1. Introduction

VANET is a hot topic in recent years; it is proposed to enhance driving safety by propagating traffic information among driving vehicles. Although great progress has been made in data routing, VANET still cannot find enough convincing applications. Modern vehicles are equipped with various sensors, the sensors are used to monitor the states of vehicle and surroundings. The information vehicular sensors collected would be very useful for some crowdsourcing applications. For example, fuel tank sensor gives fuel level value; if the fuel level increases at a location, then the vehicle probably is refueling and the location is a fuel station; a vehicle can request the locations of fuel station from its neighbors so that it knows where to refuel without the help of on-line GIS system. Vehicular sensors enrich the information VANET could propagate; therefore, the combination of wireless sensors and VANET to construct vehicular sensor network (VSN) or vehicular ad hoc and sensor network (VASNET) [1] is a good complement of VANET.

There are many good routing algorithms in VANET can be transplanted to VSN, like GPSR [2] and VADD [3], because VSN has the same network architecture as VANET. However, because vehicular sensors will collect a large amount of data, VSN needs a routing algorithm which can handle this amount of data. In most cases, routing algorithm works at network layer, it simply forwards the packet passed down from upper layers to the next hop according to routing algorithm. The packet is usually very small; therefore, the assumption that a packet can be delivered completely to the next hop in VANET is correct. But, if we look at the data delivery in the perspective of data itself and combined with the characteristics of VSN, we find that existing data delivery algorithms will cause the big amount of data to stuck at corners or crossroads and cause big time and resource waste.

There is no problem in communication of V2V or V2I in a straight road; however, when communication meets corners, especially in city environment, time will be a problem. As Figure 1 shows, there are many obstacles by the road side, like buildings, trees, and crowds, the wireless communication signals will be blocked by these obstacles. If a vehicle has to deliver data to its left direction at crossroad or corner, the time for delivery is \( \frac{d}{v} \), \( d \) is the width of road, and \( v \) is the velocity of the vehicle driving across the crossroad.
A common lane width is 3.7 meters, for a four-lane road, the road width is nearly 15 meters. Suppose \( v \) equals 40 km/h, then the time for data delivery is 1.36 seconds. Even we do not consider wireless channel competition and conflict and set the data link bandwidth to 1 MB/s, the vehicle can only transmit 1.36 MB data. For a large bulk of data which is at least 10 MB, it needs eight times for transmitting in an ideal situation, actually it will cost much more using the existing routing algorithm in VANET.

Figure 2 shows how existing routing algorithms, like VADD, handle the corner data delivery. Red block that represents a data block can be completely delivered when a vehicle driving across a corner. Figure 2(a) shows that S1 has three red blocks to deliver to D, but it can only deliver one block to D when it drives across the crossroad; Figure 2(b) shows that S1 recalculates a route to reach D, and it will deliver all the left two blocks to S2, because S2 is heading to the crossroad (the heading direction can be detected by various sensors like accelerometer, orientation) and S2 is closest to the crossroad compared to other vehicles which S1 can communicate with; S2 can only deliver one block to D, as Figure 2(c) shows, S2 recalculates a route to D and passes the left one block to S3; Figure 2(d) shows that S3 delivers the last block to D. To calculate time cost and computation cost, we suppose the time of sending one block is \( T_{\text{block}} \), the time of calculating a routing path is \( T_{\text{route}} \), the average time of driving to the crossroad is \( T_{\text{drive}} \), the computation cost of sending one block is \( C_{\text{send}} \), the computation of receiving and storing a block is \( C_{\text{recv}} \), and the computation of calculating a route is \( C_{\text{route}} \). The time cost of data delivery in Figure 2 is \( 6T_{\text{block}} + 3T_{\text{drive}} + 2T_{\text{route}} \), computation cost is \( 6C_{\text{send}} + 3C_{\text{recv}} + 2C_{\text{route}} \), here we ignore the time and computation cost of D.

The idea of solving this problem is simple: we split the data into equal blocks, each block can be completely delivered when a vehicle driving across a crossroad or corner, the blocks are distributed to the neighbors of source vehicle which are also going to drive across the same crossroad or corner, each neighbor will receive one data block. In this case, the data will be completely delivered to destination in a single round. If we use Figure 2 as an example, the time cost of our solution is \( 3T_{\text{block}} + T_{\text{drive}} \), the computation costs is \( 5C_{\text{send}} + 2C_{\text{recv}} \cdot T_{\text{drive}} \) is much bigger than the other two time cost, our solution only have one \( T_{\text{drive}} \) and the existing solutions has three. The improvement is obvious and the algorithm will be discussed in detail in the next section.

The rest of paper is organized as follows. Section 2 will discuss the details of our solution, Section 3 will give results of evaluations, related works and conclusions will be given in Sections 4 and 5.

### 2. Distribution-Based Data Delivery at Corners

In this paper, we propose a distribution-based data delivery at corners (DBDDC), it split large amount of data into small blocks and distribute them to several vehicles which are driving to the same corner or crossroad; therefore, it can deliver all the data blocks to the destination successively and need not any recalculation of routing path.

The idea of DBDDC is simple, however, it has some challenges to handle: (1) how to recognize a vehicle driving to the same corner or crossroad as that of source vehicle. (2) how to count the number of qualified vehicles to make sure there are enough vehicles to carry all the blocks.

#### 2.1. Challenges

To know whether a vehicle is driving to the same crossroad as that of source vehicle, we need to know two things: the driving direction and the segment of road the vehicle is at. The direction is easy to be obtained, accelerometer sensor or orientation sensor embedded in vehicles or mobile phones give three-dimensional readings: Azimuth, Pitch, Roll. Take accelerometer for example, Azimuth is the acceleration on the \( z \)-axis, Pitch is the acceleration on the \( x \)-axis, and Roll is the acceleration on the \( y \)-axis; therefore, we can judge the direction of driving according to the coordinate system of accelerometer. If a vehicle is heading to the same direction as that of source vehicle and they are in the same segment of road, then we can judge that the vehicle must be driving to the same crossroad as that of the source vehicle. As Figure 3 shows, when N1, N2, N3, and S are in the same segment between crossroad 1 and 2, then they are all heading to crossroad 2; in this paper, we call these vehicles distribution nodes. A vehicle can judge whether it is between crossroad 1 and 2 through comparing its GPS location with the locations of crossroads 1 and 2.

To count the number of distribution nodes, the source vehicle will broadcast a beacon message for requesting the distribution nodes to reply their IDs. As Figure 4 shows, the neighbor vehicles in the communication range of source vehicle will all receive a beacon packet, the vehicles will judge whether they are distribution nodes, only the distribution nodes will send replies to source vehicle with their IDs and broadcast the received beacon for those distribution nodes out of communication range of source vehicle, also the distribution nodes will relay the received replies from other distribution nodes to the source vehicle. After a short time,
Figure 2: Data delivery at crossroads using existing routing algorithm.

Figure 3: The vehicles which will drive across the same crossroad as source vehicle does.

Figure 4: The process of detecting distribution nodes.

Figure 2: Data delivery at crossroads using existing routing algorithm.

Figure 3: The vehicles which will drive across the same crossroad as source vehicle does.

Figure 4: The process of detecting distribution nodes.

2.2. Algorithm. Figure 5 shows that there are four cases in DBDDC. Figure 5(a) is the case that before source vehicle S drives across the crossroad, there are enough distribution nodes for carrying all data blocks. Figure 5(b) is the case when there are not enough distribution nodes and source vehicle already turned left or right, so the undelivered data will be delivered in a straight road. Figure 5(c) is the case when there are not enough distribution nodes and source vehicle drives straight and already crossed the crossroad, the undelivered blocks will be sent to the vehicles driving in a reverse direction; in this paper, we call these vehicles reverse distribution nodes. There are enough reverse distribution nodes in this case. Figure 5(d) is the case when there are not enough distribution nodes and reverse distribution nodes, source vehicle has to carry the undelivered blocks and drive to the next crossroad, when source vehicle approaches the crossroad, it will recalculate a route and do the DBDDC again. Suppose the driving direction values are straight, left, and right; the number of distribution nodes is \( N_d \); the number of reverse distribution nodes is \( N_{rd} \); the block size which can be completely delivered when a vehicle driving across a crossroad is \( Blk \); the data size is \( Data \), then the four cases can be described by

\[
\text{Case} = \begin{cases} 
\text{Case 1} : & \text{direction} = \text{straight} \quad \text{and} \quad \frac{Data}{Blk} \leq N_d, \\
\text{Case 2} : & \text{direction} = \text{left or right} \quad \text{and} \quad \frac{Data}{Blk} > N_d, \\
\text{Case 3} : & \text{direction} = \text{straight} \quad \text{and} \quad \frac{Data}{Blk} \leq (N_d + N_{rd}), \\
\text{Case 4} : & \text{Otherwise}. 
\end{cases}
\]  

(1)

Case 1. If source vehicle is approaching a crossroad and data will be delivered at the crossroad to its left/right direction, it will judge whether case 1 is satisfied (see Algorithm 1).

Case 2. If source vehicle already turned left/right, the data will be delivered in a straight road mode which is easy and implemented by existing routing algorithm (see Algorithm 2).

Case 3. If source vehicle already passed the crossroad, it will use Algorithm 3.

Case 4. If source vehicle has not delivered all data to the destination crossroad and is approaching another crossroad, it will recalculate a new route (see Algorithm 4).

Until now, we can give the DBDDC algorithm. DBDDC is not a complete routing and data delivery algorithm, it is a complement of existing routing algorithm, if existing routing algorithms like VADD combines DBDDC, it will achieve better performance at corners and crossroads as Section 3 shows.

3. Evaluation

In this paper, we use simulations to evaluate the performance of DBDDC. We compare the time cost of data delivery
Figure 5: Four cases of DBDDC.

Figure 6: Evaluations of DBDDC and VADD in four cases.
Require: Data;  
Ensure: true or false;  
send beacon to detect the number of distribution nodes;  
read reply list and get the number $N_d$;  
if $Data/Blk \leq N_d$ then  
split Data into blocks of $Blk$ size;  
send each block to a vehicle in the reply list;  
return true;  
else  
$Data_1 \leftarrow Data(1), Data(2), \ldots, Data(N_d)$; //assign the front $N_d$ blocks of data to $Data_1$  
$Data \leftarrow Data - Data_1$;  
split $Data_1$ into blocks of $Blk$ size;  
send each block to a vehicle in the reply list;  
return false;  
end if  

Algorithm 1: Data delivery for Case 1: bool DDC1 (Data).

Require: Data;  
Ensure: true or false;  
use existing routing algorithm for Data delivery in straight road mode;  
return true;

Algorithm 2: Data delivery for Case 2: bool DDC2 (Data).

Require: Data, Crossroad;  
Ensure: true or false;  
if source vehicle is approaching Crossroad then  
if DDC1(Data) = true then  
return true;  
else if source vehicle turned left/right then  
$DDC2(Data)$;  
return true;  
end if  
if DDC3(Data) = true then  
return true;  
else  
recalculate a route;  
return false;  
end if  
end if

Algorithm 3: Data delivery for Case 3: bool DDC3 (Data).

Require: Data;  
Ensure: true or false;  
repeat  
if a new reverse distribution node drive into communication range of source vehicle then  
send one block to the reverse distribution node;  
$Data \leftarrow Data - transmitted block$;  
if $Data = 0$ then  
return true;  
end if  
end if  
until source vehicle is approaching to another crossroad  
return false;

Algorithm 4: DBDDC.

between DBDDC and VADD at crossroad. Suppose there are two crossroads: crossroad 1 and crossroad 2, each road segment can hold 25 vehicles, the average velocity of vehicle is 40 km/h, the average arrival time at a crossroad of a vehicle is 23 seconds, the time a data block can completely be delivered is 2 seconds, there are 25 data blocks to be delivered on source vehicle, source vehicle is driving from west to east and it will deliver the data to the north direction of crossroad 1, and the communication range is 100 meters.

We compare our solution with VADD in four different cases. Figure 6(a) shows a case when there are enough distribution nodes and no reverse distribution nodes, which is 24 vehicles, source vehicle can distribute 24 data blocks to them, and the 25 blocks will be delivered to destination successively; if using VADD, source vehicle will deliver one block, and it cannot find any vehicle on the reverse direction to carry the left blocks back to crossroad 1.

Figure 6(b) shows a case when there are enough distribution nodes and source vehicle turns to north at crossroad 1, in our solution, source vehicle already distributed all the blocks, therefore the time cost does not change compared to Figure 6(a); if using VADD, source vehicle will deliver all the blocks directly to the destination, the time cost is small; however, in our solution, if there are no distribution nodes, the solution will act exactly as VADD does, the time cost is small too.

Figure 6(c) shows a case when there are no distribution nodes and enough reverse distribution nodes, source vehicle is driving straightly. In our solution, source vehicle deliver one block when it is driving across crossroad 1 and pass the left blocks to each reverse distribution nodes, because the average arrival time is 23 seconds, the delay time is nearly 23 seconds; if using VADD, source vehicle will deliver one block during crossroad passing, after 7 seconds, source vehicle will meet the first vehicle V1 on the reverse direction, source vehicle has 9 seconds to deliver data blocks, V1 will receive 4 blocks, and it will deliver 1 block. In a similar way, the second vehicle V2 on the reverse direction will receive 4 blocks and
deliver 1 block, and so on. Finally, 6 vehicles will receive all blocks and 6 blocks will be delivered.

Figure 6(d) shows a case when we set random numbers of distribution nodes and reverse distribution nodes, in this figure there are 8 vehicles for distribution nodes and 9 vehicles for reverse distribution nodes. Source vehicle is driving straightly. In our solution, 9 blocks will be delivered when source vehicle and distribution nodes driving across crossroad 1, 9 left blocks will be delivered to destination by reverse distribution nodes. 7 blocks are left waiting for recalculating route; if using VADD, source vehicle will deliver 4 blocks to V1, V1 will deliver 1 block. If we suppose each vehicle is 11 meters from each other, then V1 will not meet any distribution nodes because they already crossed crossroad 1; therefore, we set each vehicle at 100 meters from each other. V1 will meet the third distribution node D3, it will pass 3 left blocks to D3, D3 will deliver 1 block at crossroad 1, and it will meet the third reverse distribution node V3 and pass the left 1 block to V3. The back and forth process will keep going until no distribution nodes and reverse distribution nodes are driving to crossroad 1.

4. Related Works

VANET was a hot topic and many researches focus on routing algorithms. Table driven is the early version of routing algorithm, it is based on the routing methods in static networks. Each node will discover and maintain a routing table [4–6], because of the high mobility of VANET, discovering and maintaining a routing table become very difficult. A new concept of routing then appeared, it is based on geographic positions. The source node needs to know the location of destination node, every hop of routing, the relay node will choose a next hop which is closest to the destination geographically, it needs not routing tables; therefore, it is suitable in mobile network. Typical routing algorithms are GPSR [2] and GPCR [7]. Vehicles’ driving patterns are constrained by road patterns; therefore, routing along road is more predictable and effective, some typical routing algorithms are VADD [3] and CAR [8].

One reason why there are not enough applications in VANET is that vehicle cannot sense the world and therefore cannot generate enough information propagation in VANET. Some research works take advantage of vehicular sensors and sensors embedded in mobile phones to monitor drivers’ behaviours [9–14], these applications can enrich the quantity of information propagated in VANET and create a new kind of vehicular network, VSN. VASNET [1] is a kind of VSN which combines wireless sensors and VANET; however, it only considers the applications for highway safety. One interesting application for VSN is MobiEyes [15], it uses vehicular cameras for urban monitoring. Video streams are usually very large in data size; therefore, the solution of the corner problem in this paper has its practical meanings.

5. Conclusion

Corner problem in vehicular sensor network is neglected yet important, it will cause data blocks transmitted back and forth at corners or crossroad and cause big time waste and computation waste. Splitting big data into small blocks and distributing them to other vehicles will help solve the problem, and for most times, the data delivery rate is much higher than existing solutions. However, GPS-based localization may have some inherent defects [16], how to deal with the errors GPS brings in will be in our future works.

References

[1] M. J. Piran, G. R. Murthy, and G. P. Babu, “Vehicular ad hoc and sensor networks: principles and challenges,” International Journal of Ad hoc, Sensor and Ubiquitous Computing. In press.

[2] B. Karp and H. T. Kung, “GPSR: Greedy Perimeter Stateless Routing for wireless networks,” in Proceedings of the 6th Annual International Conference on Mobile Computing and Networking (MOBICOM ’00), pp. 243–254, August 2000.

[3] J. Zhao and G. Cao, “VADD: Vehicle-assisted data delivery in vehicular ad hoc networks,” in Proceedings of the 25th IEEE International Conference on Computer Communications (INFOCOM ’06), pp. 1–12, April 2006.

[4] C. E. Perkins and E. M. Royer, “Ad-hoc on-demand distance vector routing,” in Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications (WMCSA ’99), pp. 90–100, February 1999.

[5] G. Pei, M. Gerla, and T. W. Chen, “Fishyeye State Routing: a routing scheme for ad hoc wireless networks,” in Proceedings of the IEEE International Conference on Communications (ICC ’00), pp. 70–74, June 2000.

[6] D. B. Johnson and D. A. Maltz, “Dynamic source routing in ad hoc wireless networks,” in Mobile Computing, T. Imielinski and H. F. Korth, Eds., The Kluwer International Series in Engineering and Computer Science, Springer, 1996.

[7] C. Lochert, M. Mauve, F. Holger, and H. Hartenstein, “Geographic routing in city scenarios,” ACM SIGMOBILE Mobile Computing and Communications Review, vol. 9, no. 1, pp. 69–72, 2005.

[8] V. Naumov and T. R. Gross, “Connectivity-aware routing (CAR) in vehicular ad hoc networks,” in Proceedings of the 26th IEEE International Conference on Computer Communications (INFOCOM ’07), pp. 1919–1927, May 2007.

[9] E. Murphy-Chutorian, A. Doshi, and M. M. Trivedi, “Head pose estimation for driver assistance systems: A robust algorithm and experimental evaluation,” in Proceedings of the 10th International IEEE Conference on Intelligent Transportation Systems (ITSC ’07), pp. 709–714, October 2007.

[10] G. A. T. Holt, M. J. Reinders, and E. A. Hendriks, “Multi-dimensional dynamic time warping for gesture recognition,” in Proceedings of the 13th Annual Conference of the Advanced School for Computing and Imaging, 2007.

[11] R. Muscillo, S. Conforto, M. Schmid, P. Caselli, and T. D’Alessio, “Classification of motor activities through derivative dynamic time warping applied on accelerometer data,” Proceedings of the 29th Annual International Conference of the IEEE, vol. 2007, pp. 4930–4933, 2007.

[12] D. A. Johnson and M. M. Trivedi, “Driving style recognition using a smartphone as a sensor platform,” in Proceedings of the 14th International IEEE Conference on Intelligent Transportation Systems, Washington, DC, USA, 2011.

[13] S. Y. Cheng and M. M. Trivedi, “Turn-intent analysis using body pose for intelligent driver assistance,” IEEE Pervasive Computing, vol. 5, no. 4, pp. 28–37, 2006.
[14] J. Dai, J. Teng, X. Bai, Z. Shen, and D. Xuan, “Mobile phone based drunk driving detection,” in Proceedings of the 4th International Conference on Pervasive Computing Technologies for Healthcare (Pervasive Health), March 2010.

[15] U. Lee, B. Zhou, M. Gerla, E. Magistretti, P. Bellavista, and A. Corradi, “Mobeyes: Smart mobs for urban monitoring with a vehicular sensor network,” IEEE Wireless Communications, vol. 13, no. 5, pp. 52–57, 2006.

[16] M. J. Piran and G. R. Murthy, “A novel routing algorithm for vehicular sensor networks,” International Journal of Wireless Sensor Networks, vol. 2, no. 2, pp. 919–923, 2010.
