1of this algorithm shows that algorithm SRBAQS takes service ratio in past, number of requests processed in past and initial service queue size as input. Initial service queue size can be 1 or 2 whereas other two parameters are computed in past for very first service cycle these two can be any appropriate values. Line 2 of this algorithm shows that RSU will repeat same set of activities. Line 3 and 4 make service queue full before starting scheduling process. Lines from 5 to 8 process the service queue according to scheduling criteria and variable s counts the number of successful requests. Scheduling criteria in Line 6 can be

FCFS (First Come First Serve): Requests in service queue are served in sequence of their arrival.

EDF (Earliest Deadline First): Requests are served in increasing sequence of their deadline. Request having earliest deadline will be served first.

SDF (Smallest Data size First): Request having smallest data size will be served first and so on.

Lines 7 and 8 counts number of requests successfully served in current service cycle. Line 9 computes the service ratio of last service cycle. Line 10 computes the new service ratio. Line 11 computes total number of requests served including requests served in past. Lines 11 to 14 check if service ratio in last service cycle is greater than previous service ratio then increases service queue size by fifty percent else reduces the size of service queue by fifty per cent. Line 14 assigns new service ratio to current service ratio. After each iteration of while loop in line 2 s is initialized to zero.

Theorem: the proposed algorithm PROCESS_REQUEST computes the minimum distance to be travelled by the vehicle. Proof: Let the vehicle is moving on LWE and has requested some data then there can be two cases:

Case 1: vehicle is at some position from which it can take either a turn or move straight as shown in Fig. 3(a).

Case 2: vehicle cannot take any turn i.e. it moves only straight as shown in Fig. 3(b).

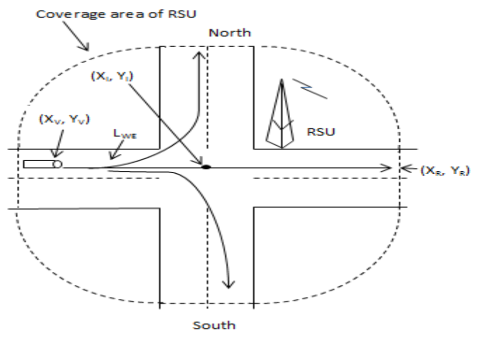

Figure 3(a). Case 1 when vehicle can take turn.

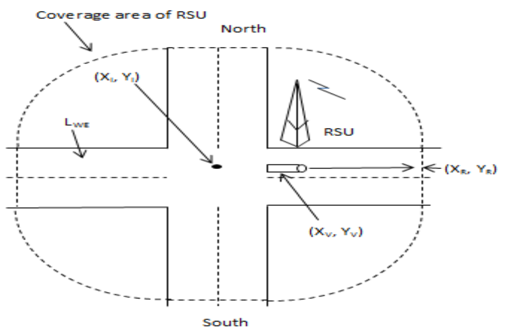

Figure 3(b). Case 2 when vehicle cannot take turn.

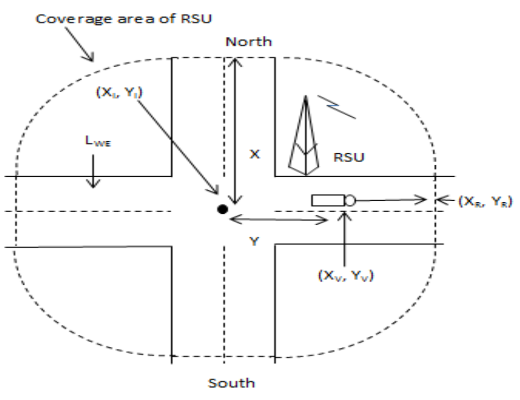

Figure 3(c). Path of d2.

In case 1, PROCESS_REQUEST computes distances d1, d2, d3 where:d1 is distance to be travelled by vehicle if it moves straight, d2 is distance to be travelled by vehicle if ittakes a turn to north, d3 is distance to be travelled by vehicle if it takes turn to south.

Proposed algorithm choses minimum of d1, d2 and d3. Hence distance to be travelled by vehicle is minimum distance.

In case 2, the proposed algorithm computes d1, d2, d3 although vehicle can move only straight.

$d2 = \sqrt{(XI-XN)2 + (YI-YN)2} + \sqrt{(XI-XV)2+(YI-YV)2}$

where $\sqrt{(XI-XV)2+(YI-YV)2}$ is distance from vehicle to point of intersection. Let it is Y. See Fig. 3(c).

And $\sqrt{(XI-XN)2+(YI-YN)2}$ is distance from point of intersection to north direction let it is X. See Fig. 3(c).

So, d2=X+Y

even for very small value of Y

$$d2 < d1 \quad (1)$$

d2 can be greater than d1 iff RSU is placed at some place other than intersection.

$$\text{Similarly, } d3<d1 \quad (2)$$

From Equation (1) and (2), we can say that d1 is minimum distance to be travelled by vehicle.

V. SIMULATIONENVIRONMENT AND RESULTS

We have simulated the proposed algorithms i.e. PROCESS_REQUEST and SRBAQS using Dev-C++ 4.9.9.2. The system model has been implemented by appropriately and randomly generating request ids, data ids, operation codes, data size, deadlines and other parameters. We have tested the performance of both algorithms over three data scheduling algorithms discussed earlier i.e. FCFS (First Come First Serve), EDF(Earliest Deadline First), SDF(Smallest Data size First). For evaluation of these algorithms, we have used Service ratio as performance metric i.e. ratio of number of requests served successfully to total number of requests. In each graph Y axis denotes the service ratio and X axis number of times window i.e. service queue processed.

Fig.4 show the performance of FCFS before and after implementing PROCESS REQUEST. Fig.5 and 6 show the performance of EDF and SDF respectively before and after implementing PROCESS_REQUEST.

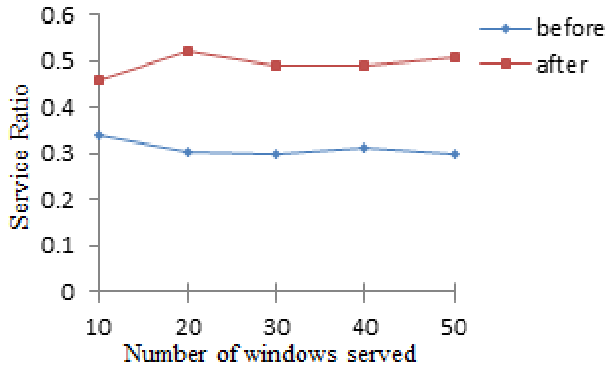

Figure 4. Performance of FCFS before and before and after implementing PROCESS_REQUEST.

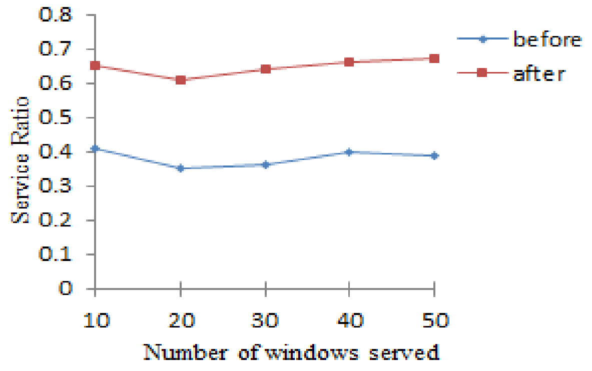

Figure 5. Performance of EDF before and after implementing PROCESS_REQUEST.

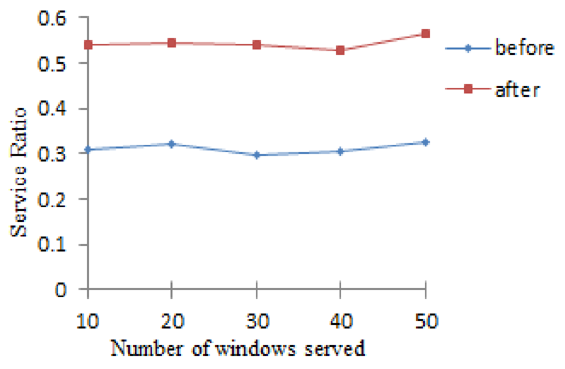

Figure 6. Performance of SDF before and after implementing PROCESS_REQUEST.

Fig. 4, 5 and 6 show that the PROCESS_REQUEST algorithm improves the performance of scheduling algorithms significantly. This improvement is achieved because the PROCESS_REQUEST eliminates the data access requests which cannot be completed within their deadline therefore avoiding the wastage of time on unnecessary processing time of these requests.

Fig. 7 shows the performance of FCFS before and after implementing both PROCESS_REQUEST and SRBAQS.

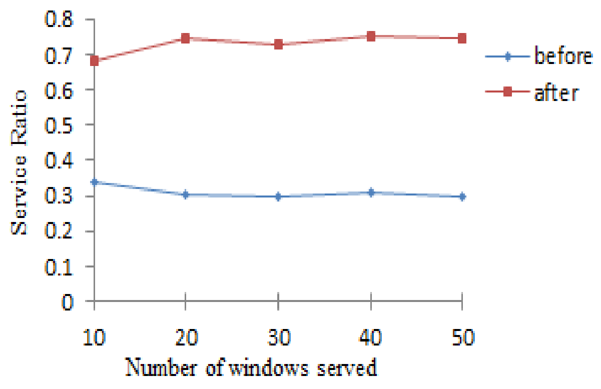

Figure 7. Performance of FCFS before and after implementing both PROCESS_REQUEST and SRBAQS.

Fig. 8 and 9 show the performance of EDF and SDF before and after implementing both PROCESS_REQUEST and SRBAQS.

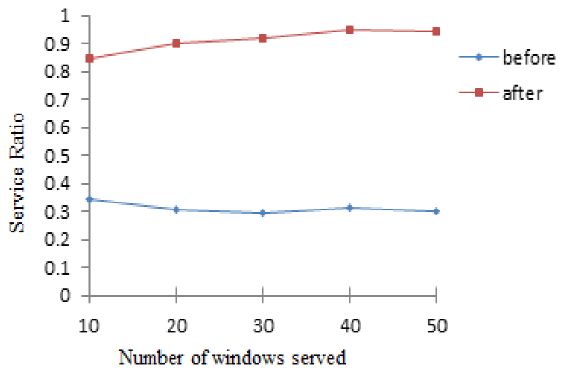

Figure 8. Performance of EDF before and after implementing both.

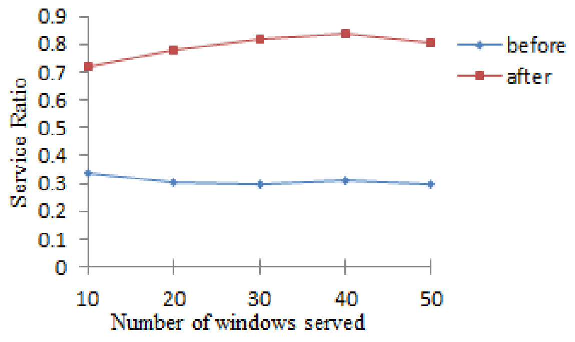

Figure 9.Performance of SDF before and after implementing both PROCESS_REQUEST and SRBAQS.

Fig. 7, 8 and 9 show that implementation of both PROCESS_REQUEST and SRBAQSfurther improve the performance. This performance gain is because PROCESS_REQUEST avoid the processing of unnecessary requests and SRBAQS make the queue size dynamic. Hence simulation results show that PROCESS_REQUEST and SRBAQS significantly improve the performance of data scheduling algorithms.

VI. Conclusion

In this paper we have proposed two algorithms PROCESS_REQUEST and SRBAQS. PROCESS_REQUEST checks whether the incoming request can be served before its deadline or not. If request cannot be completed within its deadline then it is not inserted in service queue hence rejected. It improves the processing time of other requests and service ratio. Static queue size causes various problems such as long waiting time if traffic density is very low and poor performance of scheduling algorithm if traffic density is very high. SRBAQS solves both the problems by providing dynamic queue size i.e. queue size varies depending upon service ratio. Simulation results show that both algorithms improve the performance of data scheduling algorithms FCFS, EDF and SDF.

In future, we will take other issues into consideration on scheduling such as different types of data items, multiple queues, etc. Further, other unique challenges in VANETs such as routing, clustering, data caching will motivate further research in this area.

References

[1] IEEE Standard for Information Technology-Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications Amendment 8: Medium Access Control (MAC) Quality of Service Enhancements," IEEE Std. 802.11e-2005(Amendment to IEEE Std. 802.11, 1999 Edition (Reaff 2003), 2005.

[2] "IEEE 802.11p D3.0," in IEEE StandardActivitiesDepartment, July 2007.

[3] J. Yin, T. Eibatt, G. Yeung, B. Ryu, S. Habermas, H. Krishnan, and T. Talty, "Performance Evaluation of Safety Applications over DSRC Vehicular Ad hoc Networks,"in Proc. of the 1st ACM int. workshop on Vehicular ad hoc network, Oct. 1-1, 2004, pp.1-9.

[4] X. Yang, J. Liu, F. Zhao, and N. Vaidya, "A Vehicle-to-Vehicle Communication Protocol for Cooperative CollisionWarning," in Proc. of 1st Int. Conf. on Mobile and Ubiquitous Systems: Networking and Services, Aug. 22-26, 2004, pp. 114-123.

[5] T.Mak, K. Laberteaux, and R. Sengupta, "Multi ChannelVANET providing concurrent Safety and Commercial Services," in Proc. of the 2nd ACM int. workshop on Vehicular ad hoc networks, Sep. 02-02, 2005, pp. 1-9.

[6] S. B. Lee, G. Pan, J.S. Park, M. Gerla, and S. Lu,"Secure Incentives for Commercial Ad Dissemination in Vehicular Networks," inProc. of the 8th ACM int. symp. on Mobile ad hoc networking and computing, Sept. 9-14, 2007, pp. 150-159.

[7] K. Daniel, Wong, K. Tepe, W. Chen and M. Gerla, "Inter-Vehicular Communications,"in IEEE Wireless Communications, Vol.13, No. 5, pp. 6-7, Oct. 2006.

[8] R. A. Santos, A. Edwards and O. Alvarez, "Towards an Inter vehicle Communication Algorithm," The 3rd International Conference on Electrical and Electronics Engineering, September 6-8, 2006, pp. 1-4.

[9] M. Durresi, A. Durresi and L. Barolli, "Adaptive Inter Vehicle Communications," International Journal of Wireless Information Networks,Vol. 13, No. 2, pp. 151-160, April 2006.

[10] C. Su and L. Tassiulas, "Broadcast scheduling for information distribution," in Proc. of 16TH International Conference on Computer Communications, April 7-12, 1997, Vol. 1, pp. 109-117.

[11] D. Aksoy and M. Franklin, "R*w: a scheduling approach for large scale on-demand data broadcast," IEEE/ACM Transactions on Networking, Vol. 7, No. 6, pp. 846–860, Dec. 1999.

[12] R .Gandhi, S. Khuller, Y. Kim and Y. Wan, "Algorithms for minimizing response time in broadcast Scheduling," ALGORITHMICA, Vol. 38, No.4, pp. 597–608, 2004.

[13] S. Acharya and S. Muthukrishnan, "Scheduling On DemandBroadcasts: New Metrics and Algorithms,"in Proc. of the 4th annual ACM/IEEE int.conf. on Mobile computing and networking, Oct. 25-30, 1998, pp. 43-54.

[14] J. Xu, X. Tang and W. Lee, "Time-critical On-demand Data Broadcast: Algorithms, Analysis, and Performance evaluation," IEEE Transaction on Parallel Distributed Systems, Vol. 17, No. 1, pp. 3–14, 2006.

[15] S. Jiang and N. Vaidya, "Scheduling data broadcast to impatientusers," inProc. of the 1st ACM int. workshop on Data engineering for wireless and mobile access, Aug. 20-20, 1999, pp. 52–59.

[16] Y. Zhang and G. Cao, "V-PADA: Vehicle Platoon Aware Data Access in VANETs," IEEE transactions on vehicular technology, Vol.60, No. 5, pp. 2326-2339, June 2011.

[17] Y. Zhang, J. Zhao and G. Cao, "Service Scheduling of Vehicle-Roadside Data Access," in international journal of Mobile Networks and Application.,Vol. 15, No. 1, pp. 83-96, Feb. 2010.