DIVERT: A Distributed Vehicular Traffic Re-Routing System for Congestion Avoidance

Susan Juan Pan, Iulian Sandu-Popa, Cristian Borcea

DIVERT: A Distributed Vehicular Traffic Re-routing System for Congestion Avoidance

Juan (Susan) Pan, Iulian Sandu Popa, and Cristian Borcea

***Abstract*—Centralized solutions for vehicular traffic re-routing to alleviate congestion suffer from two problems: scalability, as the central server performs intensive computation and communication with the vehicles in real-time; and privacy, as the drivers share their location as well as the origins and destinations of their trips with the server. This article proposes DIVERT, a distributed vehicular re-routing system for congestion avoidance. DIVERT offloads a large part of the re-routing computation at the vehicles, and thus, re-routing becomes practical in real-time. To make collaborative re-routing decisions, the vehicles exchange messages over vehicular ad hoc networks. DIVERT is a hybrid system because it still uses a server and Internet communication to determine an accurate global view of the traffic. In addition, DIVERT balances the user privacy with the re-routing effectiveness. The simulation results demonstrate that, compared with a centralized system, DIVERT increases the user privacy by 92% on average. In terms of average travel time, DIVERT's performance is slightly less than that of the centralized system, but it still achieves substantial gains compared to the no re-routing case. In addition, DIVERT reduces the CPU and network load on the server by 99.99% and 95%, respectively.**

***Index Terms*—proactive driver guidance, vehicular congestion avoidance, distributed traffic re-routing, VANET**

I. Introduction

The problem addressed in this article is how to perform vehicular traffic re-routing for congestion avoidance in a scalable and privacy-preserving way. Previously, we proposed in [31] a centralized vehicular traffic re-routing system for congestion avoidance. The centralized system collects real-time traffic data from vehicles and potentially road-side sensors, and it implements several re-routing strategies to assign a new route to each re-routed vehicle based on actual travel time in the road network. Rather than using simple shortest path algorithms (e.g, Dijkstra), the re-routing strategies use load balancing heuristics to compute the new path for a given vehicle to mitigate the potential congestion and to lower the average travel time for all vehicles. This individualized path is pushed to a driver when signs of congestion are observed on his current path.

However, despite achieving a substantial decrease in the travel time experienced by drivers, centralized solutions such as ours suffer from two problems. First, the central server performs intensive computation (to re-assign vehicles to new paths) and communication with the vehicles (to send the paths and to receive location updates) in real-time. This can make centralized solutions infeasible for large regions with many vehicles. Second, the server requires the real-time locations as well as the origins and destinations of the vehicles to estimate the traffic conditions and provide effective individual re-routing guidance. This leads to major privacy concerns for the drivers and may prevent the adoption of such solutions due to "big brother" fears. As long as vehicles' traces are fully disclosed, user's identity can easily be inferred even if pseudonyms are used [18]. This is due to the fact that location can contain identity information [33]. Moreover, a sequence of location samples will eventually reveal the vehicle's identity [44]. Therefore, it is important to make the system work without disclosing the users' origin-destination (OD) pairs and with the least number of location updates along a user trip.

These requirements suggest a distributed system architecture. However, a fully decentralized architecture is not suitable for a proactive re-routing system. For example, by creating vehicular ad hoc networks (VANETs), the vehicles can exchange information using multi-hop communication, and thus can detect signs of congestion in small regions while preserving their privacy. However, VANETs do not permit vehicles to get an accurate global traffic view of the road network, resulting in wrong or at least sub-optimal re-routing decisions. In addition, in a fully distributed architecture, due to the lack of a coordinator, the vehicles cannot perform synchronized actions at the same time; thus, it is infeasible to make collaborative decisions in real-time.

To tackle all these problems, this article proposes DIVERT, a **di**stributed **ve**hicular **r**e-rou**t**ing system for congestion avoidance, which leverages both cellular Internet and VANET communication. DIVERT is a hybrid system because it still uses a server, reachable over the Internet, to determine an accurate global view of the traffic. The centralized server acts as a coordinator that collects location reports, detects traffic congestion and distributes re-routing notifications (i.e., updated travel times in the road network) to the vehicles. However, the system offloads a large part of the re-routing computation at the vehicles and thus the re-routing process becomes practical in real-time. To take collaborative re-routing decisions, the vehicles situated in the same region exchange messages over VANETs. Also, DIVERT implements a privacy enhancement protocol to protect the users' privacy, where each vehicle detects the road density locally using VANET and anonymously reports data with a certain probability only from roads with high traffic density. When signs of congestion are detected, the server sends the traffic map only to the vehicles that sent the latest updates. Subsequently, these vehicles

J. Pan and C. Borcea are with the Department of Computer Science, New Jersey Institute of Technology, Newark, NJ 07102-1982 USA (e-mail: jp238@njit.edu; borcea@njit.edu). I. Sandu Popa is with the Department of Computer Science, University of Versailles Saint-Quentin-en-Yvelines, Versailles 78000, France, and also with Inria Paris-Rocquencourt, Le Chesnay 78145, France (e-mail: iulian.sandu-popa@uvsq.fr).

disseminate the traffic data received from the server in their region. User privacy is improved since this protocol reduces dramatically the number of vehicle location updates to the server and, thus, the driver exposure and identification risks. Moreover, in this hybrid architecture, the server does not know the OD pairs of the users.

Hence, the main contribution of this article is the distributed system for re-routing. This system, DIVERT, has four main features: (1) a scalable system architecture for distributed re-routing, (2) distributed re-routing algorithms that use VANETs to cooperatively compute an individual alternative path for each vehicle. (3) privacy-aware re-routing that significantly decreases sensitive location data exposure of the vehicles, and (4) optimizations to reduce the VANET overhead and thus improve vehicle-to-vehicle communication latency.

We measured the effectiveness and efficiency of DIVERT through extensive simulations over two real medium-size urban road networks. The experimental results show that, in comparison with the centralized system, DIVERT can decrease the privacy exposure by 92% in addition to not revealing the OD pairs of the user trips. In terms of average travel time, DIVERT's performance is slightly less than that of the centralized system, but it still achieves substantial gains compared to the no re-routing case. DIVERT is more scalable since it offloads most of the computation burden to the vehicles and reduces the network load on the server by 95%.

The rest of this article is organized as follows. Section II summarizes the related work. Section III explains the design principles of DIVERT. Section IV introduces the privacy enhancement mechanism and the privacy metrics. Section V introduces the two distributed re-routing strategies. The four VANET optimization techniques are presented in Sections VI. The results and associated analysis are discussed in Section VII. We conclude in Section VIII.

II. Related Work

This section discusses aspects related to traffic re-routing, traffic information sharing in vehicular networks, and privacy preserving in location-based services.

A. Traffic Guidance and Re-routing Systems

Services such as INRIX [1] provide real-time traffic information at a certain temporal accuracy, which allows drivers to choose alternative routes with lower travel times. Google Maps and Microsoft's Bing are able to forecast congestion and its duration by performing advanced statistical predictive analysis of traffic patterns. Various mobile navigation applications [2], [3], [4], [5] use such traffic information to transform smart phones into navigation devices. However, these services share the same problem: when congestion happens, they provide the same path for the affected vehicles which potentially generates another local congestion.

The above problem can be solved by dynamic traffic assignment (DTA) which assigns each driver either system-optimal or user-optimal routes [11]. However, DTA algorithms may not be able to compute the equilibrium fast enough to inform the vehicles about their new routes in time to avoid congestion. DIVERT, on the other hand, is designed to be effective and fast, although not optimal, in deciding which vehicles should be re-routed when signs of congestion occur as well as computing alternative routes for these vehicles.

Several other projects have also aimed to provide near-optimal routes to drivers but better scalability compared to DTA. In [26], the first k shortest paths from source to destination are calculated, and then the system determines which path each vehicle should take by minimizing a Lyapunov-style cost function. In [8], the authors proposed a genetic algorithm method to compute the alternative paths and assign them to cars under the assumption that the traffic is known a priori. This method computes the assignment only once as opposed to traditional iterative assignments methods. Our previous work [31] and the research in [43] emphasize that the previous route planning decisions should be considered when determining the next route. Themis [27] uses a similar approach, but it computes the re-routing alternatives based on real-time speed, predicted travel time, and anticipated traffic volume.

DIVERT differs from the above research in three aspects. First, we take full advantage of both cellular and VANET communication to perform scalable re-routing. Thus, each vehicle can get accurate global knowledge of the travel time and, at the same time, is able to exchange route planning decisions with surrounding vehicles more efficiently. Second, the route computation is performed in a distributed way over VANETs. Therefore, it is more scalable since it reduces the computation burden of the central server. Third, we designed and evaluated a privacy enhancement mechanism, where each vehicle only uploads its location report when located in low sensitivity areas.

The work in [25] proposes two urban traffic prediction models that can be used in conjunction with DIVERT. These models could improve the accuracy of our congestion estimation, especially when DIVERT tends to overestimate congestion.

B. Traffic Information Sharing in Vehicular Networks

VANETs enable traffic information sharing for intelligent transportation systems. To improve dissemination efficiency, Gao et al. [14] proposed an adaptive query evaluation plan by taking into account the road topology. Also, Loulloudes et al. presented in [28] V-Radar, an efficient protocol for traffic information retrieval using V2V communications. Several works have sought inspiration in Biology and Internet protocols communication [37], [34]. However, since they employ vehicle ad hoc networking, the above approaches have only a partial view of the traffic conditions, which may lead to less accurate re-routing. Also, simply treating vehicles as packets which always listen to the guidance ignores the nature of human behavior. Furthermore, these systems react to real-time data without insight into future conditions, thus introducing greater vulnerability to switching congestion from one spot to another.

C. Privacy Preserving in Location Based Services

A large body of works consider the problem of preserving the user's privacy in the context of location based services

(LBSs). For instance, the middle layer of DSRC defines the security services for application and message management [24]. Authentication schemes are designed to preserve the driver privacy in DSRC-based VANETs [35]. To prevent malicious tracking, a vehicle could change its anonymous key within an interval of a few minutes [42]. DIVERT has a different goal from all these works: it focuses on protecting the driver's location privacy from the central server, not from the other drivers in VANET. For driver-to-driver privacy, DIVERT can leverage the existing solutions.

SCMS [41] provides privacy protection from both outsiders (i.e., other vehicles or eavesdroppers) and insiders (i.e., administrators of the servers). DIVERT is complementary to SCMS, as its goal is to minimize the amount of privacy sensitive information uploaded to the server (i.e., location and OD pairs), not to protect the information privacy once it has been uploaded. Furthermore, SCMS relies on the organizational separation assumption to protect against insider attacks. DIVERT, on the other hand, achieves a good level of location privacy protection even if this assumption does not hold.

Many works focus on spatial cloaking [17], [44] to provide k-anonymity. The work in [16] argues that both spatial and temporal dimensions should be considered in the algorithm to achieve better k-anonymity. Fundamentally, k-anonymity reduces the quality of the user's localization, which is not applicable for continuous location based services such as real-time vehicle re-routing.

A number of mechanisms provide solutions for highly accurate real-time location updates, while achieving good privacy protection [29], [21]. However, these mechanisms require a trusted centralized entity such as a proxy server for location reporting. Our privacy aware mechanism works in a distributed and probabilistic fashion without any help from trusted entities. Thus, the risk of location tracking is distributed over VANETs, and we argue that this is qualitatively better than trusting a single central entity.

III. System Overview

A. Design Principles

DIVERT is built around two design principles corresponding to the two major requirements described in Section I. First, the re-routing path computation should be offloaded from the central server to the vehicles to reduce the computation and communication burden on the server and achieve better scalability. Therefore, the alternative routes should be computed by vehicles when there are signs of congestion on the roads. At the same time, the re-routing computation should be collaborative in order to achieve a better re-routing effectiveness. To this end, the vehicles could exchange messages over VANETs and implement a distributed re-routing process.

Second, DIVERT should be designed to respect the privacy of the users from its conception, i.e., a privacy-by-design system, which can be essential for the wide acceptance of the system. Implicitly, by offloading the path computation to the vehicles, the drivers' exposure is reduced significantly since very sensitive location information (i.e., the OD pairs) is not sent to the server anymore. Nevertheless, protecting only the OD of a vehicle is not sufficient. DIVERT needs a mechanism to protect the identity of vehicles while reporting location data.

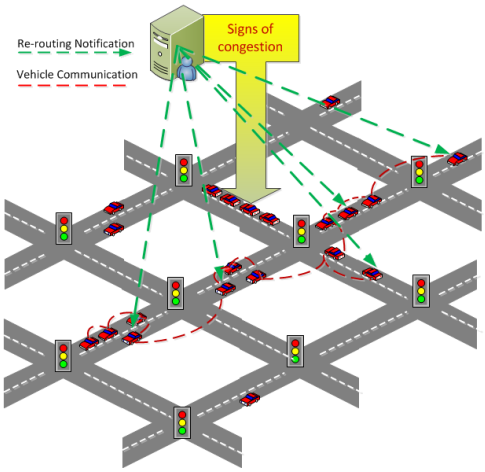


Fig. 1: DIVERT's hybrid architecture

B. System Architecture

Given the described design principles, a hybrid architecture is proposed to implement DIVERT as shown in Figure 1. The architecture is composed of a central server and a software stack running on an on-board device (e.g., a smart phone) in each participating vehicle. DIVERT uses two types of communication. The vehicles communicate with the server over a 3/4G network to report local traffic density data and to receive the global traffic density in the road network (i.e., the green lines in Figure 1). The vehicles report data according to a privacy-aware algorithm that is detailed in Section IV. Also, the vehicles that are closely located communicate with each other over VANETs to determine the local traffic density, to disseminate the traffic data received from the server, and to implement a distributed re-routing strategy (i.e., the red lines in Figure 1) as detailed in Section V.

The server uses the vehicle traffic reports to build an accurate and global view of the road network traffic. The network is represented as a directed graph where each edge corresponds to a road segment. In addition, each edge has associated a dynamic weight representing the real-time traffic density on the edge. A road segment is considered to exhibit signs of congestion when the traffic density is greater than a threshold value. Each time new road segments exhibit congestion signs, the server sends a partial weighted graph (i.e., only the edges having a travel time different from the free flow travel time) to the cars that reported recently and are close to the congestion segments (see Section IV).

The notified vehicles disseminate the information (i.e., traffic graph and vehicle route) in their regions with a limited number of hops to avoid excessive flooding. The dissemination also has a timeout, which is a constant parameter in the proposed system. When the time is up, based on the traffic graph and route information shared by other vehicles, each vehicle, whose current path traverses the congestion spot, locally computes a new route to its destination. This re-routing process is presented in Section V.

IV. Privacy-Aware Traffic Reporting

DIVERT's goal is to protect driver's location privacy against attackers at the server side, who could link traffic reports (which include locations) to driver identities. The traffic reports need to be frequent to compute a global traffic view and detect congestion accurately. Yet, the location reports, when sent frequently, can create severe privacy leakage [18], [33]. Even if pseudonyms are used for location reports and are changed frequently, attackers at the server side can use background information to identify the user identity for certain location points (home, work, etc.) and then use prediction algorithms to identify the whole location trace [18]. Therefore, DIVERT strives to minimize the driver's privacy leakage by reducing the amount of location reports uploaded at the server, while maintaining good traffic accuracy.

To this end, we introduce first a privacy metric in Section IV-A. Then, we propose in Section IV-B a privacy-aware traffic reporting mechanism based on the road traffic density to reduce the privacy leakage for the reporting vehicles. Our system considers that the vehicles are trusted. Indeed, to avoid congestion, vehicles make collaborative decisions which require a vehicle to share shortest path or OD pair information with other nearby vehicles. While privacy enforcement at the vehicle side is out of the scope of this paper, it is worth mentioning that several recent works consider the problem of shared data protection by leveraging the Trusted Execution Environment of (personal) secure devices. Be it the secure TrustZone CPU [6] of the ARM cortex-A series equipping most of mobile devices today, a tamper-resistant hardware security module securing the on-board computer of a vehicle [13], a secure portable token [38] communicating with the user smart phone or plugged inside it (e.g., Google Vault), all such secure devices offer tangible, hardware-based security guarantees. Such technologies can be leveraged for secure data processing in a distributed architecture as DIVERT to protect the shared information and prevent malicious vehicles obtaining access to unauthorized data [39]. Instead of hardware-based security, other techniques such as secure multiparty computing or protocols based on homomorphic encryption could also be considered, but they may impact the real-time constraints of DIVERT. Finally, in Section IV-C, we present the algorithm used by the central server to compute the travel time in the road network.

A. Privacy Metric

In order to measure the privacy loss due to each location update, each location report is associated with a weight. Similar to [44], the weight of a location report depends on the popularity of the location road segment. That is, the more popular a spatial region is, the more difficult it is for an adversary to single out the report sender. However, the number of vehicles along the segment is not sufficient to quantify its popularity, because some vehicles may have a dominant presence in that space. Instead, a metric is applied that is based on the entropy of the road segment.

Definition - Road segment popularity. *Let rs be a road segment and $S(rs) = v_1, v_2, ..., v_m$ be the set of vehicles that send location updates in rs. Let n_i ($1 \leq i \leq m$) be the number of location updates that vehicle v_i sent from rs and $N = \sum_{i=1}^{m} n_i$. Then the entropy of the road segment rs is $E(rs) = -\sum_{i=1}^{m} \frac{n_i}{N} \log \frac{n_i}{N}$ and the popularity of rs is $P(rs) = 2^{E(rs)}$.*

Definition - Privacy leakage. *Given the above defined popularity measure of a location, the global privacy leakage for vehicle v_i is defined as:*

$$pleak_{v_i} = \sum_{all\ updates} \frac{update_i}{P(location\ of\ update_i)}. \quad (1)$$

Definition - Average privacy leakage. *The average privacy leakage for all the vehicles is:*

$$pleak_{avg} = \frac{\sum_{all\ vehicles} pleak_{v_i}}{total\ number\ of\ vehicles} \quad (2)$$

B. Density Reporting

Our privacy-aware reporting is based on the observation that in dense areas, vehicles naturally experience a higher degree of anonymity similar to a person walking through an inner-city crowd. Therefore, a density-based traffic reporting mechanism is proposed wherein vehicles report to the server only if the road density is higher than a predefined threshold. The server computes the smoothed average of the traffic density on each road segment as it receives new traffic reports. Computing the smoothed average of the traffic density at each vehicle (using a moving time window) is of little use in our case because the vehicles do not report often due to our privacy-aware reporting protocol (e.g., a vehicle rarely reports twice from the same road segment). This mechanism is beneficial for both the re-routing effectiveness and the vehicle privacy, since the server can still accurately detect the congestion signs at the cost of lower user privacy exposure.

Our goal is to minimize the number of vehicle reports, i.e., only a fraction of the vehicles situated on a road segment will send traffic reports. Specifically, density reports sent to the server conform to the following rules: (1) cars submit reports only when they perceive that the density on the road segments is above a threshold that would signal a chance of congestion, (2) cars decide probabilistically when to submit data as function of the density - i.e., the more cars there are, the fewer reports each car submits as the reports are distributed among the cars on the segment, (3) cars send their messages through anonymizers (e.g., Tor [12]) to protect their identities.

The estimated density is computed locally by each vehicle, which obtains information about its neighbor vehicles by periodic exchange of beacons. Each vehicle counts the received beacons in a short time window (i.e., 5 seconds in our implementation) and each vehicle emits beacons with the same frequency (such that each vehicle is counted exactly once in each period). As depicted in Algorithm 1, each vehicle periodically checks the number of vehicles N_i on the current road r_i. To obtain accurate traffic information, each vehicle encapsulates in the beacon the current road identifier (i.e., r_i) and direction of traffic (i.e., $side$). When a vehicle estimates

Algorithm 1 Privacy-aware Density Reporting

1: **procedure** onDenistyCheckTimeout()
2: N_i = getEstimatedNumberofCars(r_i, $side$, $\{beacon_k\}$)
3: **if** $N_i \geq max[\theta * N_{max},\ bound]$ **then**
4: $p = 1/N_i$
5: **if** rand $< p$ **then**
6: sendtoTMC(N_i, r_i, $side$)
7: **end if**
8: **end if**

the number of cars (line 2), it only counts the beacons with the same r_i and $side$ as itself.

The vehicle reports the detected density to the central server with probability $p = 1/N_i$ only if $N_i \geq max[\theta * N_{max}, bound]$ (lines 3-4). N_{max} is the maximum number of vehicles that can occupy road r_i, and θ is a system parameter threshold. $N_{max} = L_i * Lane_i/veh\_len$, where L_i and $Lane_i$ are the length and the number of lanes of road r_i. The parameter veh_len is the sum of the average vehicle length ($\approx 5meters$) and the minimal gap between vehicles ($\approx 2.5meters$). The parameter $bound$ specifies that, when reporting, there must be at least $bound$ cars on the road. Practically, privacy is protected by N_{max} for longer roads with multiple lanes and by $bound$ for short roads with potentially few lanes (i.e., $bound$ gives a minimal privacy guarantee). Additionally, after a car reports, it stops reporting for a time period (e.g., 5 seconds in our prototype) to prevent frequent reporting that could lead to privacy leakage.

If every vehicle applies the same reporting procedure, then the probability for the server to receive x location reports from r_i is $\binom{N_i}{x}(1-p)^{N_i-x}p^x$. The expected number of updates on road r_i is $nu_i = \sum_{x=0}^{N_i} x\binom{N_i}{x}(1-p)^{N_i-x}p^x$.

A potential problem of the above algorithm is that it does not consider the case in which the vehicle transmission range is shorter than the road segment length. First of all, this is a very rare case since the typical transmission range is 500 meters, which can cover most urban road segments (e.g., the average road segment length is less than 300 meters for the two real road networks used in our experiments). Second, if indeed a segment is longer than the transmission range, the reported density is inaccurate especially when the traffic density along the segment is extremely skewed (e.g., congestion starts to form at one end of the segment, while the other end still has light traffic). In this case, our algorithm overestimates the density and may trigger re-routing. However, this is not necessarily an issue because if nothing is done, the congestion is expected to increase anyway when the incoming traffic on the segment is heavy.

C. Report Collection and Travel Time Computation

The server receives reports from vehicles indicating the number of vehicles on a road segment and computes the traffic density on the roads. Every time the server receives a report concerning a road r_i, it will smooth the computed density value K_i using the following formula: $K_i^{current} = \alpha * K_i^{old} + (1-\alpha) * K_i^{new}$. The value of α is experimentally chosen to be 0.05. The server then estimates the travel time of each road segment by considering the following three cases.

Case 1: For the road segments without any traffic reports, the server estimates the travel time to be the free flow travel time. Note that this travel time is an approximate value for some roads, as vehicles only report when the vehicle density is above the threshold density.

Case 2: For the road segments with a non-zero traffic density but for which the last report time is older than a predefined time interval, the server does an expiration operation. Specifically, if the difference between the last update time and the current time is greater than τ times the free flow travel time of the road, then the road density is reset to zero. The value for τ is also based on empirical results and is chosen to be equal to 4 in this system.

Case 3: For the road segments that do not fit in Cases 1 and 2, the server uses the Greenshield model [7] to estimate the travel time according to the speed-density-volume relation. This model is used extensively by transportation researchers and was shown empirically to describe well the speed-density relation for relatively low densities. The model considers a linear relationship between the estimated road speed V_i and the traffic density K_i (vehicles per meter) on road segment r_i:

$$V_i = V_f(1 - \frac{K_i}{K_{jam}}) \qquad T_i = L_i/V_i \tag{3}$$

where K_{jam} and V_f are the traffic jam density and the free flow speed for r_i, while T_i and L_i are the estimated travel time and length for the same segment. The free flow speed V_f is defined as the average speed at which a motorist would travel if there were no congestion or other adverse conditions. To simplify our implementation, we consider that the free flow speed is the speed limit, and the traffic jam density is when the road is fully occupied by cars. In this case, K_i/K_{jam} equals N_i/N_{max}, where N_{max} is the maximum number of vehicles on the road segment and N_i is the current number of vehicles obtained from the traffic data collected by the system.

Note that, because of the density-based reporting policy, the traffic density may not be fully accurate. However, the higher the traffic density of a road is, the more accurately the traffic density will be estimated. This is important since the re-routing effectiveness mainly depends on the traffic accuracy of the dense traffic roads. In Section VII, we show that the loss of accuracy in the traffic view has only a marginal effect on the re-routing effectiveness, but greatly improves the privacy protection of the users.

V. Distributed Re-routing Strategies

If the server detects signs of congestion in the road network, it will alert the vehicles by sending the updated map information, i.e., the "updatedMap" parameter in Algorithm 2 containing tuples ($road\ id$, $new\ computed\ travel\ time$) for all the roads that have a current travel time different from the free flow travel time. The server sends messages only to the vehicles that reported most recently and that are located near a congestion spot, i.e., no further than three road segments. The server notification triggers the re-routing process that consists of a dissemination phase and a route computation phase. The dissemination phase has two sub-phases as presented in Algorithm 2. When a vehicle receives such a notification message either directly from the server or from the surrounding vehicles, it executes the procedure

Algorithm 2 Vehicle Receives a Congestion Notification

```
1: procedure onCongestionNotification(updatedMap)
2: updateTravelTime(updatedMap) {update the travel time of the map}
3: T←computeBroadcastTime(this.rank) {compute when to start the broadcast based on this vehicle's rank}
4: broadcastUpdatedMap(T) {broadcast the updated travel time map}
5: if this.currentPath intersects congestionSpots then
6:     if dEBkSP then
7:         computekShortestPaths(this, k) {compute the k shortest path for itself}
8:         wait(T_mapBroadcast) {wait until the map broadcast phase finishes}
9:         broadcastkShortestPaths(T) {broadcast the k shortest paths at time T}
10:    end if
11:    if dAR* then
12:        getODPair(this) {get the OD pair for itself}
13:        wait(T_mapBroadcast) {wait until the map broadcast phase finishes}
14:        broadcastODPair(T) {broadcast OD pair at time T}
15:    end if
16: end if
```

Algorithm 3 When Vehicle Receives the Broadcast

```
1: procedure onReceived(vehiclemsg)
2: v = unpack(vehiclemsg) {unpack the message and extract the vehicle data, e.g., rank, k paths or OD pair}
3: receiveddata.push(v) {put the vehicle data into the priority queue based on the rank}
```

described in Algorithm 2. The first part of the procedure (lines 2-4) consists of disseminating the updated travel times to other vehicles. In the second part, the vehicles that received the notification broadcast personal route information to the other vehicles. The route information depends on the re-routing strategy employed by DIVERT, i.e., either the k-shortest paths or the OD pair of the vehicle (lines 6-10 and 11-15).

On receiving a route information message, the vehicles store the received data as indicated in Algorithm 3. The received data will be used in the route computation phase to compute a new best path for the current vehicle. Note that all the notified vehicles participate in the updated map data dissemination, but only the vehicles whose current paths traverse a congestion spot execute the computation phase (line 5). We only re-route vehicles that are directly impacted by congestion since this is sufficient to alleviate congestion and improve the travel times for all vehicles. Moreover, this approach reduces the re-routing frequency for a driver and thus the computation and communication overhead [31].

In the remainder of this section, we present an overview of our two centralized re-routing strategies that have proven to be the most effective in alleviating congestion, and then describe their distributed counterparts used in DIVERT.

A. Overview of Centralized Re-routing

The first centralized re-routing strategy is the Entropy Based k Shortest Paths (EBkSP). The server first ranks the vehicles to be re-routed in the increasing order of their remaining travel time to destinations. Then, it computes k alternative shortest paths for each vehicle. The server sequentially goes through the ranked list of vehicles and assigns to each vehicle the best path out of the k computed paths. The best path for a vehicle is considered to be the least popular path among its k-shortest paths in order to avoid potential future congestion. To compute the least popular path, a weighted footprint counter as defined below is attached to each road segment.

Definition - Weighted footprint counter. *A weighted footprint counter, fc_i, of a road segment r_i is defined as follows: $fc_i = n_i \times \omega_i$, where n_i is the total number of vehicles that are assigned to paths that include segment r_i, and ω_i is a weight associated with r_i. $\omega_i = \frac{len_{avg}}{len_i \times lane_i} \times \frac{Vf_{avg}}{Vf_i}$, where len_{avg} is the average road segment length in the network, Vf_{avg} is the average free flow speed of the network, len_i is the length of r_i, Vf_i is the free flow speed of r_i, and $lane_i$ is number of lanes of r_i.*

The first vehicle is assigned the current best path without considering others. Then, the footprints counters are updated based on the new path. When assigning the second vehicle, the popularity scores of its k-shortest paths are calculated as defined below, and the least popular path will be chosen. The process is then repeated for the rest of the re-routed vehicles.

Definition - Popularity of a path. *Let $(p_1, ..., p_k)$ be the set of paths computed for the vehicle which will be assigned next. Let $(r_1, ..., r_n)$ be the union of all segments of $(p_1, ..., p_k)$, and let $(fc_1, ..., fc_n)$ be the set of weighted footprint counters associated with these segments. The popularity of p_j is defined as $Pop(p_j) = e^{E(p_j)}$. $E(p_j)$ is the weighted entropy of p_j and is computed as $E(p_j) = -\sum_{r_i \in p_j} \frac{fc_i}{N} \ln \frac{fc_i}{N}$, $N = \sum_{r_i \in p_j} fc_i$.*

The second centralized strategy is the A* with Repulsion (AR*) algorithm. AR* modifies the classical A* algorithm to incorporate the other vehicle paths into the path computation of the current vehicle as a repulsive force. As with EBkSP, the server ranks the vehicles to be re-routed and computes the alternative path for each vehicle in this order. The classical A* [20] uses a best-first search and a heuristic function to determine in which order to visit the network nodes (crossroads in our case). Given a node x, a heuristic function $F(x)$ is computed as the sum $G(x)+H(x)$. $G(x)$ is the path-cost from the start node to x, which corresponds to the travel time in our case, while $H(x)$ is a heuristic estimation of the remaining travel time from x to the destination node. Typically, $H(x)$ is computed as the Euclidean distance divided by the maximum speed on the roads. In AR*, we modified the heuristic function $F(x)$ to include the other vehicles sharing the same path as a repulsive force. Specifically, we define the repulsive score $R(x)$ of a node x as the sum of the weighted footprint counters from the starting node to the node x. Thus, the path-cost function becomes $F(x) = (1-\beta) \times (G(x)+H(x)) + \beta \times R(x)$, where $G(x)$ and $H(x)$ are computed as in the original algorithm and β is a weighting parameter. $G(x) + H(x)$ measures the travel time factor, while $R(x)$ reflects the impact of other vehicle traces on the examined path. Since the travel time and the repulsive force use different metrics, we normalize their values and compute $F(x)$ as a linear combination of the two factors. The parameter β allows a variable weighting between the travel time and the repulsive force, and its value is determined empirically to achive the best AR* effectiveness.

B. Distributed Re-routing

EBkSP and AR* greatly improve the vehicles' travel time. However, these strategies are designed for a centralized architecture in which all the re-routing computation is done at the

server side. Our objective is twofold. First, we want to provide distributed re-routing strategies in DIVERT that are based on the same ideas as the effective centralized strategies. This is challenging, since the computation can only be done by the vehicles in order to comply with the system design principles. Second, the distributed re-routing should ideally have similar effectiveness as the centralized re-routing. In the following, we present dEBkSP (distributed EBkSP) and dAR* (distributed AR*), two distributed re-routing strategies that achieve these objectives.

Both dEBkSP and dAR* require the re-routed vehicles to be ranked. In the centralized version, the rank of each vehicle is assigned by the server. In DIVERT, each vehicle picks a rank value that is randomly selected based on the estimated total number of re-routed vehicles, $rank_{max}$. This number is calculated by each re-routed vehicle as the total number of vehicles situated on the incoming road segments no further than L segments (e.g., L is 3 in our experiments) from the congestion point. A vehicle of a certain rank computes a new route by considering the higher ranked vehicle paths. In dEBkSP, each vehicle affected by congestion calculates the k loop-less shortest paths based on its current OD pair and the updated travel times on the roads. Then, the vehicles disseminate their rank and k shortest paths in their region for a predefined time interval (see Section VI). At the end of the route dissemination phase, each vehicle receives the k shortest paths of a certain number of vehicles in the region. Given the nature of the dissemination process, the information gathered by a vehicle can be incomplete and different from one vehicle to another. In the final route computation phase, each vehicle iterates through the local sorted list of vehicles. It selects the (potentially) best path based on the original EBkSP algorithm for each received vehicle with a higher rank and eventually selects the best path for itself.

Similarly, in the case of dAR*, the notified vehicle chooses a random rank but it does not compute the k shortest paths. Instead, the notified vehicles only broadcast their OD pairs. In the event of a broadcast timeout, for each received OD pair in the buffer, the vehicle applies the original AR* algorithm to compute a virtual path. The current vehicle assumes that the vehicle with that OD pair will take that virtual path. By the end of the process, the current vehicle computes the best shortest path for itself based on other vehicles' paths.

Algorithm 4 describes how dEBkSP and dAR* are executed when the information dissemination phase ends (i.e., timeout). It is worth noticing that dEBkSP and dAR* have opposite behaviors with regards to the two main phases of the re-routing process. dEBkSP incurs a higher overhead in the communication phase than dAR*. The packets in dEBkSP are much larger than those in dAR*: dEBkSP packets contain the k shortest paths, while dAR* packets contain only the vehicle OD pair. On the other hand, the computation phase is more efficient in dEBkSP than in dAR*. Since the paths are individually computed and disseminated in dEBkSP, a vehicle only has to choose between the k paths for all the vehicles from which it has received re-routing data, which has a very low computation cost. In dAR*, a vehicle has to compute the shortest paths for all the vehicles ranked higher than itself from which it has received re-routing data. This computation difference is explained in detail in Section VII.

Algorithm 4 Distributed EBkSP and AR*

```
1: procedure onBroadcastTimeOut()
2: receiveddata {all the received data}
3: Q = empty {a queue that stores the vehicle objects that have already been
   processed}
4: while v_i != this do
5:     v_i = receiveddata.pop()
6:     if dEBkSP then
7:         getkShortestPath(v_i) {get the k shortest path for this vehicle}
8:         doEBkSP(v_i, Q) {pick a path from the k paths for vehicle v_i based
           on vehicles' paths with higher rank}
9:         Q.push(v_i){label the vehicle v_i as a processed vehicle}
10:    end if
11:    if dAR* then
12:        getODpair(v_i) {get the OD pair for this vehicle}
13:        doAR(v_i, Q) {Compute a A star shortest path with repulsion for
           this vehicle based on vehicles' paths with higher rank }
14:        Q.push(v_i){label the vehicle v_i as a processed vehicle}
15:    end if
16: end while
17: if dEBkSP then
18:    getkShortestPath(this) {get the k shortest path for itself}
19:    doEBkSP(this, Q) {pick a path from the k paths for itself based on
       vehicles' paths with higher rank}
20: end if
21: if dAR* then
22:    getODpair(this) {get the OD pair for itself}
23:    doAR(this, Q) {Compute a A star shortest path with repulsion for
       itself based on vehicles' paths with higher rank }
24: end if
```

VI. VANET Optimizations for Re-routing Information Sharing

The effectiveness of the distributed re-routing strategies depends on the amplitude of the re-routing information dissemination among vehicles. This dissemination has two related dimensions. The first is represented by the total number of vehicles that receive re-routing information in a congested region. The second regards the average volume of information received by the vehicles. The higher the number of receiving vehicles and the higher the amount of information are, the more effective the re-routing process is. Ideally, each vehicle affected by congestion should receive re-routing information about all the vehicles in their region. In this case, the re-routing process can have a similar effectiveness with centralized re-routing. However, achieving this level of dissemination in VANETs is challenging for two reasons. First, the dissemination has to be done in real-time and therefore its time interval is short. Typically, the dissemination phase is limited to 0.2s in the system. Second, regular dissemination in VANETs exhibits poor performance in congested areas because of contention (i.e., many vehicles try to communicate at the same time) [30].

In this section, we present four optimization techniques implemented in DIVERT which are applied together to improve the data dissemination efficiency in VANETs. These techniques are: i) prioritized data dissemination, ii) k-shortest paths compression, iii) XOR coding for packet loss recovery, and iv) distance-based timer for efficient broadcast.

A. Prioritized Data Dissemination

DIVERT uses a prioritized dissemination to avoid that all the notified vehicles in a region start broadcasting at the same time, and thus reduce the network contention. When receiving

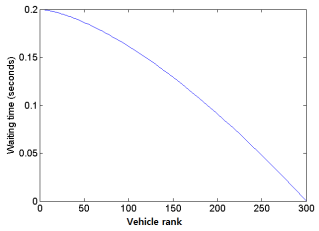


Fig. 2: The ranking function for prioritized data dissemination

a congestion notification, vehicle v_i waits T_i seconds before broadcasting its OD pair or its k-shortest paths. The waiting time is determined based on the rank of the vehicle defined in section V. The rationale is that the higher rank vehicle information is more important since each vehicle computes its own path based on the higher ranked vehicle paths. Therefore, the waiting time function in equation 4 gives the higher ranked vehicles more time to disseminate their path data:

$$t_i = \alpha * rank_i^{\gamma} + T_{max}, \alpha = -T_{max}/rank_{max}^{\gamma} \quad (4)$$

T_{max} is the total dissemination time introduced in Section V, i.e., the time after which everyone stops disseminating and starts computing the new route. $rank_i$ is the rank of vehicle v_i and $rank_{max}$ is the maximum rank of all vehicles. $rank_{max}$ is estimated by each vehicle from the road network density data received at the beginning of the re-routing process. γ is a predefined system parameter that is set to be 1.5 in our implementation. The waiting function has the following properties: i) the waiting time t_i for each vehicle is a value in the interval [0, T_{max}]; ii) the higher ranked vehicles wait less time than the lower ranked vehicles. Specifically, the vehicle with maximum rank transmits without waiting, while the vehicle with lowest rank waits T_{max} time. Figure 2 illustrates the ranking function when T_{max} is 0.2s and $rank_{max}$ is 300.

B. K Path Compression

The dAR* re-routing is very efficient from a communication point of view since vehicles only disseminate their OD pair. However, this is not the case for the dEBkSP strategy that requires vehicles to transmit their k-shortest paths. Hence, depending on the k value and the distance between the origin and destination, the size of the messages can be large. A large packet size increases the communication overhead and decreases the dissemination effectiveness. In addition, the MAC layer protocol limits the payload of a packet that can be sent on the communication channel. Therefore, one way to optimize the information dissemination between vehicles is data compression. Since the k-shortest paths generally present a high degree of overlapping, a compression algorithm is proposed to exploit this feature.

Figure 3(a) shows a simple example of the space savings that could be obtained for three paths between the roads A and J. If the three paths are naively broadcasted, 15 spaces are needed in total. However, the three paths only cover 9 distinct roads and therefore optimally need 9 spaces to be transmitted. To obtain a compact representation of the k paths without any

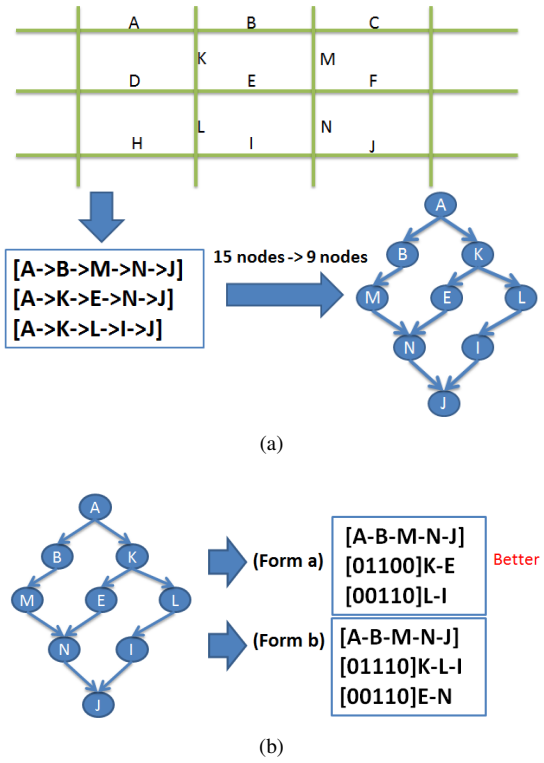


Fig. 3: Example of k-shortest paths compression

loss of information, we represent only the differences between the k paths. Specifically, for path i, only the edges that are different from path $i-1$ are included in the packet. A bit vector is used to represent the position of the edges. As depicted in Figure 3(b), the three paths can be represented either in form (a) or form (b). The '1' in the bit vector of each row indicates that the edge in that position is different compared to the previous path. Obviously, form (a) is better than form (b) since form (a) uses one space less.

The problem comes down to finding the sequence of the k paths resulting in the best compression. However, this problem is reducible to the Hamiltonian path problem and therefore it is NP complete. Hence, a "greedy" algorithm is described to iteratively compress k paths, based on the number of overlapping edges as shown in Algorithm 5. The function $Compress(P, P_k)$ compresses path P with respect to path P_k. The function produces a bitmap of size equal to the number of segments of P, in which bit i corresponds to the segment i in path P. Bit i is set to 0 if the segment i belongs to path P_k and to 1 otherwise. For each bit equal to 1 in the bitmap, the corresponding node id is also generated by the compression function in order to be able to re-compute path P based on P_k and the compressed value of P.

Algorithm 5 K Shortest Path Compression

```
1: procedure compresskpath()
2: k = KPaths.size() {the number of k}
3: P = KPaths[0] {the shortest path}
4: while k>0 do
5:     P_k = FindMostOverlappingPath(P)
6:     P = Compress(P, P_k)
7:     k = k-1
8: end while
```

C. XOR Coding for Packet Loss Recovery

Data dissemination in VANETs can be significantly affected by packet losses. To allow vehicles to recover lost packets, a supplementary XOR coding field is appended to each transmitted packet similar to the work in [23]. When receiving a packet, a vehicle may recover one of the lost packets from the XOR field. This technique helps reducing the number of transmissions and also improves vehicles' knowledge coverage.

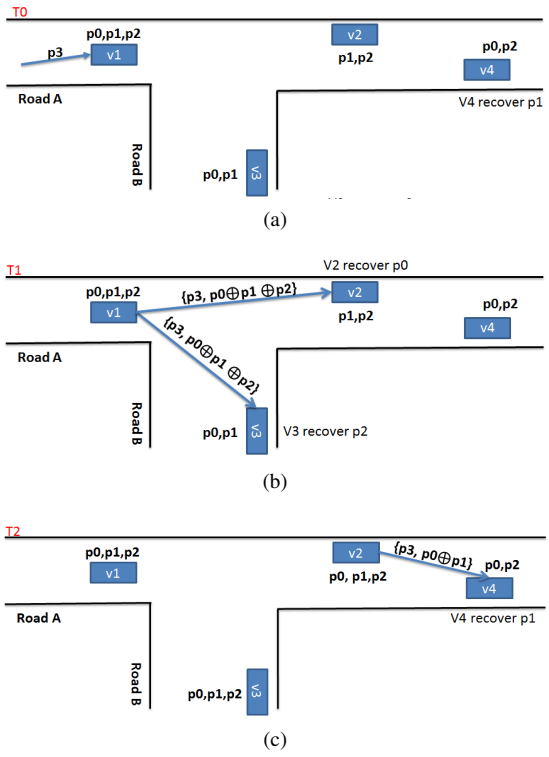


Fig. 4: Message forwarding with XOR coding field

Figure 4 illustrates this concept wherein four vehicles {v1, v2, v3, v4} are present on roads A and B. At T0, packet p3 arrives at v1. Due to losses, the packets that the vehicle has are: {v1: p0, p1, p2}; {v2: p1, p2}; {v3: p0, p1}; and {v4: p0, p2}. At time T1, v1 broadcasts the latest received packet p3 with an appended field p0⊕p1⊕p2. Hence, both v2 and v3 can recover their missing packets p0 and p2, respectively, from p3. Without the XOR coding field, two messages would be required to recover p0 at v2 and p2 at v3. Similarly, at T2, v2 broadcasts p3 with the coding field p0⊕p1, so that v4 can recover p1 as well.

The question is how to figure out which packets should be coded together. For example, if v1 appends the coding field p0⊕p1 instead of p0⊕p1⊕p2, then only v2 can recover p0, but v3 cannot recover p2. Therefore, to find the most efficient coding, each vehicle needs to be aware of the other vehicle's packets. Using channel eavesdropping, vehicles can obtain a certain amount of knowledge of the neighborhood. To further improve this knowledge, a Bloom filter is used for a compact representation of the knowledge of each vehicle. Each time a packet is sent, the forwarding vehicle attaches a Bloom filter encoding the set of vehicles' identifiers it has received so far. This is similar to the distributed caching solution in [10]. When a vehicle A receives a packet from a neighbor B, it obtains the current knowledge of B from the Bloom filter. In this way, each vehicle can build up a neighbor table containing the knowledge of its neighbors. This table is essential for finding the best coding according to COPE [23], and it is employed as described in Algorithm 6. Specifically, to determine the best XOR coding field, each vehicle uses its neighbor table to check if a neighbor can decode P⊕Q (line 6), where P and Q are two vehicle identifiers in the received data buffer.

Algorithm 6 Append XOR Coding Field

```
1: procedure setcodingfield()
2: receiveddata {all the received data}
3: P = receiveddata.pop()
4: For each neighbor i in neighbor table
5: repeat
6:     while Q = receiveddata.pop() do
7:         if neighbor i candecode (P xor Q) then
8:             P = P xor Q
9:         end if
10:    end while
11:    i = i+1
12: until i = neighbortable.end
13: End for
```

D. Distance-based Timer for Efficient Broadcast

In DIVERT, a distance-based timer approach is used to reduce excessive broadcasting when multiple vehicles are within communication range. After receiving a broadcast message, the vehicle waits for a certain time period until re-broadcasting the message. The waiting time period is inversely proportional to the distance between the receiving vehicle and the source vehicle. Therefore, a vehicle that is farther from the message source should re-broadcast the message earlier. During the waiting period, if the current vehicle receives copies of the message, it means that another vehicle has already forwarded the message. Thus, the current vehicle drops the message.

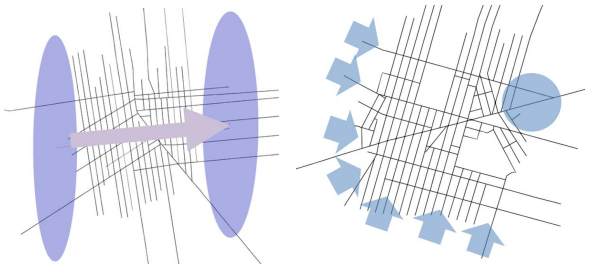
VII. Experimental Evaluation

The main objective of our simulation-based evaluation is to study the performance of the distributed re-routing strategies in DIVERT. Specifically, the evaluation has four goals:

- Assess the effectiveness and efficiency of DIVERT compared to the centralized system.
- Investigate the performance difference between DIVERT with and without privacy-aware traffic reporting.
- Quantify the strength of the privacy protection mechanism.
- Understand which VANET optimizations provide the most benefits.
- Compare the CPU and network load at the server between DIVERT and the centralized system.

A. Simulation Setup

We use Veins [36], a framework for running vehicular network simulations, to facilitate our experiments. TraCI [40] is employed to send commands to vehicles to change their routes. We use sections of Brooklyn, NY and of Newark,



(a) Brooklyn (b) Newark

Fig. 5: Traffic flow in Brooklyn and Newark road networks

NJ for the road networks, which were downloaded in osm format from OpenStreetMap [19]. We use the Netconvert tool in SUMO [9] to convert the maps into a SUMO usable format, and Trafficmodeler [32] to generate vehicle trips. All roads have the same speed limit (13.9 m/s); some roads have one lane in each direction, while others have just one lane.

Figure 5 shows the traffic flows. We used Trafficmodeler to generate a total of 1000 cars from the left area to the right area in Brooklyn, while in Newark we generate bidirectional traffic with 883 cars (775 go in the congestion direction). The number of cars is chosen in such a way as to create heavy traffic and congestion (i.e., the travel time for a majority of cars is much higher than the free flow travel time). We generate the traffic at constant rate by deploying one car each second in the simulator. For Brooklyn, the origins and the destinations are randomly picked from the left area and the right area, respectively. For Newark, the origins are from the left and bottom areas of the map, while the destinations are in the region at the top right corner. The statistics of the road networks are shown in Table I. The shortest travel time paths are automatically calculated and assigned to each vehicle at the beginning of simulation based on the speed limits. For each scenario, we average the results over 10 runs, which is sufficient considering that the variation of the results between the runs is not significant. Tables II and III show the parameters used in re-routing and in the Veins/Omnet++, respectively.

Most of the re-routing parameters of the algorithms are set as the default values of the centralized system [31] and offer the best trade-off between reducing the average travel time and system scalability in terms of computation time. We use the same values for DIVERT in order to be fair to the centralized version. However, since DIVERT offloads the computation to individual cars, it can afford higher re-routing frequency or lower congestion threshold, and thus it is expected to obtain better results. Also, having less frequent re-routing periods reduces the number of re-routings and thus helps with the overall usability of the system (as drivers do not want to be re-routed too often). The interested reader can refer to [22] for a detailed study of the impact of the re-routing parameter values on the re-routing efficiency and effectiveness.

B. Results and Analysis

Average travel time. Figure 6 shows the average travel time for DIVERT compared to the centralized system. DIVERT

TABLE I: Statistics of Brooklyn and Newark Networks

	Brooklyn	Newark
Network area	$75.85km^2$	$24.82km^2$
Total number of road segments	551	578
Total length of road segments	$155.55km$	$111.41km$
Total number of intersections	192	195

TABLE II: Parameters in Distributed Re-routing Algorithms

Period	The frequency of triggering the re-routing; by default period=450s
Threshold δ	Congestion threshold; if $K_i/K_{jam} > \delta$, the road segment is considered congested; by default $\delta = 0.7$
Level L	Network depth to select vehicles for re-routing starting from the congested segment; by default $L = 3$
# Paths k	The max number of alternative paths for each vehicle; by default $k = 8$
Repulsion weight β	The weight of repulsion in dAR*; by default $\beta = 0.05$
Broadcast timeout T_{max}	The maximum duration of the broadcast of the trip data, by default $T_{max} = 0.2$s
Privacy threshold θ	The privacy threshold; if $K_i/K_{jam} > \theta$, the vehicle starts to report the road density in the privacy enhancement component; by default $\theta = 0.5$

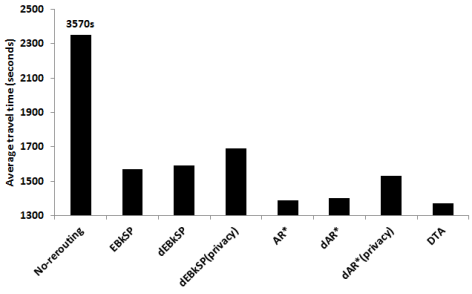
achieves similar travel time as the centralized system for both dEBkSP and dAR*, and both distributed strategies improve the travel time by more than 200% and 300% compared to the no re-routing case on Brooklyn and Newark, respectively. Specifically, the travel time for dEBkSP without privacy-aware traffic reporting is only 1.4% less on Brooklyn and 8% less on Newark than the time for EBkSP. When privacy-aware reporting is used, the performance decrease is 6.6% on Brooklyn and 25% on Newark. Similar results are obtained for dAR*, with performance decrease of 0.9% and 13% without privacy and 10.2% and 24% with privacy, on Brooklyn and Newark respectively.

Let us note that the decrease in performance observed when comparing the distributed strategies without privacy-aware reporting with the centralized strategies is due to lost packets in VANET during the traffic assignment phase of the strategies. The number of lost packets is small because our VANET dissemination optimizations (see Section VI) reduce significantly the effect of network contention. Therefore, the decrease in performance due to missing global information is very small (at most 8%) when compared to the increase in performance obtained by these strategies w.r.t. the no re-routing case (200%-300%).

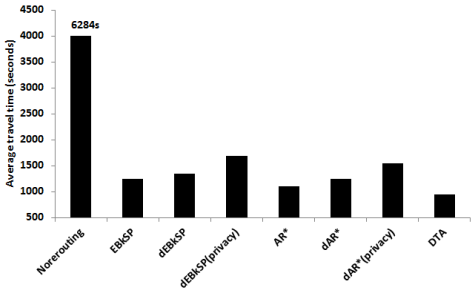
Three lessons are learned from the results: (1) Each vehicle only needs to know the trip information of its neighboring vehicles to make re-routing work effectively. Global information about all the vehicle routes brings minimal benefits. (2) The re-routing with privacy-aware reporting is less effective because the server may misestimate the travel time on certain road segments. However, the benefits of providing higher privacy overcome this performance loss. (3) dAR* achieves better performance than dEBkSP. This is due to the tiny packet size of dAR*, which leads to less contention and thus fewer packet losses. However, dEBkSP is more robust than dAR* to privacy-aware reporting and packet losses.

TABLE III: Simulation Parameters

Channel frequency	5.890e9 Hz
Propagation model	Two ray
Fading model	Jakes' model rayleigh fading
Shadowing model	LogNormal
Antenna model	Omnidirectional
Transmission power	20mW
Propagation distance	500m
Maximum hop	10
PHY model	802.11p
MAC model	EDCA



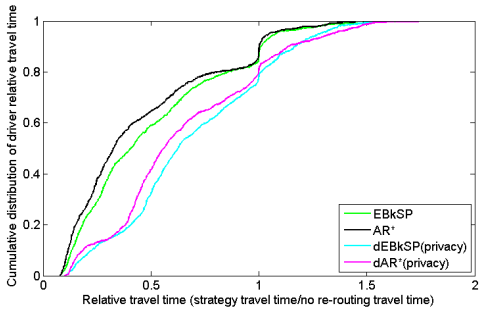
(a) Brooklyn



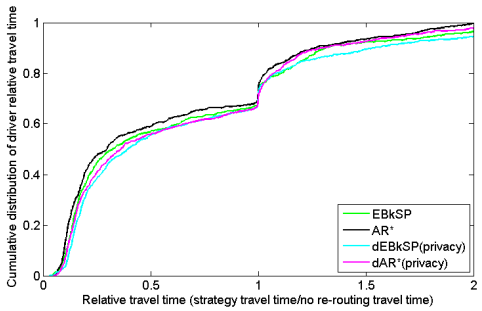
(b) Newark

Fig. 6: Average travel time (T_{max}=0.2s, k=8)

One important question is how good DIVERT's results are when compared to state-of-the-art research in traffic re-routing. Figure 6 also shows the results obtained by a DTA tool [15], which attempts to achieve stochastic user equilibrium through an iterative simulation process. We use 100 iterations, double the number specified in [15]. The higher the number of iterations, the higher the probability to reach a user equilibrium state for the traffic. Despite not being a viable solution for real-time traffic guidance due to its very high computational complexity coupled with high traffic dynamics and imperfect traffic knowledge, DTA is valuable because it gives us a lower bound on the travel time. The experimental results showed that DTA achieves results comparable with our centralized strategies. The experimental results showed that DTA achieves results comparable with our centralized strategies. AR* results in an increased average travel time of 1.4% and 11.6% for Brooklyn and Newark, respectively. EBkSP results in an increased average travel time of 13.1% and 14%, respectively. However, our previous work [31] has shown that EBkSP and AR* are more computationally efficient and scalable than DTA, which makes them more appropriate for use in real-life applications. Since dEBkSP and dAR* achieve travel times just slightly less than EBkSP and AR*, we conclude that their performance is close to the lower bound provided by DTA.



(a) Brooklyn



(b) Newark

Fig. 7: CDF of relative travel time (T_{max}=0.2s, k=8)

Distribution of travel time. The average travel time measures the performance of the system from a global point of view. Here, the performance from a driver point of view is investigated. Relative travel time (RelT) is defined as the actual travel time divided by the travel time without re-routing. It measures the travel time gains or losses for individual drivers. Figure 7 presents the CDF of RelT. It is observed that the system manages to improve the travel time for a large majority of drivers. However, there are a few drivers who experience increased travel time. This increase is limited to less than 50% for most of these drivers on both Brooklyn and Newark networks. From the figure, we notice that dAR* produces better results than dEBkSP (both include privacy-aware reporting). Compared to the centralized versions, both have just slightly worse results.

The explanation for the increase is that our focus is a system-wide optimization (i.e., reduce the average travel time), not user equilibrium which is known to be computationally expensive and difficult to achieve in the presence of congestion. From a practical point of view, a few bad experiences could impact the adoption rate. Therefore, investigating methods to bound this increase to low values is planned as future work. Also, given the similarity of the results and for the sake of brevity, we provide only the results with the Brooklyn network in the remainder of this section.

Distribution of privacy leakage. We use the formula described in Section IV-A to quantify the privacy leakage as shown in Figure 8. These results demonstrate one of the major advantages of DIVERT: it reduces the privacy leakage by up to 92% for both dEBkSP and dAR*. This is due to the fact that privacy-aware reporting avoids submitting location reports from highly sensitive low density roads and submits reports with low per-vehicle frequency in high density

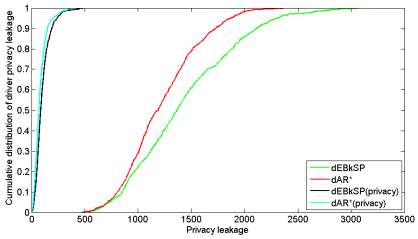


Fig. 8: CDF of privacy leakage

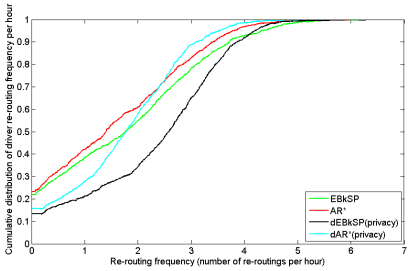


Fig. 9: CDF of re-routing frequency (T$_{max}$=0.2s, k=8)

roads. Therefore, drivers are less prone to be identified by the untrusted server and their location reports are difficult to be linked to each other; thus, driver location privacy is protected. On average, dAR* has less privacy leakage than dEBkSP because it has lower travel time: in dAR*, each vehicle finishes the journey faster, and thus there are fewer location reports.

Re-routings frequency per hour From the system point of view, having a low number of re-routings means decreasing user distraction. Figure 9 shows the CDF of re-routing frequency per hour in the distributed strategies compared to their centralized counterparts. EBkSP and AR* have a relatively low number of re-routings due to the global knowledge of all vehicles' paths. Nevertheless, the distributed strategies only result in 4.5% and 0.4% more re-routings, respectively. Thus, DIVERT proves to be practical from this perspective as well.

Computation cost. In DIVERT, the bottleneck at the server of computing all the alternative paths is removed. If the computation time is high in the centralized system, the drivers would be informed too late about alternative paths, thus defeating the purpose of the system. In DIVERT, each car performs its own path computation using information received from neighboring vehicles. Therefore, this system is expected to have higher scalability. To prove this hypothesis, experiments are performed on a smart phone (the expected computing platform in the vehicles) and the obtained results are plugged into analytical formulas for the two distributed strategies.

The total time consumed for dAR* is the sum of the communication time among vehicles to collect information and the actual local computation time at a vehicle. Since the number of received trip data is a function of T_{max}, the communication time, the total time consumed for dAR* is $T^{total}_{dAR^*} = T_{max} + f(T_{max}) * C(AR^*)$, where $f(T_{max})$ is the number of received re-routing data items and $C(AR^*)$ is the cost of computation to perform AR* on one OD pair.

For dEBkSP, $T^{total}_{dEBkSP} = C(k-paths)+T_{max}+f(T_{max})* C(EBkSP))$, where $C(k-path)$ is the computation cost for

TABLE IV: Average computation time for one OD pair

dijkstra	k shortest paths k=8	AR*
0.244s	0.386s	0.14s

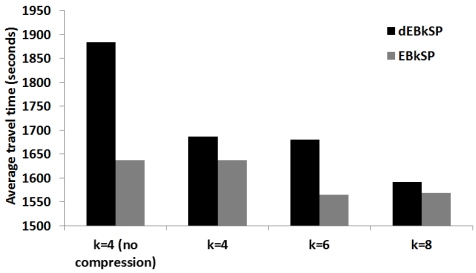


Fig. 10: Average travel time with and without compression for dEBkSP. All the other optimizations are applied in both cases.

k loop-less shortest paths for one OD pair and $C(EBkSP)$ is the cost to select one path from k paths. The complexity of EBkSP only depends on k and the average path length which can be considered negligible. Therefore, dEBkSP has higher computational efficiency than dAR* since the computation time of dAR* is influenced heavily by the number of received re-routing data items whereas the computation time of dEBkSP is basically only $C(k-path)$.

To evaluate the time taken by these strategies on existing mobile platforms, a C++ Android application is developed on Samsung Galaxy SGH-T959V using Android NDK. The average computation time is measured on the initial OD pairs for the 1000 vehicles in these experiments. Table IV shows the computation cost of a single OD pair for each algorithm. The maximum number of received trip data items for dAR*, when T_{max}=0.2s, is 82. Thus, the estimated computation time in the worst case for dAR* is 0.2+82*0.14=12s. The estimated computation time for dEBkSP is 0.386+0.2=0.6s. While both numbers demonstrate that DIVERT works well in practice for this scenario, it is clear that dEBkSP scales better. In larger regions with many vehicles, dAR* may not be able to meet the real time constraints.

Impact of VANET optimizations. During our implementation and testing phase, we noticed that the prioritized broadcast and the distance-based timer approach are essential in our system because without them very little re-routing information is exchanged among the vehicles due to contention in congested regions. Among the remaining two optimizations, i.e., K path compression and XOR coding, the path compression brings the most benefits for the dEBkSP strategy. Figure 10 shows the benefits of compression on this strategy.

We observe that compression improves the average travel time by 12% for k=4. This is due to the fact that compression reduces the packet size and improves the number of re-routing data items each vehicle can gather in VANET. When k increases, dEBkSP continues to lower the travel time. Due to the compression, when k turns from 4 to 8, the addition to the packet size remains small. Therefore, dEBkSP is able to achieve very similar performance as the centralized version.

Let us notice that only dEBkSP can take advantage of larger k values, while the centralized version cannot. A larger k allows for better traffic balancing but introduces higher

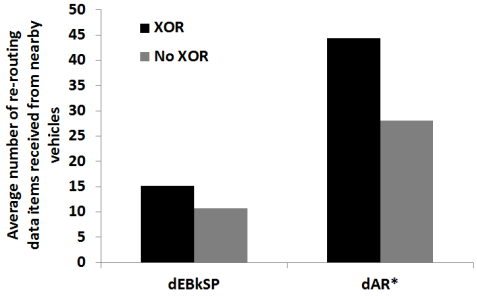


Fig. 11: The average number of received re-routing data items

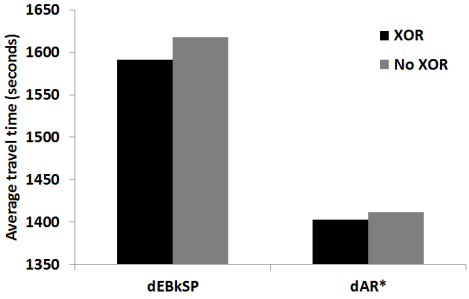


Fig. 12: The average travel time with and without XOR coding

computational complexity since the centralized server needs to compute k paths for all the selected vehicles. This is not a problem for dEBkSP which distributes the path computation to individual vehicles. Therefore, dEBkSP can result in higher performance than EBkSP when higher k values are used.

The XOR coding optimization also plays an important role in improving the data dissemination process so that each vehicle is able to receive more routing information from the nearby vehicles. The results in Figure 11 show that XOR coding increases the average number of received re-routing data items by 41% for dEBkSP and by 57% for dAR*. The benefit is lower for dEBkSP because dEBkSP generates larger packets than dAR*, and appending the XOR coding field increases even more the packet size. However, in spite of this significant improvement in the amount of disseminated data, the gain in the average travel time is only marginal as illustrated in Figure 12. The explanation is that knowing the nearby vehicles' path information is sufficient for dEBkSP and dAR* to provide similar re-routing effectiveness with their centralized counterparts, as already indicated in our first observation under the "Average travel time" results.

Scalability. DIVERT improves the system scalability by reducing the CPU utilization at the server side and reducing the number of messages exchanged between the server and the vehicles.

DIVERT saves server CPU time by offloading the majority of the work to vehicles. The graph computation in our system includes two parts. One is to update the travel time in the road network, equivalent to updating the weights of the graph's edges. The other is to compute the shortest paths in the road network. Updating the graph has an $O(E)$ complexity, while the path computation has an $O(N \log (N + E))$ complexity. We performed an experiment to compare the two parts for our specific settings. Table V shows that the graph weight updating

TABLE V: Ratio between the cost of updateEdgeWeights and OD path computation

EBkSP k=4	EBkSP k=8	AR*
0.002%	0.001%	0.001%

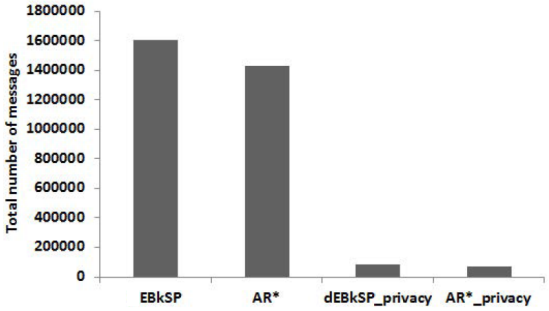


Fig. 13: Total number of messages sent between the server and vehicles

time is approximately 0.001% of the path computation for the centralized versions of EBkSP and AR*. Since our system offloads the path computation to the vehicles and the server is only responsible for the graph weight updating, these results demonstrates the substantial CPU load reduction at the server achieved by DIVERT.

DIVERT also reduces the network load on the server, which could become a major bottleneck as the number of vehicles increases. Since the privacy enhancement protocol only allows vehicles to send traffic reports when the privacy metric meets the probabilistic criterion, the number of messages is decreased by 95%, as indicated in figure 13.

VIII. Conclusions

The article demonstrates that a practical, cost-effective, and efficient traffic re-routing system can be implemented and deployed in real-life settings. This system, DIVERT, offloads a large part of the re-routing computation at the vehicles, and thus, the re-routing process becomes scalable in real-time. To make collaborative re-routing decisions, the vehicles exchange messages over VANETs. We have optimized VANET data dissemination to allow for efficient distributed re-routing computation. In addition, the system balances user privacy with the re-routing effectiveness. The simulation results demonstrate that, compared with a centralized system, DIVERT increases the user privacy substantially, while the re-routing effectiveness is minimally impacted.

References

[1] http://www.inrix.com.
[2] http://www.waze.com/.
[3] http://www.google.com/mobile/.
[4] http://www.tomtom.com/en_gb/products/mobile-navigation/.
[5] http://www.navigon.com/portal/int/index.html.
[6] ARM. *ARM Security Technology - Building a Secure System using TrustZone Technology*. ARM Technical White Paper, 2009.
[7] J.H. Banks. *Introduction to transportation engineering*. McGraw-Hill: New York, 2002.
[8] A. Bazzan, D. Cagara, and B. Scheuermann. An evolutionary approach to traffic assignment. In *Proceedings of the IEEE Symposium on Computational Intelligence in Vehicles and Transportation Systems (CIVTS)*, pages 43–50, 2014.

[9] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz. Sumo - simulation of urban mobility: An overview. In *Proceedings of the 3rd International Conference on Advances in System Simulation(SIMUL 2011)*, pages 63–68, Barcelona, Spain, 2011.

[10] A. Broder and M. Mitzenmacher. Network applications of bloom filters: A survey. *Internet Mathematics*, 1(4):485–509, 2004.

[11] Y.C. Chiu, J. Bottom, M. Mahut, A. Paz, R. Balakrishna, T. Waller, and J. Hicks. Dynamic traffic assignment: A primer. *Transportation Research E-Circular*, (E-C153), 2011.

[12] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. Technical report, Naval Research Lab Washington DC, 2004.

[13] Miad Faezipour, Mehrdad Nourani, Adnan Saeed, and Sateesh Addepalli. Progress and challenges in intelligent vehicle area networks. *Magazine Communications of the ACM*, 55(2):90–100, 2012.

[14] J. Gao, J. Han, D. Yang, and T. Wang. Road network based adaptive query evaluation in vanet. In *Mobile Data Management, 2008. MDM'08. 9th International Conference on*, pages 49–56. IEEE, 2008.

[15] C. Gawron. *Simulation-based traffic assignment–computing user equilibria in large street networks*. PhD thesis, University of Cologne, 1999.

[16] B. Gedik and L. Liu. Protecting location privacy with personalized k-anonymity: Architecture and algorithms. *Mobile Computing, IEEE Transactions on*, 7(1):1–18, 2008.

[17] M. Gruteser and D. Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *Proceedings of the 1st international conference on Mobile systems, applications and services*, pages 31–42. ACM, 2003.

[18] M. Gruteser and B. Hoh. On the anonymity of periodic location samples. In *Security in Pervasive Computing*, pages 179–192. Springer, 2005.

[19] M. Haklay and P. Weber. Openstreetmap: User-generated street maps. *IEEE Pervasive Computing*, 7(4):12–18, 2008.

[20] P.E. Hart, N.J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[21] B. Hoh, M. Gruteser, R. Herring, J. Ban, D. Work, J-C Herrera, A.M. Bayen, M. Annavaram, and Q. Jacobson. Virtual trip lines for distributed privacy-preserving traffic monitoring. In *Proceedings of the 6th international conference on Mobile systems, applications, and services*, pages 15–28. ACM, 2008.

[22] Juan (Susan) Pan. *Vehicle Re-routing Strategies for Congestion Avoidance*. PhD thesis, New Jersey Institute of Technology, 2014.

[23] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft. Xors in the air: practical wireless network coding. In *ACM SIGCOMM Computer Communication Review*, volume 36, pages 243–254. ACM, 2006.

[24] J.B Kenney. Dedicated short-range communications (dsrc) standards in the united states. *Proceedings of the IEEE*, 99(7):1162–1182, 2011.

[25] Z. Liang and Y. Wakahara. Real-time urban traffic amount prediction models for dynamic route guidance systems. *EURASIP Journal onWireless Communications and Networking*, 2014(85):1–13, 2014.

[26] S. Lim. *Congestion-aware traffic routing for large-scale mobile agent systems*. PhD thesis, Massachusetts Institute of Technology, 2012.

[27] R. Liu, H. Liu, D. Kwak, Y. Xiang, C. Borcea, B. Nath, and L. Iftode. Themis: A participatory navigation system for balanced traffic routing. In *Proceedings of the IEEE Vehicular Networking Conference (VNC)*, pages 159–166, 2014.

[28] N. Loulloudes, G. Pallis, and M.D. Dikaiakos. V-radar: A vehicular traffic query protocol for urban environments. In *Vehicular Traffic Management for Smart Cities (VTM), 2012 First International Workshop on*, pages 1–6. IEEE, 2012.

[29] J. Meyerowitz and R. Roy Choudhury. Hiding stars with fireworks: location privacy through camouflage. In *Proceedings of the 15th annual international conference on Mobile computing and networking*, pages 345–356. ACM, 2009.

[30] S.Y. Ni, Y.C. Tseng, Y.S. Chen, and J.P. Sheu. The broadcast storm problem in a mobile ad hoc network. In *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 151–162. ACM, 1999.

[31] J. Pan, I. Sandu Popa, K. Zeitouni, and C. Borcea. Proactive vehicular traffic re-routing for lower travel time. *Vehicular Technology, IEEE Transactions on*, 62(8):3551–3568, 2013.

[32] LG Papaleondiou and M.D. Dikaiakos. Trafficmodeler: A graphical tool for programming microscopic traffic simulators through high-level abstractions. In *Proceedings of the 69th IEEE Vehicular Technology Conference(VTC Spring 2009)*, pages 1–5. IEEE, 2009.

[33] R.A. Popa, A.J. Blumberg, H. Balakrishnan, and F.H. Li. Privacy and accountability for location-based aggregate statistics. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 653–666. ACM, 2011.

[34] H. Prothmann, H. Schmeck, S. Tomforde, J. Lyda, J. Hahner, C. Muller-Schloer, and J. Branke. Decentralized route guidance in organic traffic control. In *Proceedings of the 5th IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO 2011)*, pages 219–220. IEEE, 2011.

[35] K. Sampigethaya, M. Li, L. Huang, and R. Poovendran. Amoeba: Robust location privacy scheme for vanet. *Selected Areas in Communications, IEEE Journal on*, 25(8):1569–1589, 2007.

[36] C. Sommer, R. German, and F. Dressler. Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis. *IEEE Transactions on Mobile Computing*, 10(1):3–15, January 2011.

[37] B. Tatomir, S. Fitrianie, M. Paltanea, and L. Rothkrantz. Dynamic routing in traffic networks and manets using ant based algorithms. In *Proceedings of the 7th International Conference on Artificial Evolution, Lille, France*, October 2005.

[38] Quoc-Cuong To, Benjamin Nguyen, and Philippe Pucheral. Privacy-preserving query execution using a decentralized architecture and tamper resistant hardware. In *EDBT*, pages 487–498, 2014.

[39] Dai-Hai Ton-That, Iulian Sandu-Popa, and Karine Zeitouni. Pptm: Privacy-aware participatory traffic monitoring using mobile secure probes. In *IEEE MDM*, 2015. Demo paper.

[40] A. Wegener, M. Piórkowski, M. Raya, H. Hellbrück, S. Fischer, and J.P. Hubaux. Traci: an interface for coupling road traffic and network simulators. In *Proceedings of the 11th communications and networking simulation symposium*, pages 155–163. ACM, 2008.

[41] W. Whyte, A. Weimerskirch, V. Kumar, and T. Hehn. A Security Credential Management System for V2V Communications. In *Proceedings of the 2013 IEEE Vehicular Networking Conference*, pages 1–8, 2013.

[42] B. Wiedersheim, Z. Ma, F. Kargl, and P. Papadimitratos. Privacy in inter-vehicular networks: Why simple pseudonym change is not enough. In *Wireless On-demand Network Systems and Services (WONS), 2010 Seventh International Conference on*, pages 176–183. IEEE, 2010.

[43] D. Wilkie, J.P. van den Berg, M.C. Lin, and D. Manocha. Self-aware traffic route planning. In *AAAI*, 2011.

[44] T. Xu and Y. Cai. Feeling-based location privacy protection for location-based services. In *Proceedings of the 16th ACM conference on Computer and communications security*, pages 348–357. ACM, 2009.

Juan (Susan) Pan received her Ph.D. degree from New Jersey Institute of Technology in 2014. Her research interests include vehicular and ubiquitous computing, distributed systems, and social network analysis.

Iulian Sando Popa received his Ph.D. degree from the University of Versailles Saint-Quentin-en-Yvelines (UVSQ) in 2009. He is currently an Assistant Professor of computer science with the UVSQ and a member of the Secured and Mobile Information Systems Team at Inria. His main research interests include embedded database systems, spatiotemporal databases, and mobile data management and systems, with a particular focus on topics revolving around privacy and personal data management.

Cristian Borcea (M'04) received his Ph.D. degree from Rutgers University in 2004. He is currently an Associate Professor with the Department of Computer Science, New Jersey Institute of Technology. He is also a Visiting Associate Professor with the National Institute of Informatics, Tokyo, Japan. His research interests include mobile computing and sensing, ad hoc and vehicular networks, distributed systems, and cloud computing. Borcea is a member of the ACM and USENIX.