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

Li Yan, Haiying Shen, *Senior Member, IEEE* and Kang Chen

***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. Our trace-driven simulation and real experiments show the effectiveness and efficiency of MobiT.**

***Index Terms*—Vehicular delay tolerant networks, vehicular social networks, routing.**

I. Introduction

In recent few years, many research efforts have been devoted to Vehicular Delay Tolerant Networks (VDTNs) [1]–[5]. 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. In the problem of packet delivery in VDTNs, a *source vehicle* wants to deliver a packet to its *destination vehicle*. However, due to the high mobility of vehicles and disconnected nature of VDTNs, efficient packet delivery is non-trivial.

Previous opportunistic routing algorithms [6]–[11] 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 the vehicles' historical meeting records to schedule packet forwarding, which has been proven insufficiently accurate [12] due to movement randomness of some vehicles.

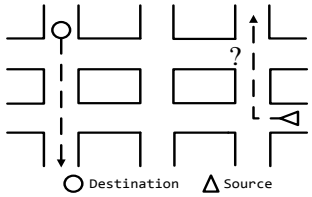


Fig. 1: Demonstration of packet delivery in VDTNs.

Determining packet forwarder based on vehicles' trajectories is effective in handling movement randomness [13]–[16]. 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 aim to develop a distributed trajectory-based routing algorithm for VDTNs. However, this task is non-trivial. First, the vehicle trajectories must be sufficiently collected in repository nodes for determination of trajectory-based routing path. Second, the trajectories cannot be updated frequently due to limited resources of the repository nodes. Third, it is hard to achieve high routing performance with partially collected vehicle trajectories due to lack of APs.

We first studied two real-world traces [17], [18] and gained several observations that help tackle the challenges. (i) Urban vehicle trips have short duration but cover several intersections. (ii) Vehicles have certain routine routes at certain times. (iii) Some vehicles share common routines at certain times. (iv) Vehicles in urban area generally move within small ranges, while service vehicles (e.g., buses, delivery trucks, and garbage trucks) move among many sub-districts.

With these observations, 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 on 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. Third, the selection of forwarders does not consider the change of road traffic at different times and road segments. We summarize our contributions as follows:

(1) We conducted real trace studies, which lays the foundation of the design for MobiT.
(2) MobiT innovatively leverages both short-term and long-term mobility information of vehicles. It relies on vehicle roles to collect mobility information, and also novelly considers changes of congestion state during routing.
(3) Our trace-driven experiments and real experiment show MobiT's superior performance and its efficiency in real environment.

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 II presents literature overview. Section III presents the trace analysis. Section IV presents the design of MobiT. Section V presents the performance evaluation of MobiT in trace-driven experiments and real implementation. Section VI concludes this paper with future directions.

II. Related Work

Due to sparse communication infrastructures, packets in VDTNs are delivered in the "store-and-forward" manner that utilizes mobility and contact information between vehicles for packet delivery. Motivated by this requirement, many methods have been proposed for packet delivery in VDTNs. Based on the mobility information utilized for packet scheduling and the way they schedule the routing of the packets, these methods can be generally categorized into following two groups.

Opportunistic routing algorithms. Data forwarding and routing in mobile opportunistic networks have gained a lot of attention recently. Generally, previous algorithms extract various kinds of utilities (e.g., meeting probability) from vehicles' historical records. The packet is forwarded at the direction maximizing the utility. SADV [6] lets packets wait at intersections until the path with minimum delay is available. Ishihara *et al.* [7] scheduled packet delivery by the packet's aggregated demand and the historical condition of neighboring vehicles having the same packet. EBT [8] utilizes users' previous encounters to construct a relation graph for packet forwarding. Tie *et al.* [9] proposed the Robust Replication Routing (R3), which unifies mesh, MANET, DTN routing paradigms by predicting the distribution of link delays. Kong *et al.* [11] proposed a frequency-divided instantaneous neighbors estimation system for vehicular networks. In [10], Schwartz *et al.* focused on guidelines for the design of data dissemination in vehicular networks.

These works all rely on historical information to predict future encounter. However, as indicated in a recent work [12], the chosen forwarder may not meet the destination vehicle in large-scale vehicular networks due to the movement randomness of some vehicles, which impairs routing accuracy and efficiency.

Trajectory-based routing algorithms. Vehicles' trajectories have recently been recognized to be helpful to handling the vehicle movement randomness. Several recent works utilizing vehicles' trajectories in routing have been proposed. Wu *et al.* [19] found and used the spatio-temporal correlation of vehicle mobility in data delivery.

TBD [14], TSF [15] and STDFS [13] used 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.

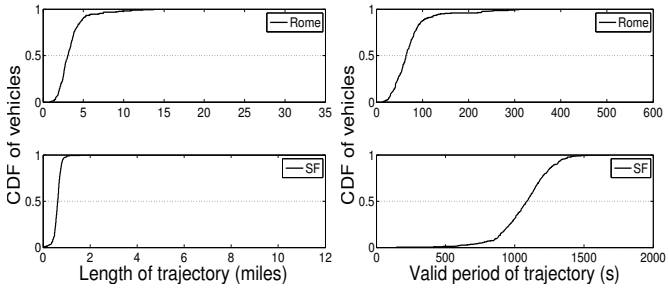
However, these algorithms cannot be directly applied for VDTNs due to their dependence on APs. In the case with inaccessibility to APs (which is quite common in VDTN context) and meanwhile the vehicle trajectory is susceptible to road congestion, the accuracy of the forwarder selection will be degraded.

III. Trace Data Analysis

In this section, we present our trace analysis on the Rome trace [18] and the San Francisco trace [17], which are two taxi traces lasting for 30 days. Each taxi reports location records (timestamp, ID, position) to a central server every 7 seconds. For each position, a precision is returned by the GPS system. We filtered out positions with a precision larger than 20 meters and taxis with few appearances (< 500), and extracted road layout from vehicle movement. The positions of vehicles are highly overlapped on popular roads, thus we extracted intersections where vehicles make turns as landmarks and simplified the records to sequences of landmarks. Finally, the Rome trace has 315 taxis and 4638 landmarks, and the San Francisco trace has 536 taxis and 2508 landmarks. When a vehicle stays at one position for more than 5min, we consider this position as an *anchor position* and it marks the ending of the previous trajectory and the beginning of the next trajectory. Thus, the anchor positions separate each vehicle's record into several trajectories. Each trajectory has a sequence of landmarks with arrival times. Additionally, we added service vehicles' records to the two traces based on the schedule of local public transit vehicles in the city of Rome [20] and San Francisco [21]. In each of the following figures, the top figure is for Rome and the bottom figure is for San Francisco.

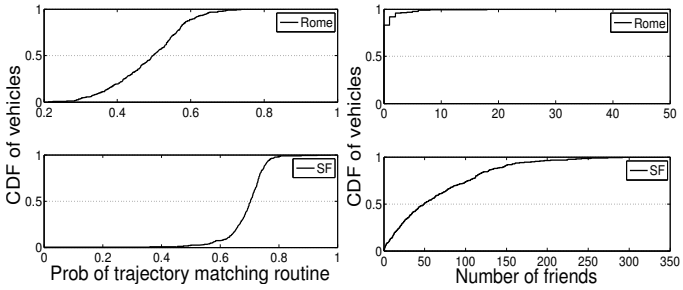
A. Properties of Vehicle Trajectory

We define trajectory length as the number of covered landmarks, and trajectory duration as the time span between



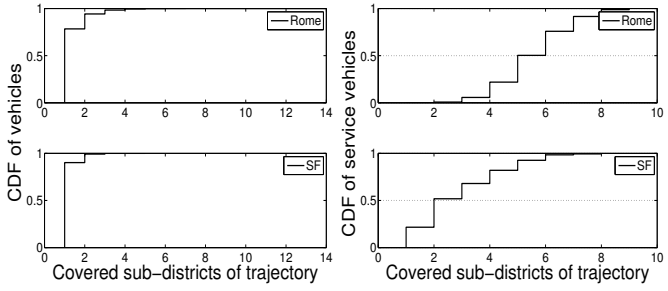
(a) Length of trajectory. (b) Time period of trajectory.

Fig. 2: Properties of vehicles' trajectories.



(a) Trajectory matches routine. (b) Number of friends.

Fig. 3: Long-term mobility of vehicles.



(a) General vehicles. (b) Service vehicles.

Fig. 4: Inter-district property of trajectories.

its start time and its end time. In previous trajectory-based routing methods, if a vehicle is congested and cannot access APs to update trajectory, its trajectory in the central server will be outdated. To confirm this, we measured the Cumulative Distribution Function (CDF) of trajectory lengths and the durations as shown in Figure 2(a) and Figure 2(b). For Rome, the trajectory length of 90% of the vehicles is less than 5miles, and the trajectory duration of 90% of the vehicles is less than 200s. For San Francisco, the trajectory length of 90% of the vehicles is less than 1mile, and the trajectory duration of 90% of the vehicles is less than 1200s. This demonstrates that urban vehicle trips usually last for a few minutes but cover several landmarks. Such short trajectory is susceptible to road congestion. For example, if a congested vehicle cannot access APs and fail to update its trajectory for only several minutes, its original trajectory information in the central server is outdated. Therefore, road congestion should be considered.

B. Long-term Mobility

If a vehicle's ratio of following a trajectory during specific time among all of its trajectories is higher than a threshold (i.e., 20%), we view the trajectory as the vehicle's *routine*. Additionally, we define two vehicles are friends if the ratios of their similar routines in their respective overall routines are higher than a threshold (i.e., 50%). Note that these two thresholds are heuristic and can be adjusted based on application requirement. To verify that routines may reveal vehicles' movement and the existence of friendship, we draw Figure 3(a) that shows the CDF of the probability that a vehicle's trajectory matching its routine and the CDF of the number of a vehicle's friends. For Rome, more than 20% and less than 80% of the vehicles have probability of trajectory matching routine between 40% and 60%, and 80% of the vehicles have more than 2 friends. For San Francisco, more than 20% and less than 80% of the vehicles have probability of trajectory matching routine between 65% and 75%, and 80% of the vehicles have more than 20 friends. This demonstrates that vehicles' trajectories usually match their routines, and most vehicles have friends sharing similar routines. Therefore, if a packet is forwarded to its destination vehicle's routine or friends, this delivery should have high probability to succeed.

C. Inter-district Property of Trajectories

We partitioned the whole areas in the two traces into several sub-districts. Specifically, we firstly evenly chose 19 landmarks in Rome and 52 landmarks in San Francisco, as center landmarks. Then, the other landmarks were clustered to the respective nearest center landmark based on the Euclidean distance. Finally, Rome has 52 sub-districts with 89 landmarks per sub-district. San Francisco has 19 sub-districts with 132 landmarks per sub-district. We use the number of sub-districts that a trajectory covers as its inter-district property. Since general urban vehicles usually travel short trips (shown in Section III-A), their trajectory may only bridge a few sub-districts. Since service vehicles have wide movement ranges, their trajectories may bridge much more sub-districts. To verify this conjecture, we measured the CDF of the number of sub-districts covered by general vehicles and service vehicles as shown in Figure 4(a) and Figure 4(b), respectively. We see that in both traces, most general vehicles' trajectories (about 98%) cover only one sub-district. While more than 75% of the service vehicles' trajectories cover more than 4 sub-districts in Rome, and more than 75% of the public service vehicles' trajectories cover more than 2 sub-districts in San Francisco. This result confirms that urban general vehicles prefer to move within a sub-district, while service vehicles move among at least 10% of sub-districts. Therefore, we can use service vehicles to collect and disseminate mobility information.

IV. 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 has a Dedicated Short Range Communication (DSRC) device [22]. 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 [23], [24]. Note that the arrival times of the trajectory can be estimated with certain accuracy considering the speed limit of the road segments. However, the estimated arrival times may be affected by various factors (e.g., traffic congestion).

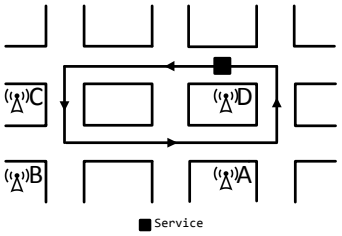


Fig. 5: Trajectory of a service vehicle.

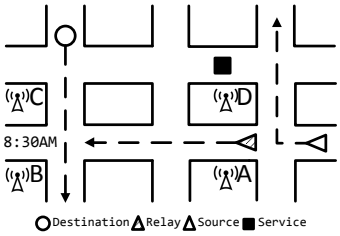


Fig. 6: Determine the relay vehicle.

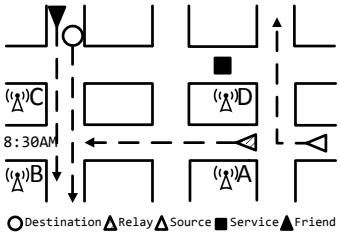


Fig. 7: Utilize the friend vehicle.

Thus, we aim to let the packet always arrive at the encounter position prior to the destination vehicle.

(3) The area in the VDTN is partitioned into multiple sub-districts with equal number of landmarks. Following [25], we assign a center landmark for each sub-district.

(4) Each intersection is installed with an RSU which uses DSRC for communication [24]. The service vehicles collect mobility information from other vehicles, and can exchange information with the RSUs, while the others can only drop a packet to an RSU. The reason we only use the service vehicles to collect mobility information is to prevent the general vehicles from suffering too much communication overhead. Moreover, since the service vehicles have large coverage on the road network, they are effective in distributing mobility information.

According to the definition provided by US Department of Transportation [26], an RSU is a transceiver that communicates with OBUs or other mobile devices. Therefore, we specify that the RSUs in the VDTN context only stores information and does scheduling. In contrast, the APs used in previous works [13]–[16] are interconnected through backbone wired network, which is not available in the VDTN scenario. There are existing works focusing on motivating users to share mobility information [27]–[29] and ensuring users' privacy [30]. We leave the work for MobiT as our future work. MobiT is a distributed routing algorithm. It leverages the following properties of vehicle networks.

- Urban vehicle trips have short duration but cover several landmarks (Figure 2(a), Figure 2(b)).
- Vehicles regularly follow routines (Figure 3(a)). Forwarding packet to the routine of the packet's destination vehicle helps increase delivery success rate.
- Vehicles share similar routines with their friends (Figure 3(b)). Forwarding packet to the routine of the packet's destination vehicle's friends help delivery.
- Urban vehicles generally move within a few sub-districts (Figure 4(a)), while service vehicles move among many sub-districts (Figure 4(b)). Therefore, service vehicles are suitable for disseminating information.

First, in Section IV-A, we make an overview of MobiT under various cases. Then, we present the representation of mobility information in Section IV-B. In Section IV-C, we elaborate how the mobility information is disseminated. Finally, in Section IV-D, we illustrate the routing process.

A. System Overview

In this section, we briefly introduce the operating process of MobiT using Figure 5, Figure 6 and Figure 7. In these figures, the service vehicle is the one with stable trajectory and wide road coverage (e.g., buses, delivery trucks or garbage trucks). The friend vehicle is the one that shares long-term mobility with the destination vehicle, the criteria for determining friendship is specified in Section IV-B2.

We let service vehicles and RSUs cooperatively collect and manage mobility information of vehicles, which includes the vehicles' short-term and long-term movements. Each vehicle reports its trajectory when it meets a service vehicle. When a service vehicle passes by an RSU, it exchanges mobility information in the RSU. The RSU can only exchange information with service vehicles and accept packet. Later, the source vehicle can request the service vehicle or the RSU to calculate next forwarders and the encounter positions with its packet's destination vehicle. For example, the service vehicle in Figure 5 circulates around the four intersections: A, B, C and D. It has the mobility information of the vehicles it meets on movement, such as the destination vehicle, the relay vehicle, etc. When passing by an RSU, the service vehicle will drop the information to the RSU.

Vehicles' trajectory is the short-term mobility information in MobiT. It indicates the vehicles' movement in the near future. In MobiT, a vehicle generates its initial trajectory at the beginning of trip. The trajectory consists of positions the vehicle will pass by and corresponding arrival times at the positions. Then the routing of a packet can be scheduled by considering the trajectory information collected from various vehicles. For example, in Figure 6, the source vehicle wants to send a packet to the destination vehicle. So the source vehicle requests the RSU at A to calculate the forwarder and the encounter position between the forwarder and the destination vehicle. Since the RSU has the trajectory information of the relay vehicle and the destination vehicle, which is dropped by the service vehicle, it knows both vehicles will pass by B. By checking road congestion state of the roads from C to B, the RSU determines the destination vehicle will arrive at B at around 08:30. By checking road congestion state of the roads from A to B, the RSU determines the relay vehicle will arrive at B no later than 08:30. Thus, the relay vehicle can carry the packet to B prior to the destination vehicle's arrival. After arrival, the packet will wait at an RSU at B until the destination vehicle passes by. The details of calculating the encounter positions and travel time are in Section IV-D.

Since vehicle mobility information is disseminated in a distributed manner, it is not always available. MobiT also utilizes the destination vehicle's long-term mobility information, which includes routine and friendship, to provide clues for deducing its movement. Routine represents the roads regularly followed by vehicles during certain times, friendship quantifies vehicle-to-vehicle relation based on vehicles' similarity on

routine. For example, in Figure 7, the destination vehicle's routine at 08:30 is moving from C to B. Meanwhile, the RSU knows the relay vehicle will be moving to B. So the RSU tells the source vehicle to forward the packet to the relay vehicle, expecting the relay vehicle to meet the destination vehicle or find the short-term mobility information of the destination vehicle near B. If the routine is also unknown, the RSU will look for the mobility information of the destination vehicle's friend, which shares common routine with the destination vehicle at around 08:30. The RSU finds the friend vehicle is moving from C to B at this time. Then the source vehicle forwards the packet to the relay vehicle, expecting it will meet the destination vehicle on way. The details of the representation of mobility information are introduced in Section IV-B.

In summary, MobiT is a distributed packet routing algorithm for VDTNs. It utilizes short-term, long-term mobility information and congestion state to schedule the forwarding of packets. Additionally, it uses service vehicles and RSUs to collect and manage mobility information, which prevents participating vehicles from heavy communication overhead.

B. Representation of Vehicle Mobility

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 as follows: $\langle N_i; \{p_i^0, p_i^1, \ldots, p_i^Q\}; T_s\rangle$, 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: As mentioned before, in previous trajectory-based algorithms [13]–[16], the estimated travel time to the positions of a vehicle's trajectory may be outdated due to random traffic issue. To keep the trajectories updated, vehicles need to report their trajectories periodically to APs. This is not suitable for MobiT because schedulers (i.e., service vehicles and RSUs) have limited resources and cannot frequently communicate with many vehicles. Instead, MobiT let schedulers store and use road segment congestion state to update the travel time to the rest of the trajectory of each encountered vehicle.

We use a 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 [31], [32]. Specifically, for a road segment s_i, its reachable speed is related to a vehicle density limit d_i^m. When the vehicle density is below d_i^m, vehicles on the road segment can drive with the free flow speed (i.e., speed limit, denoted by v_i^{max}). If the vehicle density exceeds d_i^m, the road segment will be congested and the vehicles have to drive with the congested speed (denoted by v_i^{min}). d_i^{jam} is the vehicle density that will cause s_i to be completely jammed. d_i^m can be obtained from field observation, and d_i^{jam} can be obtained from the road network's designed capacity [32]. Thus, the travel time of s_i under its current vehicle density d_i is calculated as:

$$\widetilde{t}_i = \begin{cases} l_i/v_i^{max}, & 0 \leqslant d_i < d_i^m \\ l_i/v_i^{min}, & d_i^m \leqslant d_i < d_i^{jam} \\ \infty, & d_i \geqslant d_i^{jam} \end{cases} \quad (1)$$

It is also noticeable that urban traffic pattern repeats in daily fashion [33]. 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 the table of congestion states, which uses binary vectors to describe road congestion.

For each road segment, it has distinct travel times corresponding to congested and non-congested situations [31]. The totally jammed road segments are excluded from routing consideration. We adopt existing works [31], [34] to determine whether a segment is congested or not, and calculate the travel time of an individual segment. For example, *College Ave* has 6 segments as shown in Table I. 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 I: Table of delays.

Road name	*College Ave*					
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 II. In Table II, *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+0.4×7min30s=11min36s.

TABLE II: Table of congestion states.

Road name	*College Ave*
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 [31], [35], [36].

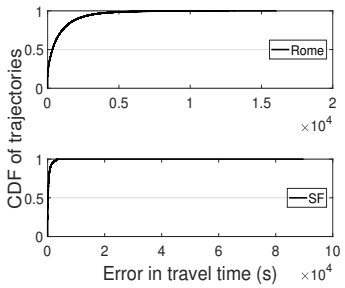


Fig. 8: Error between estimated travel time and actual travel time.

The table of congestion states and delays for all roads are computed offline. All schedulers are preloaded with the two tables. Note that the travel time of the road segments estimated in this way will be somewhat inaccurate due to various factors (e.g., traffic congestion, inaccuracy of the measured vehicle density). Thus, we analyze the deviation of the estimated travel time. In routing, we also aim to let the packet always arrive at the encounter position prior to the destination vehicle to tolerate the inaccuracy of this travel time estimation.

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 II for the road's current congestion state. Then, the scheduler refers to Table I 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 I and II, the travel time of *College Ave* has a probability of 0.6 to be 14min20s and a probability of 0.4 to be 7min30s. Therefore, the road's estimated travel time is $\mu = 0.6 \times 14$min20s+$0.4 \times$ 7min30s=11min36s, with standard deviation $\sigma = \sqrt{0.6 \times (14min20s - 11min36s)^2 + 0.4 \times (11min36s - 7min30s)^2} = 3min20s$. To demonstrate that the accuracy of the estimation of the travel time is sufficiently high, we measured the error between the estimated travel time and the actual travel time for all the vehicle trajectories in Rome and San Francisco. The results are illustrated in Figure 8. We can see that for Rome, more than 80% of the errors are lower than 1,000s, and for San Francisco, more than 99% of the errors are lower than 1,000s. Recall that we let the packet always arrive at the encounter position prior to the destination vehicle, we conclude the estimated travel time is accurate enough.

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

Routine: Vehicles' long-term mobility has regularity [12], 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 III.

TABLE III: Table of routines.

Vehicle ID	N_i		
Prob	*Route*	T_s	T_e
0.6	$\{p^0_{i1}, \ldots, p^m_{i1}\}$	*08:10 ∼ 08:20*	*08:30 ∼ 08:45*
0.2	$\{p^0_{i2}, \ldots, p^n_{i2}\}$	*13:00 ∼ 13:20*	*13:30 ∼ 13:45*

In Table III, each row represents a routine of N_i. *Route*

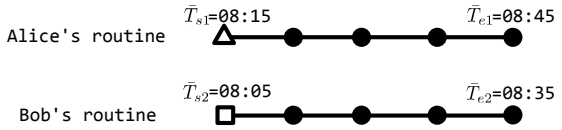


Fig. 9: Overlapping of routines.

represents the sequence 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 of each routine indicates the likelihood that the vehicle will follow the routine and is calculated as

$$P_{R_t}(N_i) = m_t/M \tag{2}$$

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:

$$\begin{aligned} \left|\bar{T}_{e1} - \bar{T}_{e2}\right| &< \tau_t \\ \left|\bar{T}_{s1} - \bar{T}_{s2}\right| &< \tau_t \end{aligned} \tag{3}$$

$$\frac{|r_1 \bigcap r_2|}{|r_1 \bigcup r_2|} > \gamma_s \tag{4}$$

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. To find the best values for these parameters, we vary each variable within certain range (e.g., [0.3, 0.8] for α_f, [10 min, 60 min] for τ_t, and [0.4, 0.9] for γ_s) and test different combinations of the values. Specifically, we use each combination to determine the friendship, and run our routing experiment with only using the mobility information of the destination vehicle's friends. Then, we choose the combination of values that results in the maximum success rate of delivery as the final setting. Finally, we found $\alpha_f = 0.5$, $\tau_t = 15$ min and $\gamma_s = 0.6$

are the best parameters for Rome and San Francisco. For example, suppose both Alice and Bob only have one routine, and their routines are as illustrated in Figure 9. We can see the temporal deviations of the start times and end times of the routines are $\left|\bar{T}_{s1} - \bar{T}_{s2}\right| = 10$ min and $\left|\bar{T}_{e1} - \bar{T}_{e2}\right| = 10$ min, respectively; the spatial deviation of the positions of the routines is $\frac{|r_1 \bigcap r_2|}{|r_1 \bigcup r_2|} = 0.67$. Therefore, these two routines are similar, and Alice and Bob are friends to each other.

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 IV.

TABLE IV: 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	$N_0(0.4), N_1(0.2)$
...	...

C. Mobility Information Collection and Distribution

Recall each vehicle reports its initial trajectory to nearby service vehicle or RSU only once. Service vehicles usually cover arterial roads [37]. Therefore, they are likely to encounter more vehicles than general vehicles. Also, as shown in Figure 4(b), service vehicles move among a wide range of sub-districts. So they are likely to stretch into majority of the corners of the road network. Therefore, MobiT utilizes service vehicles as the collector and distributor of mobility information. Based on the collected historical short-term mobility information, the schedulers count the frequencies of repeated trajectories and prepare the table of routines for each vehicle by the methods of determining routines in Section IV-B2. Also, by comparing the routines of every two vehicles, the schedulers determine vehicle-to-vehicle friendship. In addition to short-term mobility information, the schedulers exchange their deduced long-term mobility information during each other's encounter.

MobiT combines service vehicles with RSU to maximally make the mobility information discoverable to schedulers. The procedures of the distribution of the mobility information to local areas can be summarized as follows:

(1) Whenever a service vehicle N_r meets another service vehicle N_t, they exchange mobility information.
(2) Whenever N_r approaches an RSU, N_r copies its mobility information to the RSU.

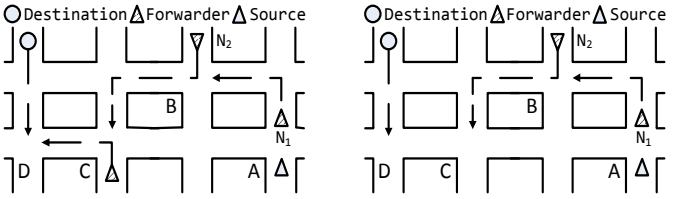
Therefore, as long as a general vehicle N_i's mobility information is carried to a sub-district, it will be kept in a local service vehicle or RSU. The service vehicles and RSUs (i.e., schedulers) are also responsible for updating vehicles' trajectories. Each time when a scheduler encounters a general vehicle N_i, it checks if it has N_i's trajectory. If affirmative and the trajectory is calculated within one hour, no update is needed because the congestion table is for each hour. Otherwise, it updates the rest of the trajectory. The distribution of the mobility information needs a period of time for initialization. Since the service vehicles generally start operating early in the morning (e.g., 05:30), and the rush hour of a city road network usually starts at $7:30$ in a daily manner [38], we set the initialization time of short-term mobility information (i.e., initial trajectory) distribution to be 2 hours. In addition, since the long-term mobility information among vehicles is relatively more stable, we set the initialization time of long-term mobility information distribution (i.e., friendship, routines) before general vehicles' usage of MobiT to be 7 days. During the initialization time, we let the service vehicles determine vehicle-to-vehicle friendship and routines based on the general vehicles' historical movement records, and fully exchange and update their stored long-term mobility information. Note that for different cities, the initialization time can be adjusted accordingly.

On the one hand, due to certain practical communication limitations (e.g., short channel coherence time, encryption-decryption delay), transferring small packets is preferred for vehicular communication [39]. On the other hand, since we aim to enable the packet to arrive at the encounter position prior to the destination vehicle, we actually only need the future positions of the destination vehicle to schedule the encounter. Therefore, in our vehicular communication, we limit the number of transferred landmarks in a trajectory to be lower than a threshold Q_{th} (e.g., 200), and a vehicle only transfers the last Q_{th} landmarks in its trajectory (i.e., future positions of the vehicles). Note that Q_{th} can be adjusted according to system requirements (e.g., communication constraint). Thus, we not only ensure that the packet size will not be too large, but also ensure that the trajectories stored in the packets are useful for scheduling the encounter.

Without loss of generality, we analyze the probability that the mobility information of a vehicle N_i can be stored in a sub-district (say A_m) to show that the mobility information is easily discoverable. It is calculated as $P_{d_i}(A_m) = 1 - \prod_{r=1}^{K}(1 - P_t(\lambda_r)P_e(A_m))$, where $P_{d_i}(A_m)$ is the probability of discovering the mobility information of N_i in A_m; K is the total number of service vehicles that have N_i's mobility information and can move to A_m; λ_r is service vehicle N_r's trajectory which will pass A_m; $P_t(\lambda_r)$ is the probability that N_r will follow trajectory λ_r in its movement; $P_e(A_m)$ is the probability that N_r can leave N_i's mobility information in A_m. Since trajectory is an explicit indicator of a vehicle's movement, $P_t(\lambda_r) \approx 1$ in most cases. The wide movement range of service vehicles guarantees there can be several of them moving through A_m at different times, namely K can be large enough. With the above information collection method, $P_e(A_m) \approx 1$, which makes mobility information easily discoverable.

D. 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



(a) Complete list of forwarders. (b) Incomplete list of forwarders.

Fig. 10: Selection of forwarders.

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.

1) Short-term Mobility Based Routing: For trajectory-based routing path determination, MobiT extends its predecessor, STDFS [13]. 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.

Routing with a complete list of forwarders: As described in Section IV-B1, for each road covered by a trajectory, we can calculate estimated travel time μ with deviation σ. Suppose the means and the deviations of the travel times of the M roads included in a chain of trajectories are $\{\mu_1, \mu_2, \ldots, \mu_M\}$ and $\{\sigma_1^2, \sigma_2^2, \ldots, \sigma_M^2\}$, respectively. Suppose the total expected travel time of the chain is U with deviation V. Since the travel times of roads are normally distributed and statistically independent [40], so $U = \sum_{i=1}^{M} \mu_i$, and $V^2 = \sum_{i=1}^{M} \sigma_i^2$. According to the "68-95-99.7 rule" of normally distributed random variables [41], the actual travel time of the chain has the probability of 68% to be within the range $[U-V, U+V]$. This helps to find a chain that can drop packet in an RSU prior to the time the destination vehicle passes by the RSU.

From the constructed encounter graph, the scheduler may find several chains of trajectories that connect the current position of the source vehicle and each of the encounter positions on the destination vehicle's trajectory. As shown in Figure 10(a), the packet can be forwarded by N_1, N_2 and N_3 along path $A \rightarrow B \rightarrow C \rightarrow D$. For the destination vehicle, its travel time (denoted by U_r) with deviation (denoted by V_r) from its current position (denoted by p_c) to an encounter position (denoted by p_e) can be estimated as explained above. For each encounter position, the wait time of a packet forwarded by a chain in the worst case equals $T_w = U_r - V_r - (U+V)$, where U and V are the expected travel time and its deviation, respectively. The scheduler firstly filters out the late chains that have $T_w \leq 0$. Then, from the remaining chains, it selects the one with the shortest $U+V$ (i.e., shortest delivery delay) as the routing path. After arrival, the packet waits in the nearby RSU for the meeting with the destination vehicle.

Routing with incomplete list of forwarders: A scheduler may not find a chain of trajectories that completely connects the source vehicle with the destination vehicle due to partial mobility information and distant destination vehicle. In this case, it finds a list of forwarders that can forward the packet to a position, say p_n, close to the destination vehicle's trajectory. As shown in Figure 10(b), the packet can be forwarded by N_1 and N_2 along path $A \rightarrow B \rightarrow C$. Obviously, the estimated delivery delay should be no longer than $U_r - V_r$ so the packet still has time to encounter its destination vehicle. Finally, we need to minimize the routing delay of the incomplete path and the delay from p_n to the destination vehicle.

When a scheduler cannot find a vehicle succeeding the current vehicle N_i (i.e., the last vehicle of a path flow from the source) that can drive to the destination vehicle closer than the position p_i that N_i can carry the packet to, this chain of trajectories becomes incomplete and p_i becomes the final position that the packet can be delivered by this path. Suppose the scheduler finds C incomplete chains of trajectories and their final delivery positions are denoted by $p_i (i = 0, 1, \ldots, C)$. For each p_i, the scheduler estimates the delivery delay U_1^i with deviation V_1^i. The scheduler then uses the map to find the geographically shortest path from p_i to an encounter position, p_e, on the destination vehicle's trajectory. Using the arrival time on p_i, the scheduler also estimates the delay of the shortest path U_2^i with deviation V_2^i. The incomplete list of forwarders needs $U_1^i + V_1^i$ to deliver the packet to position p_i and $U_2^i + V_2^i$ is needed to finish the rest path to the destination vehicle.

To make sure that the packet arrives at p_e no later than the destination vehicle, the scheduler filters out the chains with $U_1^i + V_1^i + U_2^i + V_2^i \leq U_r - V_r$. Then, the scheduler ranks the chains of trajectories by p_i's distance to p_e from short to long. The top 30% chains are selected as candidates. Finally, the scheduler selects the chain with the shortest $U_1^i + V_1^i + U_2^i + V_2^i$ from these candidates to ensure short delivery delay.

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 III) and friend table (Table IV)) 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 III. 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 IV 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

Algorithm 1 Routing executed by a general vehicle N_i.

```
if start a new trajectory then
    Report its trajectory to its first encountering scheduler
repeat
    if need to send a packet to destination vehicle N_j (source or forwarder) then
        if encounter a scheduler N_s then
            Request N_s for a routing chain to destination vehicle N_j
            if has no routing chain or this new chain is better then
                Use this new chain
until arrive at the scheduled position p_i
if encounter the next forwarder at the scheduled position then
    Forward the packet to the forwarder
else
    Drop the packet to a nearby RSU for further routing schedule or waiting for destination vehicle N_j
```

the highest ratio of similarity with N_1. Then the friend's mobility information will be used as previously described.

3) Routing without Mobility Information: It is likely the scheduler does not have any useful mobility information. If the scheduler is a service vehicle, it will keep the packet. If the scheduler is an 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. Algorithm 1 shows the pseudocode of the routing algorithm.

Although we aim to forward every packet to the scheduled encounter position with the destination vehicle, it is still possible that some packets are delivered to an RSU which is close to the encounter position but no service vehicles will pass by for a long time due to certain exceptions (e.g., abnormal traffic). To solve this problem, we set a staying time threshold for the packets (e.g., 30 minutes). If a packet has stayed in some RSUs longer than this threshold, it can be piggybacked by any general vehicles passing by and be forwarded to a service vehicle whenever it is possible. The threshold can be adjusted according to the hourly average waiting time of the successfully delivered packets in the RSUs. Specifically, we periodically (e.g., after every 5 minutes) calculate the average waiting time of all the successfully delivered packet in the previous hour.

V. Performance Evaluation

We compared MobiT with two representative algorithms: the Shared-Trajectory-based Data Forwarding method (*STDFS* in short) [13], and the Robust Replication Routing (denoted by *R3*) [9]. *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* [13]. We measured following metrics:

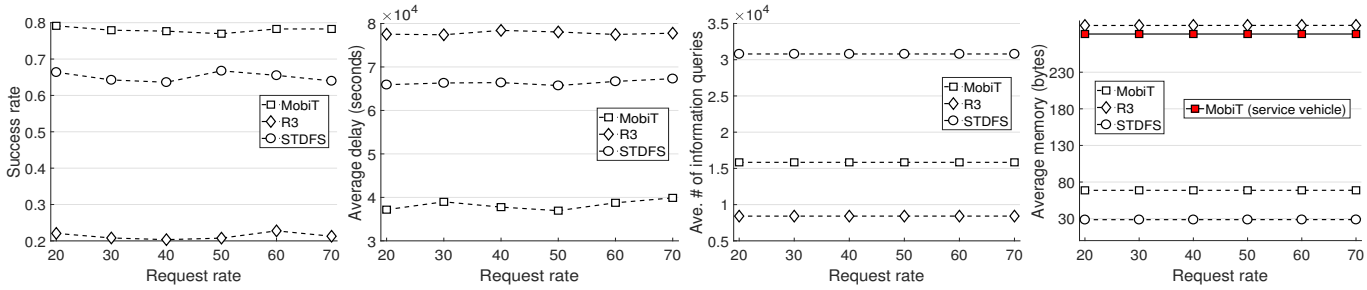
- *Success rate*: The percentage of packets that successfully reach their destination vehicles. Specifically, after the simulation, the success rate of packet delivery is calculated as the number of packets that have reached the communication range of their destination vehicles within the packet TTL, divided by the total number of packets sent out during the simulation. The packets that fail to reach their destination vehicles within the packet TTL are counted as packet loss due to signal propagation.
- *Average delay*: The average time (in seconds) used by packets to reach their destination vehicles. Note that the delay of unsuccessful packets, namely the packet TTL, is also considered.
- *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 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.

A. Simulation Experiment

We used the Rome [18] and the San Francisco [17] traces introduced in Section III for evaluation. We developed a trace-based simulation environment which is driven by each vehicle's movement event [42], [43]. Unless otherwise specified, the experiment setting is the same as that in Section III.

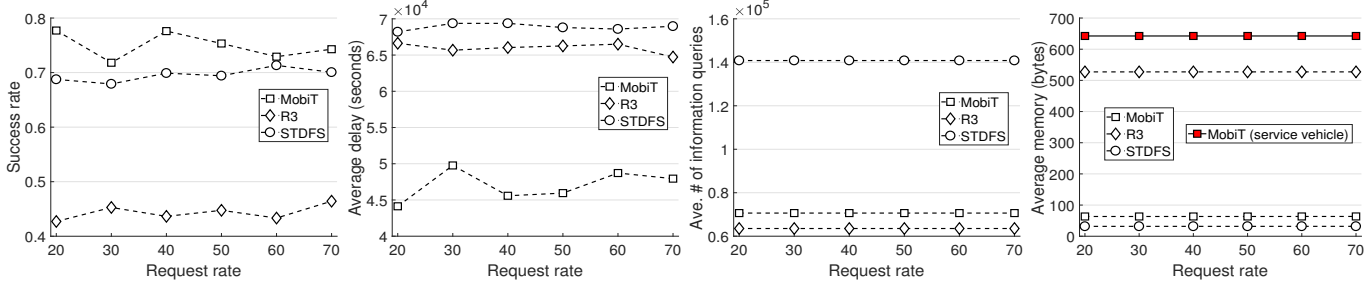
The collection of congestion table and delay table was finished offline. For Rome and San Francisco, the threshold speeds to determine congestion are 20MPH and 30MPH, respectively [17], [18]. The congestion state of each road segment was sampled per hour. The communication range of the vehicles is set to 100m according to the DSRC specifications [22], [44]. 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 by Equation (2), and determined friendship between vehicles by Equation (3) and (4) 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 conducted two experiments. In one experiment, to simulate scenarios with different levels of routing request pressure, we varied the request rate from 20 to 70 with 10 as the step size. In the other experiment, to explore the influence of packet TTL on the metrics, we varied the packet TTL from 18 hours to 33 hours with 3 hours as the step size.



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

Fig. 11: Performance with different request rates using the Rome trace.



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

Fig. 12: Performance with different request rates using the San Francisco trace.

1) Success Rate: Figure 11(a) and Figure 12(a) show the success rates of the algorithms under different request rates in Rome and San Francisco, respectively. Figure 13(a) and Figure 14(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.

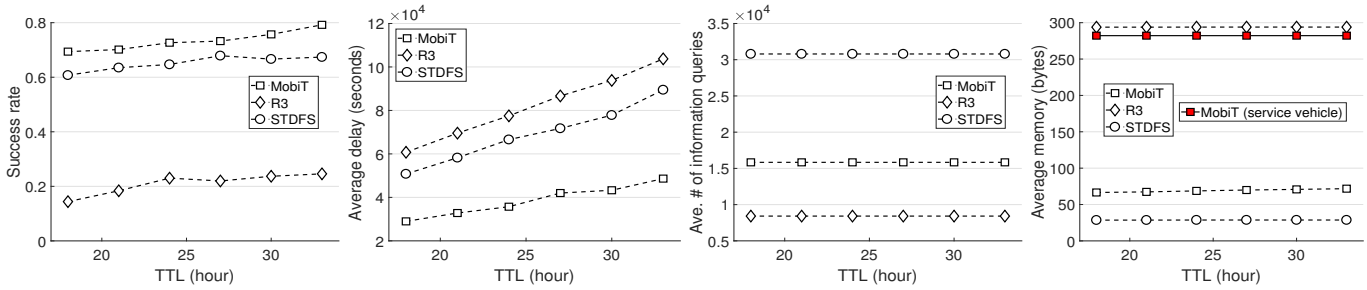
MobiT always achieves the highest success rate. This is because *MobiT* considers road congestion state in estimating vehicles' arrival times on trajectory, which makes the estimation more tolerant to traffic change. Also, *MobiT* is not afraid that trajectory may be outdated because it uses road delay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

2) Average Delay: Figure 11(b) and Figure 12(b) show the average delay of the algorithms under different request rates in Rome and San Francisco, respectively. Figure 13(b) and Figure 14(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* always achieves the best delay 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.

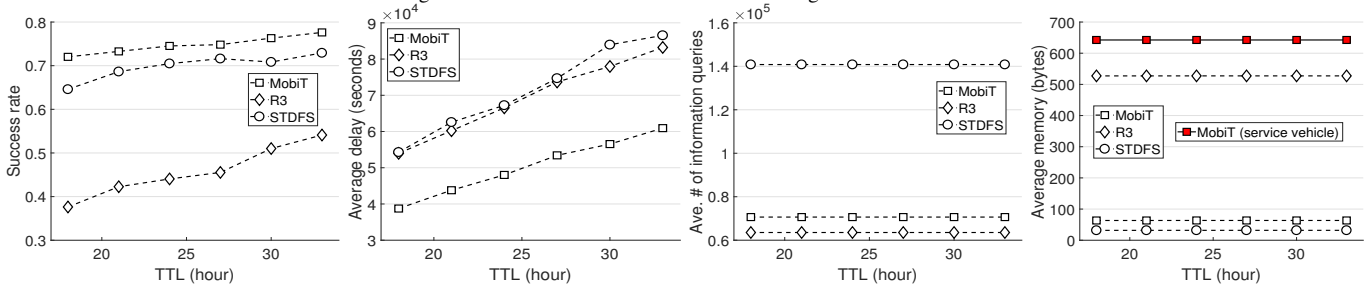
3) Average Number of Information Queries: Figure 11(c) and Figure 12(c) show the average number of information query of the algorithms under different request rates in Rome and San Francisco. Figure 13(c) and Figure 14(c) show the metric of the algorithms under different packet TTLs in both traces. The 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) Average Memory Usage: Figure 11(d) and Figure 12(d) show the average memory usage of the algorithms under different request rates in Rome and San Francisco, respec-



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

Fig. 13: Performance with different TTLs using the Rome trace.



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

Fig. 14: Performance with different TTLs using the San Francisco trace.

tively. Figure 13(d) and Figure 14(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". In Rome, the metric follows: *R3*>*Service*>*MobiT*>*STDFS*. While in San Francisco, the metric follows: *Service*>*R3*>*MobiT*>*STDFS*. We can see that the memory usage of general vehicle is comparable to the one of *STDFS*.

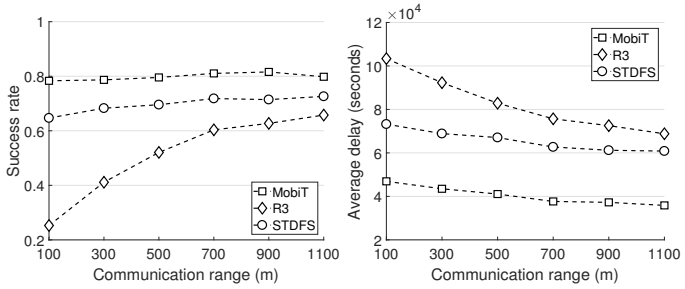
Since *R3* has duplicated packets, and each vehicle needs to maintain the distribution of path's historical delays, vehicles in *R3* have the highest memory usage. In *MobiT*, general vehicles only need to maintain short-term mobility information and occasional packets. Therefore, *MobiT* uses much lower 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 forward 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) Impact of Communication Range: To demonstrate the impact of vehicle-to-vehicle communication range on performance, we measured the success rate and the delay of packet delivery under various communication ranges. The TTL of the packets is set to 33 hours, and the request rate is set to 70. Figure 15(a) and Figure 15(b) show the success rates and the average delivery delays of the algorithms under different communication ranges in Rome, respectively. Figure 16(a) and Figure 16(b) show the success rates and the average delivery delays of the algorithms under different communication ranges in San Francisco, respectively. The success rates follow: *MobiT*>*STDFS*>*R3*, and the delivery delays generally follow: *R3*>*STDFS*>*MobiT*. We can see that *MobiT* achieves the highest success rate and lowest delivery delay in both traces.

With the increasing of the communication range, the success rates of the three methods generally increase, which is especially obvious for *R3*. This is because that *R3* does not know the possible position of the destination vehicle, so increasing the communication range can greatly increase the likelihood of finding the destination vehicle for the source vehicle. However, we can also notice that the increasing of the success rates is slowing down, which verifies the low efficiency of *R3*.

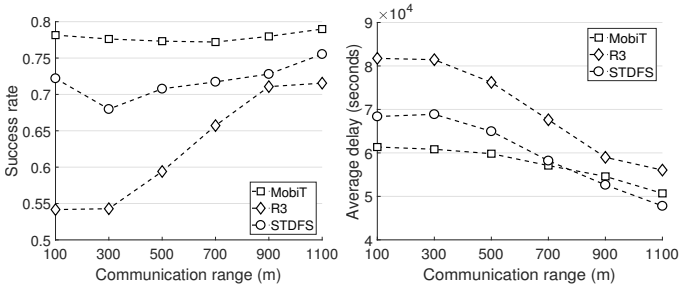
In comparison, the increasing of success rates of *STDFS* and *MobiT* is much flatter. This is because that they utilize the future position of the destination vehicle to realize packet delivery, increasing the communication range will not drastically benefit their probability of finding the destination vehicle. However, when the destination vehicle's movement is influenced by road congestion, or its trajectory in central server is not continuously updated, the delivery of packet in *STDFS* will fail, which results in its lower success rate than *MobiT*. The same reasons apply for the results of delivery delays.

6) Effectiveness of Different Mobility Information: To demonstrate that the mobility information is helpful in assisting the source vehicle to find the destination vehicle, we also measured the durations (seconds) of using short-term and long-term mobility information in the delivery of all successful packets in both traces. The TTL of the packets is set to 33 hours, and the request rate is set to 70. The measurement results are illustrated in Figure 17. We can see that for Rome, in the delivery of about 50% of the packets, the duration of utilizing short-term mobility information for packet routing is longer than 2 hours. In contrast, in the delivery of only about 20% of the packets, the duration of utilizing long-term mobility information for packet routing is longer than 1 hour. This is because that Rome's road segments are quite crowded at popular sites and have short distance [18], which makes the



(a) Success rate. (b) Average delay.

Fig. 15: Rome trace.



(a) Success rate. (b) Average delay.

Fig. 16: San Francisco trace.

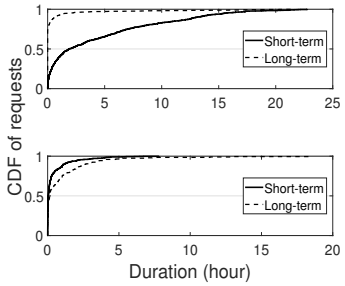


Fig. 17: Durations of using short-term and long-term mobility information.

short-term mobility information less likely to be outdated and enables the information to be shared among the schedulers (i.e., service vehicles, RSUs) more quickly. While for San Francisco, in the delivery of about 50% of the packets, the duration of utilizing long-term mobility information for packet routing is longer than 1 hour. In contrast, in the delivery of only about 20% of the packets, the duration of utilizing short-term mobility information for packet routing is longer than 1 hour. This is because that the road segment distribution of San Francisco is more uniform than that in Rome [17], and the service vehicles are more sparsely distributed in the road network, which makes it harder for the source vehicle to know the short-term mobility information of the destination vehicle.

B. Real Experiment

In this section, we introduce the details and results of a real experiment of MobiT in a university campus. 4 vehicles and one RSU participated in this experiment.

1) Experiment Settings: The experiment lasts for 40min. Each vehicle drives approximately at the speed of 20MPH, which is the speed limit of the campus. Under such settings, we consider a vehicle is in congestion if its driving speed is lower than 10MPH. Each vehicle is equipped with a laptop that can transmit and receive packets within the transmission range around 70m$\sim$ 80m. We use User Datagram Protocol (UDP) [45] as the transmission protocol between laptops. The communication range of a laptop netcard is around $70 \sim 80m$, which makes it qualified as a DSRC device. Figure 18(a) shows the experiment site, which consists of two circle routes and one connection route. We use 9 positions as landmarks, as numbered in Figure 18(b). The laptops are placed in cars as shown in Figure 18(c). The RSU, as shown in Figure 18(d), is placed at landmark 6. Each route is represented with a series of these landmarks. Vehicles need about 30s to move between neighbor landmarks. Specifically, V_1 follows route: $4 \rightarrow 5 \rightarrow 6 \rightarrow 1 \rightarrow 2 \rightarrow 3$; V_2 follows route: $9 \rightarrow 7 \rightarrow 6 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 8$; V_3 follows route: $9 \rightarrow 8 \rightarrow 7 \rightarrow 6 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7$; V_4 follows route: $9 \rightarrow 7 \rightarrow 8$. V_2 and V_3 are service vehicles. Vehicles generate and update trajectories since the beginning of movement. Routes are tagged with timestamps at the moment vehicles start moving, which generates trajectories. Routines are deduced by the service vehicles. From the definition of friendship, we see that the friends of V_1 are: V_2 and V_3; the friends of V_2 are: V_1 and V_3; the friends of V_3 are: V_1 and V_2; while V_4 has no friends. For comparison, we also implemented *R3* and *STDFS* in our experiment.

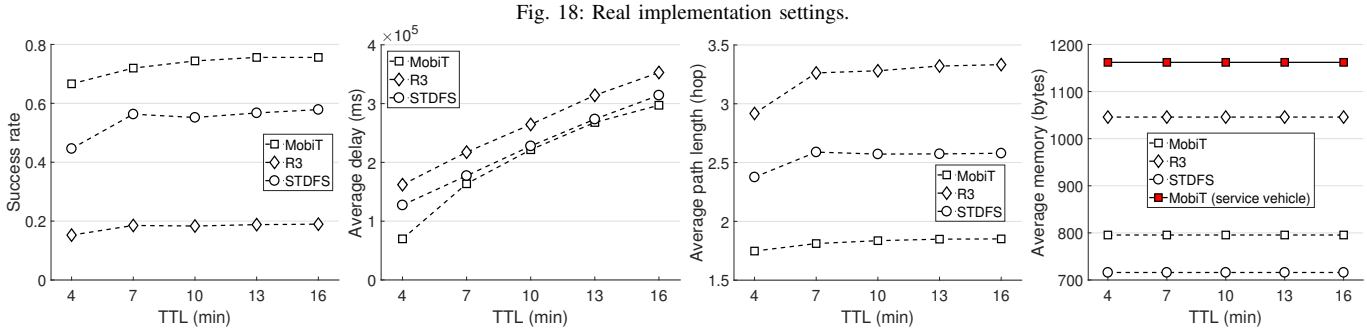
2) Experimental Results: We let vehicles fully exchange mobility information during initial period, which lasts for $6min$. After the initial period, each vehicle began to generate 70 packets every 6min until the end of experiment. Each packet randomly chose a vehicle as its destination vehicle. We recorded success status, TTL, timestamp of each forwarding of a packet and the hops the packet transited. For each vehicle, we also calculated the number of packets and mobility information it processed during the experiment. Since the experiment of our small-scale real-world prototype lasts for only 40min, we correspondingly shrink the TTL of packets. We varied the packet TTL from 4min to 16min with 3min as the step size. The results are as shown in Figure 19.

Success rate Figure 19(a) shows the success rates of packets in the three algorithms under various packet TTLs. We see that after the packet TTL is larger than $13min$, the success rate of *MobiT* stabilizes at around 75%, the success rate of *STDFS* stabilizes at around 58%, and the success rate of *R3* stabilizes at around 20%. We found that most failed packets have encounter positions between landmark 7, 8 and 9. Since there is no RSU at this road section, these packets have to move with carriers. In *R3*, the possible position of the destination vehicle is unknown, and the packet can only be delivered when it directly meets the destination vehicle, which results in a very low success rate. In *STDFS*, the packets are delivered to the scheduled position to encounter with the destination vehicle, so the success rate is greatly improved. In *MobiT*, with the help of short-term and long-term mobility information of the destination vehicle, the packets always move with the carriers that have high likelihood to meet the destination vehicle, which results in the highest success rate.

Average delay Figure 19(b) shows the average delays of all packets and successfully delivered packets in the three algorithms under various packet TTLs. Note that the average delay is measured in milliseconds. In contrast to the success rates, the average delays of the algorithms are more similar

(a) Map of experiment site. (b) Landmarks. (c) Laptop in vehicle. (d) RSU setting.

Fig. 18: Real implementation settings.



(a) Success rate. (b) Average delay. (c) Average path length. (d) Memory usage of vehicle.

Fig. 19: Performance with different TTLs in real implementation.

to each other, and *MobiT* still achieves the shortest delay. This is largely caused by the small scale of the road network. Similar to the success rate, the delay of successfully delivered packets stabilizes after the packet TTL reaches $13min$, while the delay of all packets keeps increasing. It is because that most successfully delivered packets can arrive at their destination vehicle within $13min$, while unsuccessful packets' delay increases with TTL.

Average path length Figure 19(c) shows the average number of hops a packet transited in the three algorithms. We can see that *MobiT* always has the shortest path length. Note that the difference under various packet TTLs is tiny. Note that the path length increases fast from $4min$ to $7min$ but stabilizes after $13min$. This is because when packet TTL is small, many packets expired on the way to their destination vehicles. When packet TTL is larger than $13min$, most packets either reached their destination vehicles or are carried by the last forwarder. This shows the outstanding performance of MobiT in selecting forwarders.

Memory usage of vehicles Figure 19(d) shows the average memory usage of vehicles in the three algorithms and schedulers of *MobiT* (i.e., service vehicles and RSUs) under different TTLs. The reasons are the same as those explained in Section V-A4.

VI. Conclusion

Message delivery is important in VDTNs. Previous opportunistic routing algorithms cannot achieve high success rate and low delay due to inaccurate 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. Our trace analysis provides foundation for the design of MobiT. MobiT aims to let the packet arrive at an RSU prior to the destination vehicle by scheduling based on vehicle short-term trajectory and long-term mobility patterns including the routines routes of itself and its common-route vehicles. By taking advantage of traveling 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. Our trace-driven and real-world experiments show MobiT's higher efficiency and effectiveness over previous algorithms. In the future, we will further exploit vehicles' relationship in routing.

Acknowledgements

This research was supported in part by U.S. NSF grants OAC-1724845, ACI-1719397 and CNS-1733596, and Microsoft Research Faculty Fellowship 8300751.

References

[1] Y. Wei, Z. Yu, and Y. Guan, "COTA: A robust multi-hop localization scheme in wireless sensor networks," in *DCOSS*, 2006.
[2] O. Choi, S. Kim, J. Jeong, H.-W. Lee, and S. Chong, "Delay-optimal data forwarding in vehicular sensor networks," *IEEE TVT*, vol. 65, no. 8, 2016.
[3] B. Jarupan and E. Ekici, "Prompt: A cross-layer position-based communication protocol for delay-aware vehicular access networks," *Ad Hoc Networks*, vol. 8, no. 5, 2010.
[4] J. Kim, X. Lin, N. B. Shroff, and P. Sinha, "Minimizing delay and maximizing lifetime for wireless sensor networks with anycast," *IEEE/ACM TON*, vol. 18, no. 2, 2010.
[5] E. C.-H. Ngai, J. Liu, and M. R. Lyu, "An adaptive delay-minimized route design for wireless sensor–actuator networks," *IEEE TVT*, vol. 58, no. 9, 2009.
[6] Y. Ding and L. Xiao, "SADV: static-node-assisted adaptive data dissemination in vehicular networks," *TVT*, vol. 59, no. 5, 2010.
[7] S. Ishihara, N. Nakamura, and Y. Niimi, "Demand-based location dependent data dissemination in VANETs," in *Proc. of MobiCom*, 2013.

[8] A. Symington and N. Trigoni, "Encounter based sensor tracking," in *Proc. of MobiHoc*, 2012.

[9] X. Tie, A. Venkataramani, and A. Balasubramanian, "R3: robust replication routing in wireless networks with diverse connectivity characteristics," in *Proc. of MobiCom*, 2011.

[10] R. S. Schwartz, H. W. Van Dijk, and H. Scholten, "Towards opportunistic sensed data dissemination in vehicular environments," in *Proc. of PerCom*, 2011.

[11] L. Kong, X. Chen, X. Liu, and L. Rao, "Fine: Frequency-divided instantaneous neighbors estimation system in vehicular networks," in *Proc. of PerCom*, 2015.

[12] Y. Zhu, Y. Wu, and B. Li, "Trajectory improves data delivery in urban vehicular networks," *TPDS*, vol. 25, no. 4, 2014.

[13] F. Xu, S. Guo, J. Jeong, Y. Gu, Q. Cao, M. Liu, and T. He, "Utilizing shared vehicle trajectories for data forwarding in vehicular networks," in *Proc. of INFOCOM*, 2011.

[14] J. Jeong, S. Guo, Y. Gu, T. He, and D. H. Du, "Trajectory-based data forwarding for light-traffic vehicular ad hoc networks," *TPDS*, vol. 22, no. 5, 2011.

[15] J. Jeong, S. Guo, Y. Gu, T. He and D. H. Du, "Trajectory-based statistical forwarding for multihop infrastructure-to-vehicle data delivery," *TMC*, vol. 11, no. 10, 2012.

[16] J. Jeong, S. Guo, Y. Gu, T. He, and D. H. Du, "TSF: Trajectory-based statistical forwarding for infrastructure-to-vehicle data delivery in vehicular networks," in *Proc. of ICDCS*, 2010.

[17] M. Piórkowski, N. Sarafijanovic-Djukic, and M. Grossglauser, "A parsimonious model of mobile partitioned networks with clustering," in *Proc. of COMSNETS*, 2009.

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

[19] Y. Wu, Y. Zhu, and B. Li, "Trajectory improves data delivery in vehicular networks," in *Proc. of INFOCOM*, 2011.

[20] "Real time rome," http://senseable.mit.edu/realtimerome/, accessed Sep 27, 2015.

[21] "Muni system map in san francisco," https://www.sfmta.com/maps/muni-system-map, accessed Sep 27, 2015.

[22] C. Cseh, "Architecture of the dedicated short-range communications (DSRC) protocol," in *Proc. of VTC*, 1998.

[23] X. Cheng, A. Thaeler, G. Xue, and D. Chen, "TPS: A time-based positioning scheme for outdoor wireless sensor networks," in *Proc. of INFOCOM*, 2004.

[24] K. Liu, M. Li, Y. Liu, X.-Y. Li, M. Li, and H. Ma, "Exploring the hidden connectivity in urban vehicular networks," in *Proc. of ICNP*, 2010.

[25] Y. Zheng, Y. Liu, J. Yuan, and X. Xie, "Urban computing with taxicabs," in *Proc. of UbiComp*, 2011.

[26] J. Chang, "An overview of usdot connected vehicle roadside unit research activities," Tech. Rep., 2017.

[27] L. Yan, H. Shen, and K. Chen, "TSearch: Target-oriented low-delay node searching in dtns with social network properties," in *Proc. of INFOCOM*, 2015.

[28] K. Chen, H. Shen, and L. Yan, "Dsearching: Using floating mobility information for distributed node searching in dtns," *IEEE TMC*, vol. 15, no. 1, 2016.

[29] S. M. Hur, S. Mao, Y. T. Hou, K. Nam, and J. H. Reed, "Exploiting location information for concurrent transmissions in multihop wireless networks," *IEEE TVT*, vol. 58, no. 1, 2009.

[30] S. Marti, T. J. Giuli, K. Lai, and M. Baker, "Mitigating routing misbehavior in mobile ad hoc networks," in *Proc. of MobiCom*, 2000.

[31] J. P. Kharoufeh and N. Gautam, "Deriving link travel-time distributions via stochastic speed processes," *Transportation Science*, vol. 38, no. 1, 2004.

[32] Y. Xu, Q.-J. Kong, S. Lin, and Y. Liu, "Urban traffic flow prediction based on road network model," in *Proc. of ICNSC*, 2012.

[33] V. Jain, A. Sharma, and L. Subramanian, "Road traffic congestion in the developing world," in *Proc. of DEV*, 2012.

[34] U. Mori, A. Mendiburu, M. Álvarez, and J. A. Lozano, "A review of travel time estimation and forecasting for advanced traveller information systems," *Transportmetrica A: Transport Science*, vol. 11, no. 2, 2015.

[35] S. Sun, C. Zhang, and G. Yu, "A bayesian network approach to traffic flow forecasting," *ITS*, vol. 7, no. 1, 2006.

[36] Y. Zhou, S. Chen, Z. Mo, and Y. Yin, "Privacy preserving origin-destination flow measurement in vehicular cyber-physical systems," in *Proc. of CPSNA*, 2013.

[37] K. De Zoysa, C. Keppitiyagama, G. P. Seneviratne, and W. Shihan, "A public transport system based sensor network for road surface condition monitoring," in *Proc. of NSDR*, 2007.

[38] D. Zhang, J. Huang, Y. Li, F. Zhang, C. Xu, and T. He, "Exploring human mobility with multi-source data at extremely large metropolitan scales," in *Proc. of MobiCom*, 2014.

[39] M. Kim, J. Lim, H. Yu, K. Kim, Y. Kim, and S.-B. Lee, "ViewMap: Sharing private in-vehicle dashcam videos," in *Proc. of NSDI*, 2017.

[40] R. Li, H. Chai, and J. Tang, "Empirical study of travel time estimation and reliability," *Mathematical Problems in Engineering*, 2013.

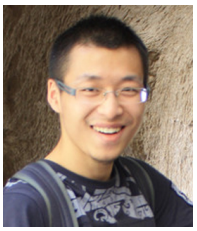
[41] F. Pukelsheim, "The three sigma rule," *The American Statistician*, vol. 48, no. 2, 1994.

[42] K. Chen and H. Shen, "Multicent: A multifunctional incentive scheme adaptive to diverse performance objectives for DTN routing," in *Proc. of SECON*, 2013.

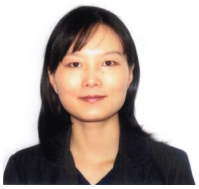
[43] K. Chen and H. Shen, "Smart: Lightweight distributed social map based routing in delay tolerant networks." in *Proc. of ICNP*, 2012.

[44] S. R. C. Shah, S. Roy, and W. Brunette, "Standard specification for telecommunications and information exchange between roadside and vehicle systems-5 ghz band dedicated short range communications (dsrc) medium access control (mac) and physical layer (phy) specifications." ASTM, 2003.

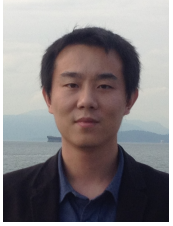
[45] J. Postel and U. D. Protocol, "RFC 768," *User datagram protocol*, 1980.



Li Yan Li Yan received the B.E. degree in Information Engineering from Xi'an Jiaotong University, China in 2010, and the M.S. degree in Electrical Engineering from University of Florida in 2013. He is currently a Ph.D. student in the Department of Computer Science at University of Virginia. His research interests include Cyber-Physical Systems and Wireless Networks, with an emphasis on Data-driven Intelligent Transportation Systems and Mobile Opportunistic Networks.



Haiying Shen Haiying Shen received the B.S. degree in Computer Science and Engineering from Tongji University, China in 2000, and the M.S. and Ph.D. degrees in Computer Engineering from Wayne State University in 2004 and 2006, respectively. She is currently an Associate Professor in the Department of Computer Science at University of Virginia. Her research interests include distributed computer systems and computer networks, with an emphasis on content delivery networks, mobile computing, wireless sensor networks, cloud computing, big data and cyber-physical systems. She is a Microsoft Faculty Fellow of 2010, a senior member of the IEEE, and a member of the ACM.



Kang Chen Kang Chen (S'13-M'15) received the BS degree in Electronics and Information Engineering from Huazhong University of Science and Technology, China in 2005, the MS in Communication and Information Systems from the Graduate University of Chinese Academy of Sciences, China in 2008, and the Ph.D. in Computer Engineering from the Clemson University in 2014. He is currently an Assistant Professor in the Department of Electrical and Computer Engineering at Southern Illinois University. His research interests include emerging wireless networks and software defined networking.