MobiT: A Distributed and Congestion-Resilient Trajectory Based Routing Algorithm for Vehicular Delay Tolerant Networks

Li Yan
University of Virginia
ly4ss@virginia.edu

Haiying Shen
University of Virginia
hs6ms@virginia.edu

Kang Chen
Southern Illinois University
kchen@siu.edu

ABSTRACT

Packet routing is important for Vehicular Delay Tolerant Networks (VDTNs). Opportunistic routing algorithms based on historical records are insufficiently accurate in forwarder selection due to movement randomness of vehicles. Trajectory-based routing algorithms tackle vehicle movement randomness but cannot be directly used in VDTNs due to the dependence on APs. In this paper, we develop a distributed trajectory-based routing algorithm (called MobiT) for VDTNs. This non-trivial task faces three challenges. First, vehicle trajectories must be sufficiently collected. Second, the trajectories cannot be updated frequently due to limited resources of the repository nodes. Third, achieving high routing performance even with partially collected trajectories. Our real trace study lays the foundation of the design of MobiT. Taking advantage of different roles of vehicles, MobiT uses service vehicles that move in wide areas to collect vehicle trajectories, and rely on the service vehicles and roadside units (called schedulers) for routing scheduling. By using regular temporal congestion state of road segments, MobiT schedules the packet to arrive at a roadside unit prior to the destination vehicle to improve routing performance. Further, MobiT leverages vehicles' long-term mobility patterns to assist routing. Extensive trace-driven and real experiments show the effectiveness and efficiency of MobiT.

CCS CONCEPTS

•Networks →Network control algorithms; Cyber-physical networks;

ACM Reference format:
Li Yan, Haiying Shen, and Kang Chen. 2016. MobiT: A Distributed and Congestion-Resilient Trajectory Based Routing Algorithm for Vehicular Delay Tolerant Networks. In *Proceedings of The 2nd ACM/IEEE International Conference on Internet-of-Things Design and Implementation, Pittsburgh, PA, USA, April 18-21, 2017 (IoTDI '17),* 6 pages.
DOI: http://dx.doi.org/10.1145/3054977.3054996

IoTDI '17, Pittsburgh, PA, USA

DOI: http://dx.doi.org/10.1145/3054977.3054996

1 INTRODUCTION

In recent few years, many research efforts have been devoted to Vehicular Delay Tolerant Networks (VDTNs) [1, 8]. VDTNs can alleviate bandwidth burden on networks and serve areas with sparse infrastructures. In such vehicular networks with sparse connection, packet delivery between vehicles is important for many purposes. For example, a vehicle needs to report a traffic accident to a police vehicle far from the crash site, or a vehicle using a mobile social network wants to share a newsfeed with its friend vehicle miles away. However, due to the high mobility of vehicles and disconnected nature of VDTNs, efficient packet delivery is non-trivial.

Previous opportunistic routing algorithms [7, 8, 21, 22] define different utilities (e.g., meeting probabilities) and forward the packet to vehicles or Roadside Units (RSUs) that have larger utilities with the destination vehicle. However, these algorithms use vehicles' historical meeting records to schedule packet forwarding, which has been proven insufficiently accurate [27] due to movement randomness of some vehicles.

Determining packet forwarder based on vehicles' trajectories is effective in handling movement randomness [10–12, 24]. In the trajectory-based routing algorithms, vehicles repeatedly report trajectories to Access Points (APs) sparsely located along roads. A central server then uses these shared trajectories to schedule forwarders to carry the packet to the destination vehicle in its driving route. However, these algorithms cannot be directly used in VDTNs due to the dependence on APs.

In this paper, we design MobiT, which derives vehicle Mobility from Trajectories for routing. First, MobiT uses service vehicles to collect vehicle trajectories, and relies on the service vehicles and roadside units (RSU) (both are called schedulers) for determining routing path. Second, MobiT only requires each participating vehicle to report its trajectory to a scheduler when it starts moving (called initial trajectory) rather than repeated reporting. To tackle outdated trajectory, MobiT considers the temporal change of road congestion state when using the initial trajectories for vehicle movement prediction. Third, MobiT exploits both short-term mobility (i.e., trajectory) and long-term mobility (i.e., road/area visiting pattern) as complementary approaches. When determining routing path, MobiT schedules the packet to arrive at an RSU prior to the destination vehicle, which generates higher performance than scheduling direct meeting as in the previous trajectory-based algorithms. When a routing path cannot be found, MobiT finds a path to let

the packet approach the destination vehicle. If the trajectory of the destination vehicle is unavailable, MobiT uses the vehicle's long-term mobility to forward the packet.

MobiT can also overcome some problems in the previous centralized trajectory-based methods. First, due to fluctuating road traffic, it is very difficult to schedule an exact meeting with the destination vehicle regardless of the powerful capacity of the central server. Second, vehicles sometimes rely vehicular communication to maintain contact with the APs. The trajectories in the central server may be outdated due to vehicles' intermittent connection to the APs and possible packet loss in communication. Third, the selection of forwarders does not consider the change of road traffic at different times and road segments. To our best knowledge, this work is the first that realizes efficient distributed trajectory-based routing algorithm in VDTNs. The remainder of the paper is organized as follows. Section 2 presents literature overview. Section 3 presents the design of MobiT. Section 4 presents the performance evaluation. Section 5 concludes this paper with future directions.

2 RELATED WORK

Opportunistic routing algorithms. These algorithms extract utilities from vehicles' historical records. The packet is forwarded at the direction maximizing the utility. SADV [7] lets packets wait at intersections until the path with minimum delay is available. Ishihara *et al.* [8] schedules packet delivery by the packet's aggregated demand and the historical condition of neighboring vehicles having the same packet. EBT [21] utilizes users' previous encounters to construct a relation graph for packet forwarding. Tie *et al.* [22] proposed the Robust Replication Routing (R3), which unifies mesh, MANET, DTN routing paradigms by predicting the distribution of link delays. Kong *et al.* [14] proposed a frequency-divided instantaneous neighbors estimation system for vehicular networks. In [19], Schwartz *et al.* focuses on guidelines for the design of data dissemination in vehicular networks. These works rely on historical information to predict future encounter, but cannot guarantee high efficiency.

Trajectory-based routing algorithms. Several recent works utilizing vehicles' trajectories in routing have been proposed. Wu *et al.* [23] found and used the spatio-temporal correlation of vehicle mobility in data delivery. TBD [11], TSF [12] and STDFS [24] use APs to collect vehicle trajectories. Then, the rendezvous position between the destination vehicle and the packet is determined based on accumulated trajectories, and the packet is forwarded to the rendezvous position. Due to the sparsity of APs, ad-hoc network is needed to bridge APs and vehicles. However, these algorithms have drawbacks of susceptibleness to road congestion and possibly outdated trajectories due to inaccessibility to APs, which affect the accuracy of the forwarder selection.

3 SYSTEM DESIGN

We consider a VDTN with n vehicles denoted by $N_i (i = 1, 2, \ldots, n)$ and make following assumptions.

(1) Each vehicle is equipped with a Dedicated Short Range Communication (DSRC) device [5]. When two vehicles are within each other's communication range, an encounter happens.
(2) Each vehicle is equipped with a navigation system, which generates trajectory consisting of future positions and estimated arrival times, and road maps [15, 25].
(3) The area in the VDTN is partitioned into multiple sub-districts with equal number of landmarks. Following [26], we assign a center landmark for each sub-district.
(4) Each intersection is installed with an RSU which uses DSRC for communication [6, 18]. Service vehicle can exchange information with RSUs, while the others can only drop a packet to an RSU.

There are existing works focusing on motivating users to share mobility information [4] and ensuring users' privacy [16]. We leave the work for MobiT as our future work.

3.1 Representation of Vehicle Mobility

3.1.1 Short-term Mobility and Congestion-considered Update. At the beginning of a trip, each vehicle generates its initial trajectory. An initial trajectory of vehicle N_i is $\langle N_i; \{p_i^0, p_i^1, \ldots, p_i^Q\}; T_s \rangle$, where $\{p_i^0, p_i^1, \ldots, p_i^Q\}$ represents the sequence of Q positions on the trajectory. T_s is the starting time of this trajectory. Each vehicle maintains its own short-term mobility information. While service vehicle collects short-term mobility information from every vehicle it meets. If a service vehicle meets another service vehicle or an RSU, they exchange their known mobility information.

Congestion state table: MobiT use road segment, which is the interval between two neighbor intersections, as the basic unit of roads. It has been shown that the travel time on a road can be estimated based on the congestion state of composing road segments [13]. It is also noticeable that urban traffic pattern repeats in daily fashion [9]. Thus, we use the congestion states of roads under different times to assist determining vehicle arrival times. We firstly design the table of delays, which records the travel times of a road's composing segments under congested and non-congested cases based on historical statistics. Then, for each road, we design a table of binary vectors to describe its proad congestion.

For each road segment, it has distinct travel times corresponding to congested and non-congested situations [13]. For example, *College Ave* has 6 segments as shown in Table 1. If segment 1 is congested, it takes the vehicle 2min to drive through. Otherwise, 50s is needed for driving through. Suppose the segments 1, 2 and 4 are congested, and the other segments are non-congested, the travel time needed to drive through *College Ave* is 50s+ 5min+6min+20s+2min+10s=14min20s.

Table 1: Table of *College Ave*'s delays.

Segment ID	0	1	2	3	4	5
Congested (1)	2min	5min	6min	1min	2min	30s
Otherwise (0)	50s	2min	1min	20s	1min	10s

Then, for each sequence of road segments, we can use a binary vector to depict its congestion states. For example, if the segments 1, 2 and 4 of *College Ave* are congested, current congestion state of the road is $[0, 1, 1, 0, 1, 0]$. To collect all

possible congestion states of a road during different times, the congestion state of the road is sampled by a time unit, say per hour. Thus, for each hour, we have several sampling results representing all the possible congestion states of the road at this time. Finally, we classify these congestion states along with their respective probabilities in ascending order of time, and get the table of congestion states of the road as shown in Table 2. In Table 2, *College Ave* has several congestion states under each time interval. Each congestion state has a probability, which measures its appearance frequency among all possible congestion states. For example, during the interval from 00:00 to 01:00, *College Ave* has the probability of 0.6 to be in congestion state $[0, 1, 1, 0, 1, 0]$ and the probability of 0.4 to be in congestion state $[1, 0, 0, 0, 1, 0]$. Therefore, the estimated travel time of *College Ave* during this interval is 0.6×14 min20s+$0.4 \times$ 7min30s=11min36s.

Table 2: Table of *College Ave*'s congestion states.

Time interval	*Congestion states*
00:00∼01:00	[0, 1, 1, 0, 1, 0], 0.6; [1, 0, 0, 0, 1, 0], 0.4
01:00∼02:00	[1, 1, 1, 1, 1, 0], 0.7; [0, 1, 1, 0, 0, 0], 0.3
...	...

Since road traffic follows certain long-term pattern even under accident and weather change, using historical data to describe the congestion states is reasonable [13, 20]. The table of congestion states and delays for all roads are computed offline. All schedulers are preloaded with the two tables.

Estimation of travel time and deviation: To estimate the travel time of a trajectory, a scheduler decomposes it into roads. For each road, the scheduler refers to Table 2 for the road's current congestion state. Then, the scheduler refers to Table 1 for corresponding delays of the covered segments. By summing the delays from the start of each trajectory, the scheduler estimates the vehicle's future travel time. For example, suppose a vehicle will drive through *College Ave* between 00:00 and 01:00. According to Tables 1 and 2, the travel time of *College Ave* has probability of 0.6 to be 14min20s and probability of 0.4 to be 7min30s. Therefore, the road's estimated travel time is $\mu = 0.6 \times$ 14min20s+$0.4 \times$7min30s=11min36s, with standard deviation $\sigma = \sqrt{0.6 \times (14min20s - 11min36s)^2 + 0.4 \times (11min36s - 7min30s)^2} = 3min20s$.

3.1.2 Long-term Mobility. In this section, we introduce long-term mobility, namely routine and vehicular friendship.

Routine: Vehicles' long-term mobility has regularity [27], which is reflected as certain roads that are frequently driven by the vehicle at specific times. For example, people usually take the same routine routes to commute between home and work place. Moreover, vehicles tend to repeat their routine routes on a daily basis. For example, people regularly drive from home to work place at around 8:10 every morning. Therefore, we depict the routines of a vehicle, say N_i, as shown in Table 3.

Table 3: Table of a vehicle's routines.

Prob	*Route*	T_s	T_e
0.6	$\{p_{i1}^0, \ldots, p_{i1}^m\}$	*08:10 ∼ 08:20*	*08:30 ∼ 08:45*
0.2	$\{p_{i2}^0, \ldots, p_{i2}^n\}$	*13:00 ∼ 13:20*	*13:30 ∼ 13:45*

In Table 3, each row represents a routine of N_i. *Route* represents the series of positions a routine covers. T_s is the start time range of the routine. T_e is the end time range of the routine. T_s and T_e are determined from N_i's historical records. The probability indicates the likelihood that the vehicle will follow the routine and is calculated as $P_{R_t}(N_i) = m_t/M$, where m_t is the number of occurrences that N_i followed routine R_t as a trajectory; M is the total number of trajectories that N_i once drove during a time period, say 30 days. The routine table stores the routines with the sum of probabilities larger than and closest to 80%. This threshold can be adjusted based on system constraints.

Friendship: People living in the same area are likely to follow similar routines. For example, suppose Alice and Bob live in the same suburban community, every morning they drive the same highway to downtown. Given a packet targeting at Alice, the mobility information of Bob may be helpful. Based on such observation, MobiT measures the relationship between vehicles in terms of their similarity on routine.

Overlapping of routines: We define two vehicles are friends if the ratios of their similar routines in their respective overall routines are higher than a threshold, say α_f. For example, suppose vehicle N_i has routines: R_1, R_2 and R_3, vehicle N_j has routines: R_2 and R_3, and α_f is 0.5. Since R_2, R_3 are the similar routines of N_i and N_j, and they take up 66.7% and 100% of the total routines of N_i and N_j, respectively, which are higher than 0.5, these two vehicles are friends.

Since the similar routines of two vehicles will not be completely identical, we use spatiotemporal overlap to measure their similarity. Given two routines, say R_1 of N_i and R_2 of N_j. R_1 covers positions: $r_1 = \{p_{i1}(0), \ldots, p_{i1}(m)\}$, start time range T_{s1} and end time range T_{e1}, while R_2 covers positions: $r_2 = \{p_{j2}(0), \ldots, p_{j2}(m^{'})\}$, start time range T_{s2} and end time range T_{e2}. We use $\bar{T}_s$ and $\bar{T}_e$ to denote the mean of T_s and T_e, respectively. Then, R_1 and R_2 are similar if:

$$\begin{aligned} |\bar{T}_{e1} - \bar{T}_{e2}| &< \tau_t \\ |\bar{T}_{s1} - \bar{T}_{s2}| &< \tau_t \qquad (1) \\ \frac{|r_1 \bigcap r_2|}{|r_1 \bigcup r_2|} &> \gamma_s \qquad (2) \end{aligned}$$

where τ_t is the threshold bounding the temporal deviation of start times and end times. γ_s is the threshold bounding the spatial deviation of the positions. The thresholds are determined based on the traffic flow of specific scenes. For metropolitan cities, relatively high deviation should be tolerated. Therefore, we set $\alpha_f = 0.5$, $\tau_t = 15$min and $\gamma_s = 0.6$.

In MobiT, routine extraction and friendship determination are conducted by service vehicles and RSUs since they have the bulk of vehicles' short-term mobility information. A representative friend list is as shown in Table 4.

Table 4: Table of friends.

Vehicle ID	*Friends*
N_1	$N_0(0.5), N_2(0.3), N_3(0.2), N_4(0.1)$
N_4	$N_0(0.4), N_1(0.2)$
...	...

3.2 Routing Process

MobiT aims to deliver the packet to the encounter position prior to the destination vehicle, the packet then waits at a

nearby RSU for the destination vehicle. In MobiT, service vehicles and RSUs schedule the forwarding of packets since they have collected vehicles' mobility information. The scheduling of routing can be summarized to three cases. 1) When the destination vehicle's short-term mobility information is available, the packet will be forwarded to the destination vehicle's future position (or nearby position) along a trajectory-based routing path that leads to the shortest delay; 2) When the scheduler only has the destination vehicle's long-term mobility information, the packet will be forwarded to the destination vehicle's or its friend's routine; 3) When no mobility information of the destination vehicle is available, the packet will be forwarded to service vehicles, aiming to increase its probability of finding more useful information.

3.2.1 Short-term Mobility Based Routing. For trajectory-based routing path determination, MobiT extends its predecessor, STDFS [24]. In STDFS, based on travel time predictions, the central server constructs an encounter graph and finds the chain of trajectories connecting current position of the source vehicle and the destination vehicle with acceptable delay and delivery probability. MobiT further considers road traffic of each road segment in different times for calculating vehicle travel time. Moreover, MobiT aims to deliver the packet to the encounter position prior to the destination vehicle.

3.2.2 Long-term Mobility Based Routing. It is possible that a scheduler cannot find the proper short-term mobility information. In this case, the long-term mobility information (i.e., the routine table (Table 3) and friend table (Table 4)) will be used to guide the packet to the activity area of the destination vehicle. Specifically, from the routine table, according to current time, the scheduler firstly determines which routine the destination vehicle is likely to use. The routine is represented by the positions covered by the routine ($\{p_{i1}(0), \ldots, p_{i1}(m)\}$) with mean end time $\bar{T}_e$. Then the scheduler also uses encounter graph to find chains of trajectories that connect current position of the source vehicle with the destination vehicle's routine.

The process of selecting routing chain is the same as that with short-term mobility. Since the routines can only be auxiliary, we filter out the invalid chains. The remaining time of the destination vehicle's routine from current time T_c is $(\bar{T}_e - \bar{T}_c)$ from Table 3. Then, to ensure the packet can be forwarded to the destination vehicle before it arrives at the ending point of the routine, the scheduler filters the chains with $D_i \leq \bar{T}_e - T_c$, where D_i is the estimated delivery delay. Finally, the chain with the shortest travel time is selected.

If the destination vehicle's routine is also unavailable, the scheduler refers to the table of friends. For example, given destination vehicle N_1, Table 4 shows its friends are N_0, N_2, N_3 and N_4. Among the friends with available mobility information, the scheduler chooses the friend vehicle that has the highest ratio of similarity with N_1. Then the friend's mobility information will be used as previously described.

3.2.3 Routing without Mobility Information. It is likely the scheduler doesn't have any useful mobility information. If the scheduler is a service vehicle, it will keep the packet. If the scheduler is a RSU, it will first keep the packet and then transfer it to the service vehicle passing by.

Because each scheduler stores partial mobility information of vehicles in the system, a routing path generated by a scheduler may not be the best routing path. Therefore, whenever a node carrying a packet encounters a scheduler, it requests the scheduler to update the routing path if a chain with shorter delivery delay is found. In the routing process, if the packet misses the next forwarder, it requests nearby scheduler to launch a new round of routing.

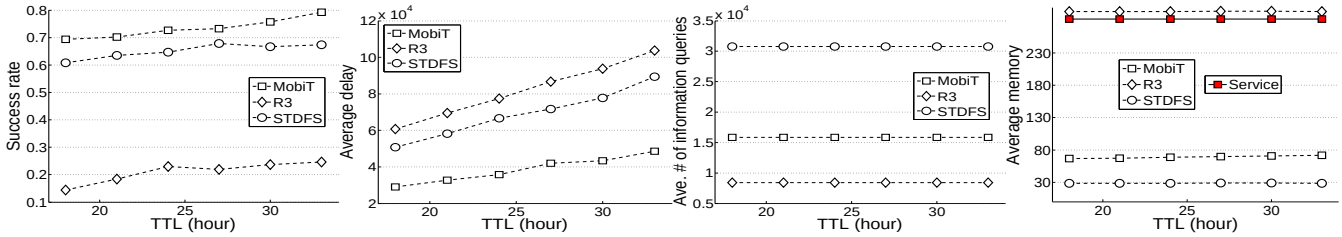
4 PERFORMANCE EVALUATION

We used the Rome [2] and the San Francisco [17] traces, which last for 30 days, for evaluation. The Rome trace has has 315 taxis and 4638 landmarks, and the San Francisco trace has 536 taxis and 2508 landmarks. We develop a trace-based simulation environment which is driven by each vehicle's movement event [3, 4].

The collection of congestion table and delay table was finished offline. For Rome and San Francisco, the threshold speeds to determine congestion are 20MPH and 30MPH, respectively [2, 17]. The congestion state of each road segment was sampled per hour. For both traces, we set the initial period to 7 days, during which service vehicles collected and disseminated mobility information. Meanwhile, service vehicles and RSUs extracted vehicles' routines as in Section 3.1.2, and determined friendship between vehicles by Equation (1) and (2) with $\alpha_f = 0.5$, $\tau_t = 15$min and $\gamma_s = 0.6$. Request rate is the number of packets generated every 24 hours in both traces and was set to 40 by default. Packet TTL, which is the valid time of a packet, was set to 24 hours. The TTL for short-term mobility information depends on trip duration.

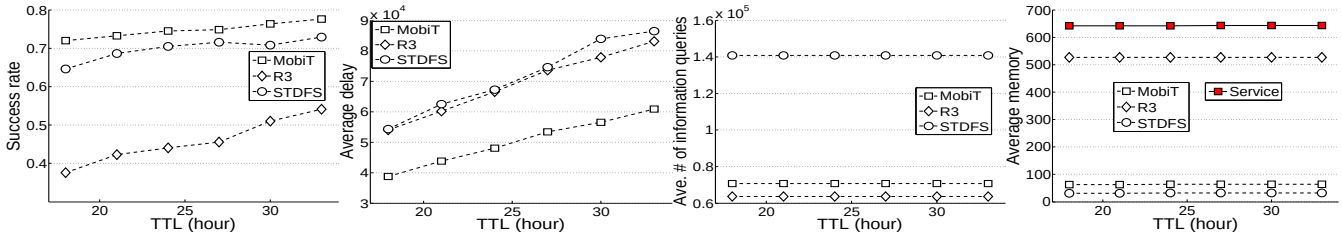
We compared MobiT with two representative algorithms: the Shared-Trajectory-based Data Forwarding method (*STDFS* in short) [24], and the Robust Replication Routing (denoted by *R3*) [22]. *STDFS* depends on vehicles' trajectories reported through APs to schedule future meeting position between forwarder and destination vehicle. In *R3*, vehicles record their historical contact with others. The packet carrier utilizes the historical delays of the vehicles to the destination vehicle to guide packet routing. In simulations, we equipped 2782 and 1504 landmarks with RSUs/APs in Rome and San Francisco, respectively, which is as specified in *STDFS* [24]. We measured following metrics:

- *Success rate*: The percentage of packets that successfully reach their destination vehicles.
- *Average delay*: The average time (in seconds) used by packets to reach their destination vehicles.
- *Average number of information queries*: The average number of information queries transmitted among nodes.
- *Average vehicle memory usage*: The average number of memory units used by each vehicle. Since the basic data of MobiT (i.e., a congestion state vector, delays) usually



(a) Success rate. (b) Average delay. (c) Average number of info. queries. (d) Average memory usage.

Figure 1: Performance with different TTLs using the Rome trace.



(a) Success rate. (b) Average delay. (c) Average number of info. queries. (d) Average memory usage.

Figure 2: Performance with different TTLs using the San Francisco trace.

include several integers (4 bytes) or doubles (8 bytes), we set each memory unit takes 50 bytes. Each piece of mobility information (i.e., trajectory, routine) includes sequences of integers, doubles and strings, so it takes around $4 \sim 8$ memory units. Each entry of table (i.e., congestion state, delay, friend) takes 1 memory unit.

4.1 Experimental Results

In the experiment, we varied the packet TTL from 18 hours to 33 hours with 3 hours as the step size.

4.1.1 Success Rate. Figure 1(a) and Figure 2(a) show the success rates of the algorithms under different packet TTLs in both traces. In these figures, the success rates follow: *MobiT*>*STDFS*>*R3*. We can see that *MobiT* always has the highest packet delivery success rate than the other two algorithms under various situations.

The success rates of all algorithms remain nearly constant under different request rates, but increase with the ascending of packet TTL. *R3* always has the lowest success rate. This is because vehicles have independent and random movement, vehicles' historical meeting records with the destination vehicle does not guarantee another meeting with it in future. Thus the selection of forwarder may be mistaken.

In contrast, *STDFS* has much higher success rate. It is because the packet is always forwarded to a future position of the destination vehicle with certain accuracy. If vehicles' movement is not influenced by road congestion, or its trajectory in central server is continuously updated without disconnection, the forwarder is highly likely to meet the destination vehicle.

MobiT always achieves the highest success rate. This is because *MobiT* considers road congestion state in estimating vehicles' arrival times on trajectory, which makes the estimation more tolerant to traffic change. Also, *MobiT* is not afraid that trajectory may be outdated because it uses road delay table corresponding to road congestion to dynamically estimate the arrival times of vehicles when scheduling routing. Moreover, *MobiT* aims to let packet arrive at the meeting position prior to the destination vehicle through considering various kinds of mobility information, which also increases the success rate.

4.1.2 Average Delay. Figure 1(b) and Figure 2(b) show the metric under different packet TTLs in both traces. In Rome, the average delays follow: *MobiT*<*STDFS*<*R3*. While in San Francisco, the average delays follow: *MobiT*<*R3*<*STDFS*. We can see that *MobiT* achieves the best performance.

R3 does not know the position of the destination vehicle, so it is likely to select the vehicle that will not meet the destination vehicle as forwarder. Therefore, it has the highest delay. Although *STDFS* knows the future position of the destination vehicle from its trajectory, it can only forward packet when a complete chain of trajectories connecting the source vehicle and the destination vehicle is available. Also the destination vehicle's disconnection to APs will make its trajectory outdated, thereby hindering the efficient delivery of the packet. So *STDFS* ranks the second. *MobiT* utilizes various kinds of vehicles' mobility information to help the packet keep approaching the actual activity area of its destination vehicle, so it has the shortest delay.

4.1.3 Average Number of Information Queries. Figure 1(c) and Figure 2(c) show the metric of the algorithms under different packet TTLs in both traces. The average number of information query follow: *R3*<*MobiT*<*STDFS*. We can see that *MobiT* achieves less information query overhead than *STDFS* but more information query overhead than *R3*.

This is because *STDFS* requires vehicles to repeatedly report their trajectories to the APs, so it has the highest number of information query. In contrast, *MobiT* only needs vehicles to report their initial trajectory to schedulers. Therefore, it ranks the second. In *R3*, information query only happens in the encounter of nodes with suitable delay predictions. Therefore, it ranks the lowest.

4.1.4 Average Memory Usage. Figure 1(d) and Figure 2(d) show the metric of the algorithms under different packet TTLs in both traces. For *MobiT*, we additionally measured the metric for service vehicle and RSU, which is represented with "Service". The metric follows: *R3*>*Service*>*MobiT*>*STDFS* in Rome, and *Service*>*R3*>*MobiT*>*STDFS* in San Francisco. We can see that the memory usage of general vehicle is comparable to the one of *STDFS*.

Since *R3* has duplicated packets, and each vehicle needs to maintain the distribution of path's historical delays, vehicles in *R3* have the highest memory usage. In *MobiT*, general vehicles only need to maintain short-term mobility information and occasional packets. Therefore, *MobiT* uses much lower memory than *R3*. On the other hand, service vehicles and RSUs need to maintain much mobility information and awaiting packets. Therefore, their memory usage is comparable to that of *R3*. In *STDFS*, vehicles only need to record their trajectory. In contrast to *MobiT*, in which a vehicle may need to help forwarding packet even if they are not determined to approach the destination vehicle, *STDFS* only requires vehicles that can form a complete chain of trajectories between the source vehicle and the destination vehicle. Therefore, *STDFS* uses the least memory.

5 CONCLUSION

Message delivery is an important function in VDTNs for Intelligent Transportation Systems. Previous opportunistic routing algorithms for VDTNs cannot achieve high success rate and low delay due to insufficiently accurate estimation of vehicles' future encounter. Previous trajectory-based routing algorithms can overcome this drawback but require APs hence cannot be directly used for decentralized VDTNs. We propose MobiT, a distributed trajectory-based routing algorithm for VDTNs. MobiT aims to let the packet arrive at a RSU prior to the destination vehicle. By taking advantage of travelling features of different vehicles, MobiT uses public service vehicles and RSUs to collect vehicle mobility information in a distributed manner and schedule trajectory-based routing paths to destination vehicles. To avoid frequent communication for trajectory updates, trajectories only need to be reported once and then are updated based on stored road segment congestion state at different times. Extensive trace-driven and real-world experiments show MobiT's higher efficiency and effectiveness compared with previous routing algorithms. In the future, we plan to further exploit vehicles' relationship in routing.

ACKNOWLEDGEMENTS

This research was supported in part by U.S. NSF grants ACI-1719397 and CNS-1733596, and Microsoft Research Faculty Fellowship 8300751.

REFERENCES

[1] Tamer Abdelkader, Kshirasagar Naik, Amiya Nayak, Nishith Goel, and Vineet Srivastava. 2013. SGBR: A routing protocol for delay tolerant networks using social grouping. *TPDS* 24, 12 (2013).

[2] R. Amici, M. Bonola, L. Bracciale, P. Loreti, A. Rabuffi, and G. Bianchi. 2014. Performance assessment of an epidemic protocol in VANET using real traces. In *Proc. of MoWNeT*.

[3] Kang Chen and Haiying Shen. 2012. SMART: Lightweight distributed Social Map based Routing in Delay Tolerant Networks.. In *Proc. of ICNP*.

[4] Kang Chen, Haiying Shen, and Li Yan. 2015. Multicent: A Multifunctional Incentive Scheme Adaptive to Diverse Performance Objectives for DTN Routing. *IEEE TPDS* 26, 6 (2015).

[5] Christian Cseh. 1998. Architecture of the dedicated short-range communications (DSRC) protocol. In *Proc. of VTC*.

[6] Kakan C Dey, Li Yan, Xujie Wang, Yue Wang, Haiying Shen, Mashrur Chowdhury, Lei Yu, Chenxi Qiu, and Vivekgautham Soundararaj. 2016. A Review of Communication, Driver Characteristics, and Controls Aspects of Cooperative Adaptive Cruise Control (CACC). *IEEE TITS* 17, 2 (2016).

[7] Yong Ding and Li Xiao. 2010. SADV: static-node-assisted adaptive data dissemination in vehicular networks. *TVT* 59, 5 (2010).

[8] Susumu Ishihara, Nobuhiro Nakamura, and Yuya Niimi. 2013. Demand-based location dependent data dissemination in VANETs. In *Proc. of MobiCom*.

[9] Vipin Jain, Ashlesh Sharma, and Lakshminarayanan Subramanian. 2012. Road traffic congestion in the developing world. In *Proc. of DEV*.

[10] Jaehoon Jeong, Shuo Guo, Yu Gu, Tian He, and David HC Du. 2010. TSF: Trajectory-based statistical forwarding for infrastructure-to-vehicle data delivery in vehicular networks. In *Proc. of ICDCS*.

[11] Jaehoon Jeong, Shuo Guo, Yu Gu, Tian He, and David HC Du. 2011. Trajectory-based data forwarding for light-traffic vehicular ad hoc networks. *TPDS* 22, 5 (2011).

[12] Jaehoon Jeong, Shuo Guo, Yu Gu, Tian He, and David HC Du. 2012. Trajectory-based statistical forwarding for multihop infrastructure-to-vehicle data delivery. *TMC* 11, 10 (2012).

[13] Jeffrey P Kharoufeh and Natarajan Gautam. 2004. Deriving link travel-time distributions via stochastic speed processes. *Transportation Science* 38, 1 (2004).

[14] Linghe Kong, Xi Chen, Xue Liu, and Lei Rao. 2015. FINE: Frequency-divided instantaneous neighbors estimation system in vehicular networks. In *Proc. of PerCom*.

[15] Zhuozhao Li and Haiying Shen. 2015. Designing a Hybrid Scale-Up/Out Hadoop Architecture Based on Performance Measurements for High Application Performance. In *Proc. of ICPP*.

[16] Sergio Marti, Thomas J Giuli, Kevin Lai, and Mary Baker. 2000. Mitigating routing misbehavior in mobile ad hoc networks. In *Proc. of MobiCom*.

[17] Michał Piórkowski, Natasa Sarafijanovic-Djukic, and Matthias Grossglauser. 2009. A parsimonious model of mobile partitioned networks with clustering. In *Proc. of COMSNETS*.

[18] Ankur Sarker, Chenxi Qiu, and Haiying Shen. 2016. A Decentralized Network with Fast and Lightweight Autonomous Channel Selection in Vehicle Platoons for Collision Avoidance. In *Proc. of MASS*.

[19] Ramon S Schwartz, Hylke W Van Dijk, and Hans Scholten. 2011. Towards opportunistic sensed data dissemination in vehicular environments. In *Proc. of PerCom*.

[20] Shiliang Sun, Changshui Zhang, and Guoqiang Yu. 2006. A Bayesian network approach to traffic flow forecasting. *TITS* 7, 1 (2006).

[21] Andrew Symington and Niki Trigoni. 2012. Encounter based sensor tracking. In *Proc. of MobiHoc*.

[22] Xiaozheng Tie, Arun Venkataramani, and Aruna Balasubramanian. 2011. R3: robust replication routing in wireless networks with diverse connectivity characteristics. In *Proc. of MobiCom*.

[23] Yuchen Wu, Yanmin Zhu, and Bo Li. 2011. Trajectory improves data delivery in vehicular networks. In *Proc. of INFOCOM*.

[24] Fulong Xu, Shuo Guo, Jaehoon Jeong, Yu Gu, Qing Cao, Ming Liu, and Tian He. 2011. Utilizing shared vehicle trajectories for data forwarding in vehicular networks. In *Proc. of INFOCOM*.

[25] Li Yan, Haiying Shen, Juanjuan Zhao, Chengzhong Xu, Feng Luo, and Chenxi Qiu. 2017. CatCharger: Deploying Wireless Charging Lanes in a Metropolitan Road Network through Categorization and Clustering of Vehicle Traffic. In *Proc. of INFOCOM*.

[26] Yu Zheng, Yanchi Liu, Jing Yuan, and Xing Xie. 2011. Urban computing with taxicabs. In *Proc. of UbiComp*.

[27] Yanmin Zhu, Yuchen Wu, and Bo Li. 2014. Trajectory Improves Data Delivery in Urban Vehicular Networks. *TPDS* 25, 4 (2014).