SAINT: Self-Adaptive Interactive Navigation Tool for Cloud-Based Vehicular Traffic Optimization

Jaehoon Jeong, *Member, IEEE*, Hohyeon Jeong, *Student Member, IEEE*, Eunseok Lee, *Member, IEEE*, Tae Oh, *Senior Member, IEEE*, and David H. C. Du, *Fellow, IEEE*

***Abstract*—This paper proposes a self-adaptive interactive navigation tool (SAINT), which is tailored for cloud-based vehicular traffic optimization in road networks. The legacy navigation systems make vehicles navigate toward their destination less effectively with individually optimal navigation paths rather than network-wide optimal navigation paths, particularly during rush hours. To the best of our knowledge, SAINT is the first attempt to investigate a self-adaptive interactive navigation approach through the interaction between vehicles and vehicular cloud. The vehicles report their navigation experiences and travel paths to the vehicular cloud so that the vehicular cloud can know real-time road traffic conditions and vehicle trajectories for better navigation guidance for other vehicles. With these traffic conditions and vehicle trajectories, the vehicular cloud uses a mathematical model to calculate road segment congestion estimation for global traffic optimization. This model provides each vehicle with a navigation path that has minimum traffic congestion in the target road network. Using the simulation with a realistic road network, it is shown that our SAINT outperforms the legacy navigation scheme, which is based on Dijkstra's algorithm with a real-time road traffic snapshot. On a road map of Manhattan in New York City, our SAINT can significantly reduce the travel delay during rush hours by 19%.**

***Index Terms*—Cloud, congestion, interactive, navigation, road network, self-adaptive, trajectory, vehicular network.**

I. Introduction

EFFICIENT vehicle navigation is very important in terms of time and fuel. In daily life, many people drive from their homes to their workplaces and back during rush hours and waste many hours and much fuel by driving on congested roadways. For efficient navigation, we use navigators in the form of either dedicated navigation systems (e.g., Garmin [1] and TomTom [2]) or smartphone navigation Apps (e.g., Waze [3] and Navfree [4]). These navigators provide vehicles with their navigation paths from the source to the destination with road traffic statistics or real-time traffic conditions. Efficient navigation will save time and fuel effectively; hence, research on navigation will continue in the future.

Recently, vehicular ad hoc networks (VANETs) have been spotlighted for communications among vehicles and infrastructure for driving safety, driving efficiency, and entertainment services [5], [6]. The VANET can be constructed by dedicated short-range communications (DSRC) [7] among vehicles moving either in one road segment or in multiple road segments. This DSRC technology has been standardized by IEEE 802.11p that is an extension of IEEE 802.11a for vehicular networks. Since these DSRC-based vehicular networks will be deployed by the government for public safety and convenience, they can mitigate the cost of mobile users caused by cellular networks (e.g., fourth-generation Long-Term Evolution (4G-LTE) [8], [9]) that mainly provide mobile devices (e.g., vehicles and smartphones) with mobile wireless network access to cloud services (e.g., navigation). Moreover, as previously mentioned, Global Positioning System (GPS) navigation systems are popularly used by drivers for efficient driving. Due to the trend of wireless access diversity and navigator popularity, one natural research question is how to design an efficient navigation system by utilizing both real-time traffic conditions and vehicle navigation paths.

In this paper, a vehicular cloud for navigation is constructed with the following settings. As a core computing and storage node in the vehicular cloud, the Traffic Control Center (TCC) [10] collects road traffic statistics from road infrastructure and vehicles. The TCC also maintains the trajectories of vehicles in the target road network and services the navigation of vehicles as a vehicular cloud system. As wireless nodes connected to the Internet, road-side units (RSUs) [11] are deployed at intersections to provide vehicles with Internet connectivity to the TCC. Vehicles with navigation systems communicate with RSUs to interact with the TCC for their navigation. The vehicles periodically report their own trajectory (i.e., future navigation path) and their current position to the TCC. Of course, if RSUs are not available, the self-adaptive interactive navigation tool (SAINT) allows vehicles to communicate with base stations in cellular networks, such as evolved Node B (eNodeB) in 4G-LTE [8], for wireless access to the vehicular cloud. Nowadays, most current navigation services [3], [4], [12], [13] are provided for mobile devices via 4G-LTE communications.

Manuscript received March 15, 2015; revised July 10, 2015; accepted August 23, 2015. Date of publication September 4, 2015; date of current version June 16, 2016. This work was supported by the National Research Foundation of Korea funded by the Ministry of Science, ICT, and Future Planning through the Basic Science Research Program (2014006438) and the Next-Generation Information Computing Development Program (2015045358). The review of this paper was coordinated by Dr. A. Chatterjee. *(Corresponding author: Eunseok Lee.)*

J. Jeong is with the Department of Interaction Science, Sungkyunkwan University, Seoul 03063, Korea (e-mail: pauljeong@skku.edu).

H. Jeong and E. Lee are with the Department of Computer Science and Engineering, Sungkyunkwan University, Suwon 16419, Korea (e-mail: jeonghh89@skku.edu; leees@skku.edu).

T. Oh is with the Department of Information Sciences and Technologies, Rochester Institute of Technology, Rochester, NY 14623 USA (e-mail: tom.oh@rit.edu).

D. H. C. Du is with the Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN 55455 USA (e-mail: du@cs.umn.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVT.2015.2476958

In the form of a *one-way noninteractive navigation service*, most current navigation systems [1]–[4], [12]–[14] provide each vehicle with an individually optimized navigation path instead of the overall optimization for the road network. When some road segments are less congested, those navigation systems will construct navigation paths using the light-traffic road segments. In this case, many vehicles will simultaneously use the light-traffic road segments for their travel paths in a greedy way. As a result, those road segments will soon be congested with high probability. This phenomenon happens because the current navigation systems compute time-wise shortest travel paths, based only on the snapshot of road traffic conditions without considering the near-future congestion in currently light-traffic road segments. Thus, based on the snapshot of road traffic conditions, this locally optimal navigation scheme does not work effectively during rush hours with high vehicular traffic, leading to ineffective navigation service.

This paper proposes the SAINT for cloud-based vehicular traffic optimization. To the best of our knowledge, this paper is the first attempt to guiding vehicles through global transportation optimization, utilizing vehicle trajectories and predicting bottleneck road segments in a target road network. In the form of a *two-way interactive navigation service*, the SAINT allows vehicles to interact with the vehicular cloud for optimal navigation experience with vehicles reporting their current trajectories and navigation experiences. This two-way interactive navigation service prevents vehicles from simultaneously using currently light-traffic road segments for their travel in a greedy way. The TCC predicts the near-future congested road segments through the estimation of road segment congestion levels along with the trajectories reported by the vehicles over time. Whenever a new vehicle requests its navigation path to the TCC, the TCC selects a globally optimal navigation path as its vehicle trajectory that does not include highly congested road segments in the near future with a high probability. Therefore, this coordination by the TCC can spread out vehicular traffic throughout the target road network, leading to the fast navigation of all the vehicles in the target road network. Our intellectual contributions are as follows.

- **Vehicular navigation architecture**: We propose a two-way interactive navigation system. The vehicles interact with the TCC using RSUs in DSRC [7] or eNodeBs in 4G-LTE [8] so that the TCC can compute the optimal navigation paths for newly navigating vehicles. The paths are computed by identifying future congested road segments using the reported vehicle trajectories as the vehicles move in the target road network.
- **Congestion contribution formulation**: Given a vehicle's trajectory, a certain congestion weight is accumulated to each road segment on the trajectory for the future congestion level, considering the arrival time that the vehicle will reach the road segment.
- **Trajectory computation algorithm**: Considering the congestion contributions on road segments, the algorithm computes a trajectory that has a bounded travel delay increase (e.g., by, at most, 50%) for the individually optimized shortest path but minimizes the path congestion contribution in the target road network.

Therefore, these contributions will be able to reduce road traffic congestion in urban areas during rush hours through our cloud-based navigation service along with real-time road traffic, saving both time and fuel.

The rest of this paper is organized as follows. Section II summarizes and analyzes the current navigation systems. Section III formulates our navigation scheme. In Section IV, we describe our travel delay model for travel time prediction. Section V explains the design of SAINT navigation. Section VI describes the navigation procedure in the SAINT. Section VII evaluates the SAINT along with a state-of-the-art legacy navigation system under realistic settings. Finally, in Section VIII, we conclude this paper along with future work.

II. Related Work

Currently, in the industry, GPS navigation systems are popularly used in the form of either dedicated navigators (e.g., Garmin [1], TomTom [2], and iNAVI [14]) or smartphone navigator Apps (e.g., Waze [3], Navfree [4], Skobbler [12], and Tmap [13]). Most of them are using the shortest travel delay path algorithm (e.g., Dijkstra's shortest path algorithm [15]) based on real-time road traffic measurement or road traffic statistics. In the past, most dedicated navigators worked on the basis of the geographically shortest path, but some navigators worked on the basis of the time-wise shortest travel path with real-time traffic information by smartphone tethering or road traffic statistics according to each hour in a day. The limitation of the legacy navigators is that the routes provided by them are individually optimized paths for each vehicle [1]–[4], [12]–[14]. That is, they do not consider the collaboration among vehicles in congested road areas for better navigation. On the other hand, our SAINT predicts road segments that have a high possibility for traffic congestion in the near future and then allows vehicles to bypass those road segments by detouring. Eventually, since vehicles navigate by the coordination of the TCC, the traffic spreads out evenly throughout the road network to provide uniform traffic density. This uniform traffic density allows the vehicles to experience less traffic congestion on road segments and shorter waiting time at intersections. Finally, our SAINT assists drivers with optimal and adaptive navigation information from the vehicles in the target road network.

In academia, travel path planning for efficient navigation has been researched [16], [17]. Wang *et al.* proposed real-time path planning based on both cellular networks and VANETs [16]. The proposed scheme considers both each individual driver's driving preferences (e.g., short travel distance and driving easiness) and the overall road network utilization. However, this scheme cannot handle the case where many vehicles will use the same road segment, which is currently idle, in the near future in a similar time zone. As a result, the vehicles will experience traffic congestion in the near future. On the other hand, our SAINT uses the concept of the reservation of each road segment along the vehicle navigation path in terms of congestion increase by the vehicle in the timeline. Thus, the SAINT prevents some road segments from being congested by spreading out vehicle traffic uniformly in the road network. Khosroshahi *et al.* proposed a real-time traffic collection scheme

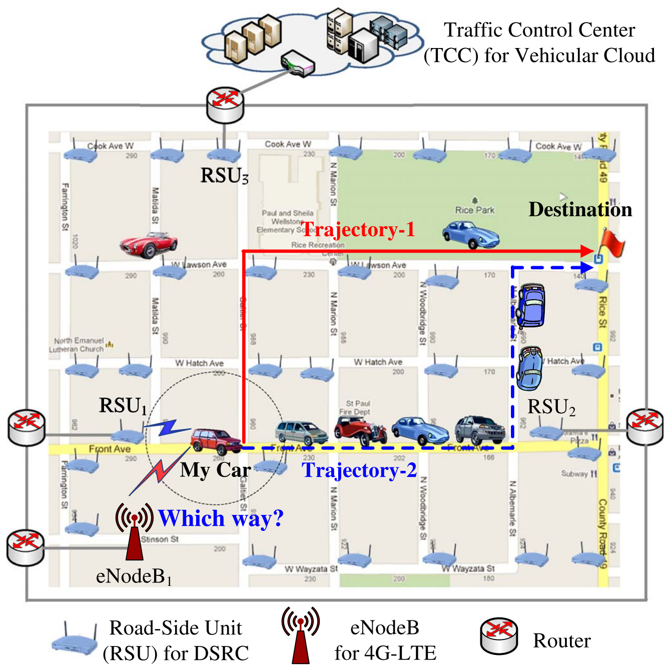


Fig. 1. Vehicular navigation architecture.

based on VANETs [17]. This scheme defines an appropriate cost function and its parameters for route guidance with real-time traffic information. With the cost function, this scheme uses the A^* search algorithm for best-route search [18]. In the same way with [16], this scheme provides an optimal travel path for individual vehicles rather than the whole road network; hence, it cannot achieve global optimization for all the vehicles moving in the road network. Therefore, since the legacy schemes in the industry and academia are based on the current network conditions without the reservation concept in SAINT, they allow currently idle road segments to be congested soon by vehicles that greedily take those idle road segments as parts of their navigation paths.

III. Problem Formulation

This section describes the goal, assumptions, and high-level design for our navigation system called SAINT. Given the destinations of vehicles in a target road network, our goal is to determine an optimal vehicle trajectory (i.e., navigation path) of each vehicle. Our SAINT aims to minimize the traffic congestion level per road segment for network-wide traffic optimization, while bounding the expected vehicle detour travel time. The increased detour delay is, at most, the product of a detour constraint parameter α (e.g., 50%) and the shortest travel time from the source to the destination computed by Dijkstra's algorithm [15], as discussed in Section V-C.

A. Vehicular Navigation Architecture

This section describes our vehicular navigation architecture and component nodes for vehicular cloud. Fig. 1 shows the vehicular navigation architecture for our SAINT system. The following items define nodes in the vehicular navigation architecture.

- **TCC**: The TCC is a road traffic management node for a vehicular cloud system in a target road network [10]. The TCC maintains the trajectories and locations of vehicles for location management as used in Mobile IPv6 [19]. The TCC has up-to-date vehicular traffic statistics, such as vehicle arrival rate and average speed per road segment in the road network under its management. With the vehicular traffic statistics and vehicle trajectories for the target road network, the TCC computes the vehicular traffic congestion level per road segment. With this congestion level per road segment, the TCC computes an optimal navigation path for a new navigation vehicle or a reroute requested vehicle with the vehicle trajectory for the demanding vehicle. Section V-A explains how to compute the congestion level per road segment for a given vehicle trajectory as a navigation route candidate. Section V-B explains how to select a vehicle trajectory along with the congestion level information and vehicular traffic statistics. For a large-scale road network, one TCC may not be able to accommodate a large number of vehicles for the navigation service. To address the scalability, the large-scale road network could be separated into multiple regions, where each region could have a dedicated TCC. Moreover, even in each region, the TCC can have multiple servers for efficient navigation service by allowing them to have the replicas of the vehicle trajectories for computation. The design issues for this vehicular cloud are left as future work.
- **RSU**: An RSU is a wireless gateway to connect a wireless VANET to the wired network (i.e., the Internet) [11]. The RSU has a DSRC communication device to communicate with vehicles with a DSRC communication device. One RSU is deployed at each intersection or road segment in a target road network such that vehicles can exchange data for navigation with the RSU. RSUs are connected to each other through wired networks (e.g., the Internet). RSUs play a role of a backbone communication network in the target road network. Moreover, RSUs collect the trajectories of the vehicles participating in SAINT navigation service and report them to the TCC. Furthermore, they receive the route requests from vehicles and deliver them to the TCC. When receiving the route responses including vehicle trajectories from the TCC, they forward those responses to the navigating vehicles.
- **eNodeB**: eNodeB is a base station that connects mobile devices (e.g., smartphones and vehicles) to 4G-LTE cellular networks [8], [9]. It allows vehicles to access vehicular cloud in the TCC in an ubiquitous way, that is, anywhere and anytime. Whenever a vehicle cannot communicate with RSUs, it will contact a nearby eNodeB to access the vehicular cloud.
- **Vehicle**: The vehicle is equipped with a DSRC communication device [7], [20] to communicate with RSUs and a 4G-LTE communication device [8] to communicate with eNodeBs. The vehicle is also equipped with a

GPS-based navigation system having digital road maps [1]–[4], [12], [13]. Vehicles report their travel experience in road segments and at intersections along their travel path to let RSUs compute road traffic statistics (such as the mean and variance of the travel time for each road segment).

B. Assumptions

This section lists assumptions for the SAINT as follows.

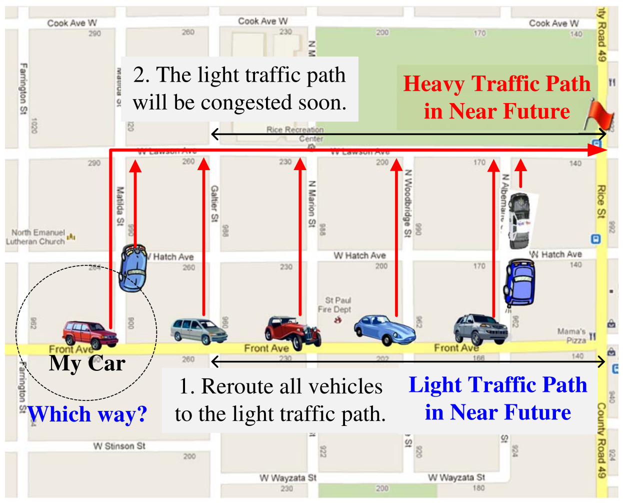
- Vehicles can work with multiple wireless links, such as DSRC [7], [20] in vehicular networks and 4G-LTE [8] in cellular networks in a cost-effective way. Although the current navigation services [3], [4], [12], [13] are provided to mobile devices using 4G-LTE, our SAINT will accommodate vehicles in DSRC vehicular networks that will be deployed by governments and be opened to drivers without service charge for the safety and efficiency operations in road networks. These vehicular networks will be able to reduce the cost of 4G-LTE usage by allowing vehicles to maximize the usage of DSRC links and minimize the usage of 4G-LTE links. Under either fault in communication systems or error in wireless media, the SAINT can perform its navigation task well, since a vehicle's travel delay is several orders of magnitude longer than the communication delay. For example, it takes 90 s for a vehicle to travel along a road segment of 1 mi with a speed of 40 mi/h; however, it takes only tens of milliseconds to forward a packet to an RSU or an eNodeB, even after considering the retransmission due to wireless link noise or packet collision. Thus, this travel delay can accommodate both packet delivery delay and computation time in the vehicular cloud for the navigation service. Therefore, during their travel, vehicles can smoothly interact with the TCC by DSRC links or cellular links to exchange navigation information between vehicles and the TCC.
- Vehicles are equipped with GPS-based navigation systems and digital road maps for navigation on the road network [1]–[4], [12], [13]. Traffic statistics, such as vehicle arrival rate λ and average vehicle speed v per road segment, are collected and processed by RSUs and reported to the TCC. With these traffic statistics, the TCC computes link travel delay per road segment and intersection waiting time per intersection for selecting navigation paths for vehicles moving in a target road network.
- The TCC can accommodate all of the vehicles moving in its corresponding road network for the navigation service, while tracking the position and direction of vehicles subscribed to the SAINT navigation service. On a large-scale road network, one TCC might not scale up to provide large numbers of vehicles for navigation service. For scalability, the TCC can have multiple servers good enough to compute trajectories in a prompt way [21]. That is, as more computation power is required for navigation service, the TCC can be equipped with more servers in vehicular cloud. Moreover, the large-scale road network can be divided into multiple regions that have their own TCC for navigation service. For the sake of privacy and security, the navigation request and response between vehicles and the TCC are encrypted and decrypted in a privacy-preserving manner.
- Drivers input their travel destination into their GPS-based navigation system as a navigation request before their travel starts. The navigation system sends the navigation request to the TCC. As a vehicular cloud, the TCC computes the vehicle trajectory based on the current location and the final destination of each vehicle. SAINT-service-participatory vehicles send navigation requests to the TCC through either an **RSU** by the vehicle-to-infrastructure (V2I) data delivery scheme [6], [22] or an **eNodeB** by 4G-LTE communications [8]. They also receive navigation responses from the TCC via either an **RSU** by the infrastructure-to-vehicle (I2V) data delivery scheme [23], [24] or an **eNodeB** by 4G-LTE communications.

C. Concept of Self-Adaptive Interactive Navigation

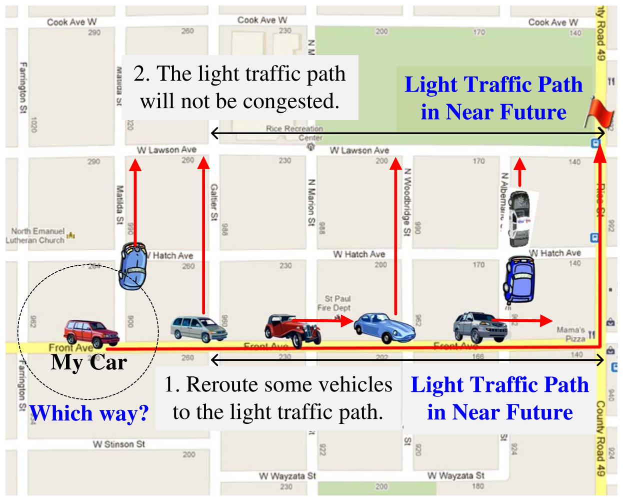
This section explains the concept of *self-adaptive interactive navigation*. We define *self-adaptiveness* in the viewpoint of the *road network* rather than *individual vehicles*. Thus, our *self-adaptive interactive navigation* means that the TCC in the road network provides an efficient navigation service for vehicles through the interaction with the vehicles, considering dynamically changing traffic conditions in road segments or intersections. The TCC identifies future bottleneck road segments with the vehicle trajectories and provides the navigating vehicles with the navigation paths avoiding such future bottleneck road segments. The detailed interaction between the TCC and vehicles will be explained in Section VI.

Fig. 2(a) shows local traffic optimization for individual vehicles, which is used by legacy navigators [1]–[4], [12], [13]. Since the upper road network area is less congested, all of the vehicles in the high-traffic path can simultaneously reroute their paths toward the alternative path, which is a light-traffic path, as shown in the figure. The simultaneous rerouting will cause the alternative path to be congested very soon. Thus, this local traffic optimization causes traffic congestion in light-traffic road segments because it allows all of the vehicles to behave in a greedy way only for their own navigation. As a result, the vehicles will take longer navigation time due to the traffic congestion from individually optimal navigation paths, not considering global traffic optimization.

The main contribution of the SAINT is to spread out vehicular traffic throughout the road network using a global traffic optimization approach. For this global optimization, the SAINT takes a collaborative approach based on the interaction between the TCC and vehicles via RSUs or eNodeBs in the vehicular cloud system, as shown in Fig. 1. As shown in Fig. 2(b), the SAINT supports global navigation optimization, considering the mobility of all the vehicles in a target road network. In Fig. 2(b), only some of the vehicles in a certain area reroute to the light-traffic road segments rather than all of the vehicles in the area. This strategy allows the high-traffic road segments to be lighter and all of the road segments to serve the vehicles almost with the same traffic load. As a result, the vehicles


2. The light traffic path will be congested soon.
Heavy Traffic Path in Near Future
My Car
Which way?
1. Reroute all vehicles to the light traffic path.
Light Traffic Path in Near Future

(a)


2. The light traffic path will not be congested.
Light Traffic Path in Near Future
My Car
Which way?
1. Reroute some vehicles to the light traffic path.
Light Traffic Path in Near Future

(b)

Fig. 2. Vehicular traffic optimization. (a) Local traffic optimization. (b) Global traffic optimization.

will experience not only shorter travel delay at road segments but shorter queuing delay at intersections as well, overall leading to faster navigation toward their destination. Therefore, the SAINT can achieve global traffic optimization in the target road network. In the following section, we will explain the travel delay prediction for our global traffic optimization approach.

IV. Travel Delay Prediction

This section explains the modeling of travel delay on both road segments and an end-to-end (E2E) travel path, based on our early work [24], [25].

A. Travel Delay on Road Segment

Many researchers on transportation have demonstrated that the travel delay of one vehicle over a fixed distance in light-traffic vehicular networks follows the Gamma distribution [24]–[26]. The travel delay through a road segment i is defined as link travel delay. Let d_i denote the link travel delay for road segment i. $d_i \sim \Gamma(\kappa_i, \theta_i)$, where κ_i is a shape parameter, and θ_i is a scale parameter [27]; note that $d_i \sim \Gamma(\alpha_i, \beta_i)$, where

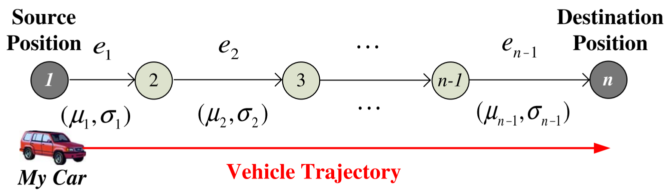


Fig. 3. E2E vehicle trajectory for navigation.

$\alpha_i (= \kappa_i)$ is a shape parameter, and $\beta_i (= 1/\theta_i)$ is an inverse-scale parameter [27]. Parameters κ_i and θ_i can be computed by mean μ_i and variance σ_i^2 of link travel delay d_i, using the formulas for κ_i and θ_i in [24], where the traffic statistics μ_i and σ_i^2 can be computed through 1) the travel experience reports from vehicles participating in our navigation service or 2) the vehicular traffic measurement at loop detectors in intelligent transportation systems [28], [29].

B. Travel Delay on E2E Path

This section explains the delay model of an E2E travel path (i.e., vehicle trajectory) from one position to another in a given road network [24], [25]. As previously discussed, the link travel delay is modeled as the Gamma distribution of $d_i \sim \Gamma(\kappa_i, \theta_i)$ for road segment i. Note that, as shown in Fig. 3, two contiguous edges (e.g., e_1 and e_2) are connected via a common vertex (e.g., n_2) corresponding to an intersection; hence, there are some *intersection waiting delay* at the common vertex. For simplicity, we assume that *intersection waiting delay* w_j at intersection n_j for an edge e_{ij} (denoted as e_i) is included in *link travel delay* d_i.

Given a specific travel path (i.e., vehicle trajectory), we assume that the link travel delays of different road segments on the path are independent. Under this assumption, we approximate the mean (or variance) of the E2E travel delay as the sum of the means (or variances) of the link travel delays for the links along the E2E path. In the case where the vehicle trajectory consists of $n-1$ road segments, as shown in Fig. 3, the mean and variance of the E2E travel delay D are computed on the basis of link travel delay independence as follows:

$$E[D] = \sum_{i=1}^{n-1} E[d_i] = \sum_{i=1}^{n-1} \mu_i \tag{1}$$

$$Var[D] = \sum_{i=1}^{n-1} Var[d_i] = \sum_{i=1}^{n-1} \sigma_i^2. \tag{2}$$

From (1) and (2), we model the E2E travel delay D as a Gamma distribution as follows: $D \sim \Gamma(\kappa_D, \theta_D)$, where κ_D and θ_D are computed by $E[D]$ and $Var[D]$, using the formulas for κ_D and θ_D in [24]. It is noted that our travel delay prediction can accommodate any better E2E path delay estimation if it is available from either another mathematical model (considering traffic congestion) or empirical measurement (e.g., navigator's travel experience in real time). Therefore, we can compute a vehicle's E2E travel delay from the source to the destination for a given vehicle trajectory. In the following section, based on our delay model, we will explain the design of our SAINT navigation.

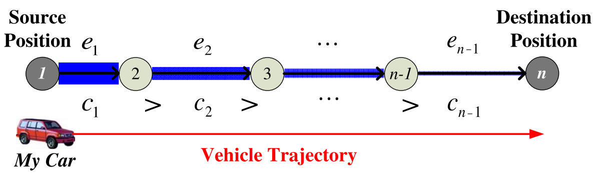


Fig. 4. Link congestion contribution model.

V. Design of Self-Adaptive Interactive Navigation Tool Navigation

This section explains the design of SAINT navigation with the following three parts: 1) link congestion contribution metric; 2) delay-constrained shortest path (DSP) algorithm; and 3) detour constraint parameter selection. First, we define a road network graph for a given target road network for navigation.

Definition V.1 (Road Network Graph): Let the **Road Network Graph** be a directed graph $G = (V, E)$ for a road map, where V is the set of vertices (i.e., intersections), and E is the set of directed edges e_{ij} (i.e., road segments) for $i, j \in V$. Let d_{ij} be the link travel delay for edge e_{ij} whose length is l_{ij} and whose average speed is v_{ij}. Note that d_{ij} includes intersection waiting delay w_j at exit intersection j toward the next edge. Thus, the link travel delay for edge e_{ij} is computed as $d_{ij} = l_{ij}/v_{ij} + w_j$.

The goal of each navigator in SAINT is *to find the lowest congestion increase travel path from source position u to destination position v, while satisfying the travel delay increase bound* (e.g., *α-percent increase*) *for a time-wise shortest travel path.* Let α be a delay increase threshold (e.g., 30%) for the time-wise shortest travel path delay as the detour constraint parameter. With this travel delay increase bound, we define **α-increase travel path** as follows.

Definition V.2 (α-Increase Travel Path): Let D_{uv} be the travel delay of a time-wise shortest travel path p_{uv} from source position u to destination position v. Let **α-Increase Travel Path** $\hat{p}_{uv}$ be a travel path whose travel delay $\hat{D}_{uv}$ does not exceed α percent of travel delay D_{uv} plus D_{uv} such that $\hat{D}_{uv} \leq (1+\alpha)D_{uv}$.

A. Link Congestion Contribution Metric

This section introduces a **link congestion contribution metric** in each road segment along a vehicle trajectory for global traffic optimization. This link congestion contribution metric measures *the congestion level of each road segment* caused by both current and near-future vehicular traffic. Fig. 4 shows our link congestion contribution model for a given vehicle trajectory. **Link congestion contribution** (denoted c_i) is defined as *the increase in vehicular traffic volume* caused by a vehicle in the fractional number of vehicles passing through an edge (denoted as e_i) currently or in the near future. In this figure, a vehicle (labeled My Car) has its vehicle trajectory such that $e_1 \rightarrow e_2 \rightarrow \cdots \rightarrow e_{n-1}$. For this given vehicle trajectory, a link congestion contribution for each edge e_i is denoted as c_i. **Our design of link congestion contribution** is *that as the edge is farther away from the vehicle, the link congestion contribution for the edge decreases.* This design reflects the observation that the edge farther away from the vehicle will be visited later by

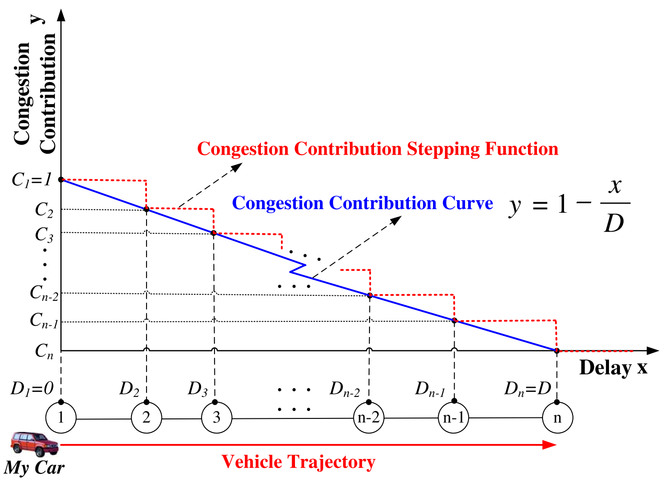


Fig. 5. Congestion contribution curve.

the vehicle in time, and the vehicle will contribute to vehicular traffic volume for the edge later.

For example, in Fig. 4, the first edge e_1 has the link congestion contribution of the vehicle as $c_1 = 1$ because the vehicle enters edge e_1; hence, e_1 becomes having one more vehicle. According to our design of link congestion contribution, the inequality in link congestion contributions of the edges along the trajectory is that $c_1 > c_2 > \cdots > c_{n-1}$. This inequality means that the vehicle will contribute to vehicular traffic volume by a fractional number inversely proportionally to the visit time (i.e., travel time) to each edge along its vehicle trajectory.

Now, we explain how to implement our design of link congestion contribution mathematically such that the congestion contribution curve is a linearly decreasing function for a given vehicle travel time to an edge along a vehicle trajectory, as shown in Fig. 5. We have the following settings for the optimization for a target road network graph G. Let $M = (m_{ij})$ be a **congestion contribution matrix**, where m_{ij} is the *link congestion contribution metric* (i.e., *cumulative link congestion contribution value*) by the vehicle trajectories of all vehicles, passing currently or in the near future through road segment e_{ij} for $i, j \in V$. Let v_s be a vehicle with the SAINT navigator. Let c_{ij}^s be the *per-trajectory link congestion contribution value* by the vehicle trajectory of a vehicle v_s passing currently or in the near future through edge e_{ij}. Thus, the link congestion contribution metric m_{ij} for an edge e_{ij} is the sum of c_{ij}^s's for all vehicles $v_s \in S$ passing through edge e_{ij}, where S is the set of vehicles in the target road network graph G.

Let $T_{1,n}$ be the vehicle trajectory that is the path of intersections visited by vehicle v_s such that $T_{1,n} = 1 \rightarrow 2 \rightarrow 3 \rightarrow \cdots \rightarrow n$ without loss of generality, as shown in Fig. 4. Let an edge e_i denote e_{ij} for $i = 1, \ldots, n-1$ on trajectory $T_{1,n}$. Let a link delay d_i denote d_{ij} for $i = 1, \ldots, n-1$ on $T_{1,n}$. Let D_i be the subpath delay from the starting intersection 1 to an intermediate intersection i on $T_{1,n}$ such that $D_i = \sum_{k=1}^{i-1} d_k$ for $i = 1, \ldots, n$ on $T_{1,n}$. Note that D_1 for the starting intersection 1 is 0 as the initial value for trajectory $T_{1,n}$ and that $D_n (= D)$ is the travel delay from the starting intersection 1 to the destination intersection n such that $D_n = \sum_{k=1}^{n-1} d_k$. Let a congestion contribution c_i denote c_{ij}^s for $i = 1, \ldots, n-1$ on

$T_{1,n}$, which represents the congestion contribution of vehicle v_s to edge e_{ij} on trajectory $T_{1,n}$.

Now, we show a formula to compute the congestion contribution c_i per edge e_i on a given trajectory $T_{1,n}$ as follows:

$$c_i = 1 - \frac{D_i}{D}. \tag{3}$$

Fig. 5 shows the **congestion contribution curve** that is a *linear decreasing function* $y = 1 - (x/D)$, where x is the travel time from the starting intersection 1 to an intermediate position (e.g., intersection) on the vehicle trajectory $T_{1,n}$, and D is the E2E travel time along the vehicle trajectory. In our design, we use (3) to compute the congestion contribution on road segment e_i caused by vehicle v_s according to the vehicle arrival time on e_i. Equation (3) has the form of this *linear function* of the **congestion contribution curve** in Fig. 5. This congestion contribution curve considers both *vehicle velocity* and *traffic light*. The travel time from an intersection i to the next intersection j includes the link travel delay from i to j (denoted as d_{ij}) and waiting delay at intersection j (denoted as w_j). Link travel delay d_{ij} is determined by vehicle velocity v_{ij}, and intersection waiting delay w_j is determined by traffic light scheduling such that $d_{ij} = l_{ij}/v_{ij} + w_j$ for the given edge length l_{ij}, as specified in Definition V.1.

In our design, the congestion contribution c_i for edge e_i is constantly maintained during the link travel on e_i. When a vehicle enters and travels on edge e_i, we regard one vehicle occupying the road segment with the congestion increase c_i computed before the vehicle started navigation until the vehicle passes through the road segment. Therefore, the congestion contribution for the road segments on the trajectory uses the **congestion contribution stepping function**, as shown in Fig. 5. Note that our design can accommodate any better congestion contribution curve (e.g., nonlinear decreasing function) if it is available from another mathematical modeling (considering traffic congestion).

We now explain **a key point in our congestion contribution modeling**. As road segment e_i is farther away from the starting intersection n_1, vehicle v_s will reach e_i later. Therefore, v_s will contribute to the increase in vehicular traffic congestion on the farther road segment e_i by a lesser amount right now. In other words, vehicle v_s reserves its future passing to the edge by the vehicular traffic increase inversely proportional to the visiting time to the edge. By this rationale, (3) makes the congestion contribution c_i caused by vehicle v_s on road segment e_i be decreasing as i increases along trajectory $T_{1,n}$. Finally, we can compute $c_1, c_2, \ldots, c_{n-1}, c_n$ by (3) such that $c_1 > c_2 > \cdots > c_{n-1} > c_n$ because $D_1 < D_2 < \cdots < D_{n-1} < D_n$, as shown in Fig. 4. Thus

$$\begin{aligned}
c_1 &= 1 - \frac{D_1}{D} = 1\\
c_2 &= 1 - \frac{D_2}{D} < 1\\
&\cdots\\
c_{n-1} &= 1 - \frac{D_{n-1}}{D} \ll 1\\
c_n &= 1 - \frac{D_n}{D} = 0 \text{ for } D_n = D.
\end{aligned}$$

Therefore, the congestion contribution c_{ij}^s (denoted as c_i) is computed for each road segment e_{ij} on trajectory $T_{1,n}$ of vehicle v_s. These link congestion contribution values c_{ij}^s's will be added to the link congestion contribution metric m_{ij} in congestion contribution matrix M before vehicle v_s starts navigation along its trajectory $T_{1,n}$. Note that the congestion contribution matrix M is maintained and updated by the vehicular cloud for the navigation path computation. During navigation, whenever vehicle v_s passes through edge e_{ij}, the corresponding congestion contribution c_{ij}^s is subtracted from the link congestion contribution m_{ij} in M.

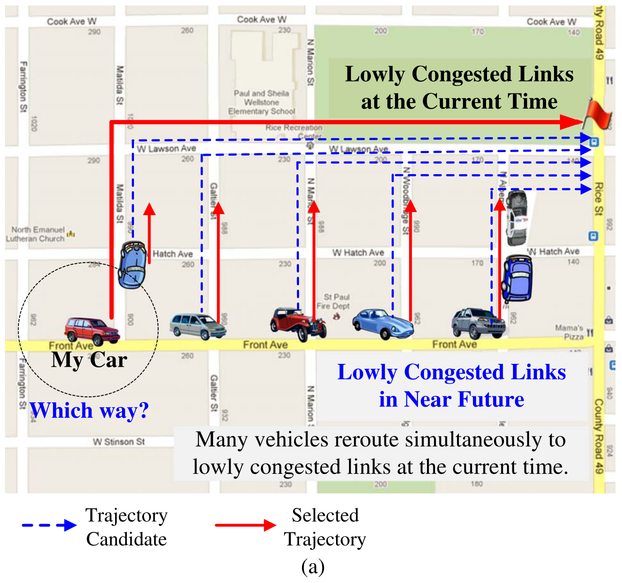
So far, we have explained the design of SAINT navigation along with the manipulation of link congestion contribution metric m_{ij} and congestion contribution c_{ij}^s on an edge e_{ij} involved in a vehicle trajectory. In the following section, we will explain a navigation trajectory selection algorithm considering the bounded detour delay.

B. Delay-Constrained Shortest Path Algorithm

This section describes a delay-constrained shortest path (DSP) algorithm for computing a navigation trajectory. Given a target road network graph $G = (V, E)$, **the goal of our DSP algorithm** is *to let a navigator find the smallest congestion increase path from its source intersection u to its destination intersection v, while guaranteeing the delay constraint of the α-increase travel path.*

First, the difference between Dijkstra's algorithm and our DSP algorithm is explained in Fig. 6. As shown in Fig. 6(a), Dijkstra's algorithm makes many vehicles reroute to lowly congested links simultaneously for their travel with the road traffic snapshot information (i.e., short-term traffic statistics). This routing will make the lowly congested links be populated and congested by those rerouting vehicles in a short time. On the other hand, as shown in Fig. 6(b), the DSP algorithm lets vehicles select their vehicle trajectory, considering the future congestion in links. As a result, some vehicles will reroute to lowly congested links at this point, and other vehicles will keep going along their original trajectory, as shown in Fig. 6(b). Therefore, since the vehicles will be spread out over a certain area of the road network, those vehicles will quickly navigate toward their destination.

Now, let us explain our DSP algorithm in detail as follows: In Algorithm 1, a delay-constrained time-wise shortest path P_{dsp} is returned for the input of the target road network graph G, source u, and destination v along with the α-increase value. In line 2, P_{dsp} is allocated for a list of intersections for the delay-constrained time-wise shortest path. Line 3 computes the shortest travel delay from source u to destination v in G by Dijkstra's shortest path algorithm and stores the delay into D_{uv}. Line 4 computes the α-percent delay to select a shortest travel path with α-percent delay, while minimizing the congestion contribution for the E2E path from u to v. In line 5, *Compute-k-Smallest-Congestion-Increase-Paths()* computes k shortest paths in terms of the congestion contribution value. These paths are called k *smallest congestion increase paths* because they increase the *minimum congestion contribution value* in the congestion contribution matrix M for the target



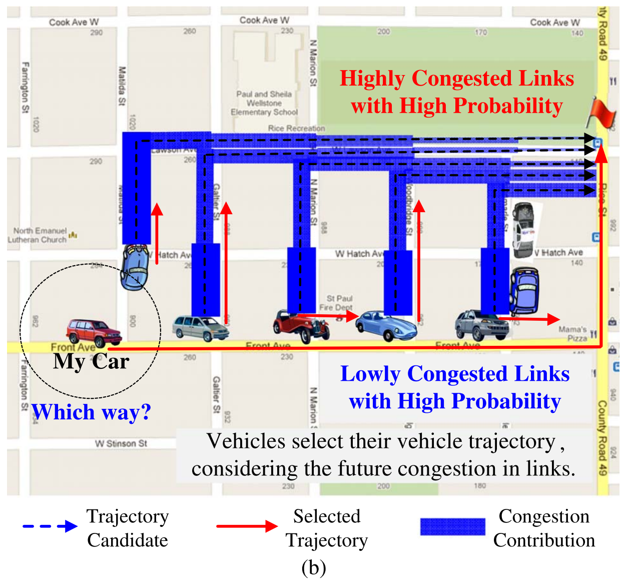


Fig. 6. Vehicle trajectory selection for navigation. (a) Selection by Dijkstra's algorithm. (b) Selection by the DSP algorithm.

road network. They are computed by *Yen's k-shortest-path algorithm* [30] along with matrix M and stored into a set K. In line 6, we set n to the number of the available shortest paths in K for the for-loop in lines 7–13 such that $n = |K|$. In lines 7–13, a delay-constrained time-wise shortest path P_{dsp} is selected such that P_{dsp} is the smallest congestion increase path, while satisfying the delay constraint of α-increase for the shortest travel delay path computed by Dijkstra's algorithm. If there is no such path among the k smallest congestion increase paths in K, in line 14, the time-wise shortest path computed by Dijkstra's algorithm is set to P_{dsp} as a last resort.

Algorithm 1 DSP Algorithm

```
1: function CONSTRUCT-DELAY-CONSTRAINED-SHORTEST-PATH($(G, u, v, \alpha)$)
2:     $P_{\text{dsp}} \leftarrow \emptyset$ ▷ $P_{\text{dsp}}$ will contain the list of intersections for a DSP.
3:     $D_{uv} \leftarrow$ Compute-Dijkstra-Path-Value$(G, u, v)$ ▷ $D_{uv}$ is the time-wise shortest travel delay from $u$ to $v$ in $G$ by Dijkstra's shortest path algorithm.
4:     $\Delta_{uv} \leftarrow \alpha \times D_{uv}$ ▷ $\Delta_{uv}$ is the $\alpha$-percent delay increase for $D_{uv}$.
5:     $K \leftarrow$ Compute-$k$-Smallest-Congestion-Increase-Paths$(G, u, v)$ ▷ compute the next $k$ smallest congestion increase paths arranged in nondecreasing order by Yen's $k$-shortest-path algorithm.
6:     $n \leftarrow$ Count-Path-Numbers$(K)$ ▷ count the number of paths in $K$.
7:     for $i \leftarrow 1, n$ do
8:         $D \leftarrow$ Compute-Path-Value$(K, i)$ ▷ compute the travel delay for the $i$th path in $K$.
9:         if $D \le D_{uv} + \Delta_{uv}$ ▷ check the travel delay constraint of $\alpha$-percent increase.
10:            $P_{\text{dsp}} \leftarrow$ Get-Path$(K, i)$ ▷ get the $i$th path in $K$.
11:            return $P_{\text{dsp}}$
12:        end if
13:    end for
14:    $P_{\text{dsp}} \leftarrow$ Compute-Dijkstra-Path$(G, u, v)$ set $P_{\text{dsp}}$ to the time-wise shortest travel delay path from $u$ to $v$ in $G$ by Dijkstra's shortest path algorithm.
15:    return $P_{\text{dsp}}$
16: end function
```

The time complexity of Algorithm 1 is $O(kN(M + N \log N))$, where $N = |V|$, $M = |E|$, and k is the number of paths in the k-shortest-path algorithm [called *Compute-k-Smallest-Congestion-Increase-Paths()*] in line 5 [30]. This is because the k-shortest-path algorithm is the dominant function in Algorithm 1, and its time complexity is $O(kN(M + N \log N))$ [30].

Once a trajectory P_{dsp} is selected as a travel path for v_s in the road network graph G, the congestion contribution value c_{ij} for edge e_{ij} on the trajectory will be added to link congestion contribution metric m_{ij} on the congestion contribution matrix $M = (m_{ij})$ for $i, j \in V$. This matrix M is used to select a travel path for v_s in line 5 in Algorithm 1. Whenever the vehicle passes through an edge $e_{ij} \in P_{\text{dsp}}$, the corresponding congestion contribution value c^s_{ij} is subtracted from the corresponding matrix entry m_{ij} in M. Note that since the TCC computes a time-wise shortest travel path for a navigating vehicle as a cloud system, matrix M is maintained by the TCC interacting with navigating vehicles.

C. Detour Constraint Parameter Selection

In the previous section, a DSP is computed using a detour constraint parameter α to determine the detour-determinant factor. That is, to prevent vehicles from being directed to light-traffic road segments in the near future, the α-increase travel delay path lets vehicles travel uniformly in road segments in a target road network. This prevents light-traffic road segments from being highly congested in a short time. Thus, a natural question is how to select the value of the detour constraint parameter α according to the current vehicle density, that is, the population of vehicles in the target road network.

According to the vehicle density, the value of the detour constraint parameter α can affect the performance of navigation. In light-traffic cases, $\alpha = 0.1$ gives the best performance in terms of average link delay, average maximum link delay, and average E2E delay. This is because these small α values allow the SAINT to work similarly as Dijkstra. On the other hand, in most traffic cases, except for light-traffic cases, $\alpha = 0.5$ gives the best performance. This means that when vehicles using the SAINT take 50% delay increase travel paths with minimum congestion increase for global traffic optimization, they can travel faster than vehicles using Dijkstra that lets vehicles navigate in a greedy way, leading to local traffic optimization. We will show the impact of the detour constraint parameter α on performance through simulations in Section VII-C. In the following section, we will explain the navigation procedure in SAINT.

VI. Self-Adaptive Interactive Navigation Tool Navigation Procedure

This section explains the navigation procedure in SAINT involving vehicles as navigators, RSUs (or eNodeBs), and the TCC in vehicular cloud. The navigation procedure is as follows.

1) As a SAINT Client, a vehicle with a navigator contacts the SAINT Server in the TCC for navigating from its source to its destination. The SAINT Client sends a navigation request to the SAINT Server by the V2I data delivery scheme, such as that proposed by Jeong *et al.* in [22] or 4G-LTE communications [8].
2) SAINT Server maintains a matrix (called congestion contribution matrix M) for a target road network graph to estimate the level of congestion per road segment in the graph.
3) With this matrix, the SAINT Server computes an optimal route for the SAINT Client to minimize the path congestion contribution from the source to the destination for global traffic optimization while bounding the detoured travel delay by parameter α.
4) SAINT Server gives a navigation response including an optimal route to SAINT Client for navigation. Data delivery from SAINT Server to SAINT Client is performed by the I2V data delivery scheme, such as that proposed by Jeong *et al.* in [24] or 4G-LTE communications [8]. This is possible because the TCC maintains the trajectory of the vehicle as SAINT Client for location management.
5) When receiving this route from the SAINT Server, the SAINT Client starts its travel along the guided route.
6) If the SAINT Client goes out of the guided route, it repeats steps 1 through 5 to get a new route from the SAINT Server.

So far, we have explained the design and procedure of our SAINT. In the following section, we will evaluate our SAINT with a state-of-the-art commercial navigation scheme.

VII. Performance Evaluation

This section evaluates the performance of *SAINT* in terms of *link delay* of all the vehicles for all the road segments and

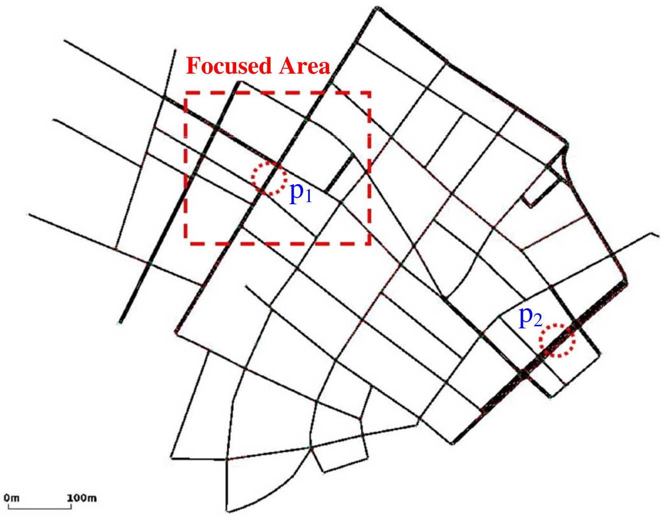


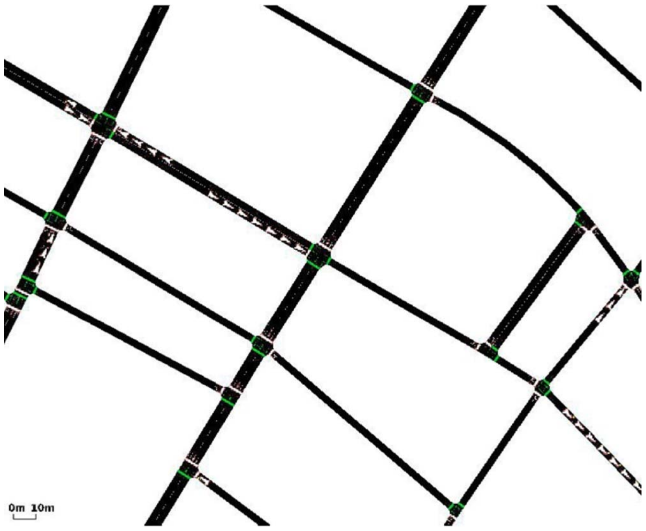
Fig. 7. Road network in Manhattan for simulation.

E2E travel delay of sample vehicles commuting between a pair of intersections as its source and destination positions. The evaluation setting is as follows.

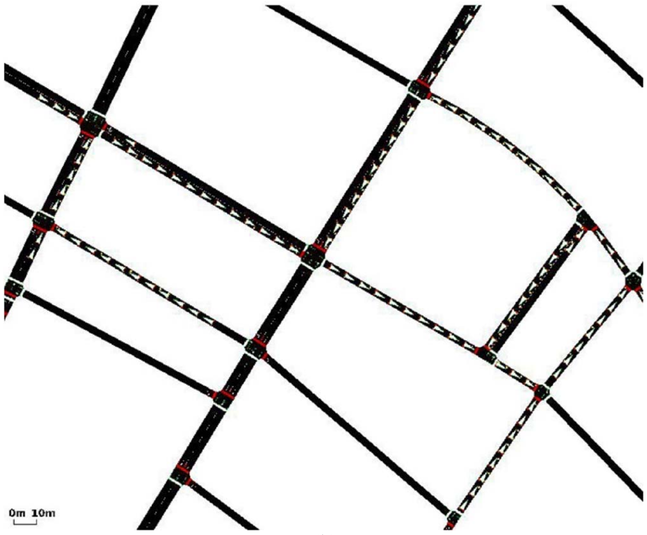
- **Performance Metrics**: We use 1) *average link delay*, 2) *average maximum link delay*, and 3) *average E2E travel delay* as metrics.
- **Baseline**: Since the state-of-the-art navigation schemes in the industry [1]–[4], [12]–[14] use Dijkstra's shortest path using real-time traffic statistics, we make a baseline scheme (called *Dijkstra*) to use Dijkstra's algorithm with real-time traffic statistics.
- **Parameters**: In the performance evaluation, we investigate the impacts of the following parameters: 1) *vehicular traffic density* N (i.e., number of vehicles); 2) *detour constraint parameter* α (i.e., additional delay ratio for the E2E shortest path delay); 3) *maximum vehicle speed* $v_{\max}$ (i.e., speed limit); 4) *vehicle acceleration* a_v (i.e., rate of vehicle speed change per unit time); and 5) *SAINT vehicle ratio* β (i.e., ratio of the number of vehicles following the SAINT scheme to the total number of vehicles).

We have built our *SAINT* and the baseline *Dijkstra* on top of a popular mobility simulator called Simulation of Urban MObility (SUMO) [31] with the following settings. For wireless communications, the SUMO is extended for the communications between vehicles and vehicular cloud. The TCC in the vehicular cloud maintains real-time traffic statistics and the trajectories of vehicles for the SAINT navigation. A road network of 99 intersections from Manhattan within New York City in the United States is used in the simulation. Note that Fig. 7 shows the road network for simulation. Fig. 8 shows road network snapshots for two navigation schemes, such as SAINT and Dijkstra.

Each vehicle's movement pattern is determined by the *car-following model* [32]. According to the *city section mobility model* [33], the vehicles are randomly placed at one intersection as *starting position* among the intersections on the road network and move toward another randomly selected intersection as *ending position*. The vehicles move realistically according to the *car-following model* along the roadway from their starting



(a)



(b)

Fig. 8. Road network snapshots for two navigation schemes. (a) Navigation by SAINT. (b) Navigation by Dijkstra.

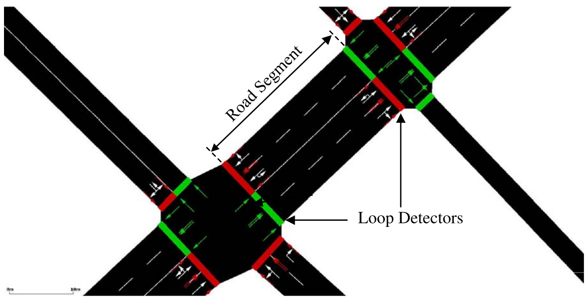
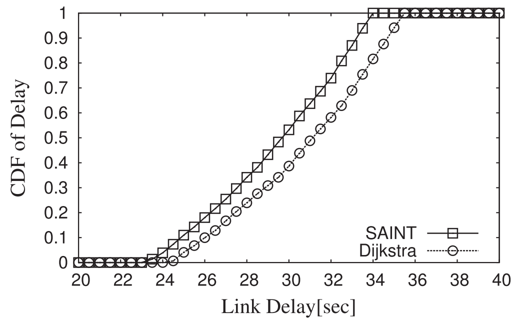


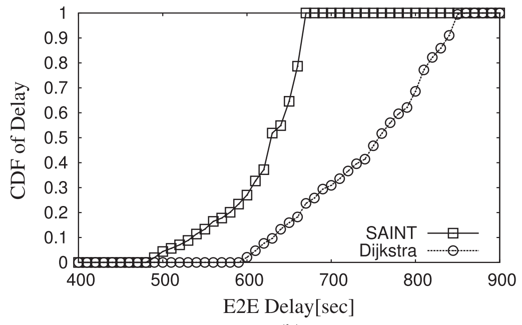
Fig. 9. Loop detector installation in the road network.

TABLE I
SIMULATION CONFIGURATION

Parameter	Description
Road network	Manhattan area in New York City in US with 99 intersections and the area of 875 m$\times$$950$ m (i.e., 0.54 miles$\times$$0.59$ miles).
Number of vehicles (N)	The number N of vehicles moving within the road network. The default of N is 300.
Delay increase threshold (α)	Delay increase threshold for DSP Algorithm. The default of α is 0.5.
Maximum vehicle speed (v_{max})	Maximum vehicle speed (i.e., speed limit) for road segments. The default is 40 km/h (i.e., 24.85 MPH).
Vehicle acceleration (a_v)	Vehicle acceleration per unit time for vehicles. The default is 3 m/s^2.



(a)



(b)

Fig. 10. Cdfs of navigation delays. (a) Link delay cdf. (b) E2E delay cdf.

position to their ending position. Moreover, the vehicles wait for green traffic light at intersections. Note that traffic light phases are determined by a *static traffic light scheduler* [31]. In our simulation, we measure the link delay including intersection waiting delay that is determined by traffic light scheduling (see Fig. 9). We install two detectors into the start point and end point of each road segment, respectively, as shown in Fig. 9. When a vehicle arrives at an edge, the start detector stores the vehicle's arrival time into its repository. The end detector stores the vehicle's departure time into its repository when the vehicle leaves the edge. With these timestamps, the link delay for the road segment is measured. Once the vehicle arrives at its ending position, it randomly selects another ending position for another travel. Thus, this vehicle travel process is repeated during the simulation time, based on both the car-following model and the city section mobility model. On the other hand, among the vehicles, 20 vehicles among 300 vehicles (i.e., 6.67%) are used as sample vehicles, commuting between two positions (denoted as p_1 and p_2 in Fig. 7) as their source and destination positions to measure E2E travel delay.

The vehicle speed is updated according to realistic vehicular mechanics based on the car-following model under the speed limit confined per road segment, as shown in Table I. For simplicity, we let all of the road segments have the same speed limit in the road network for the simulation; note that our design can easily extend this simulation setting to having the variety of

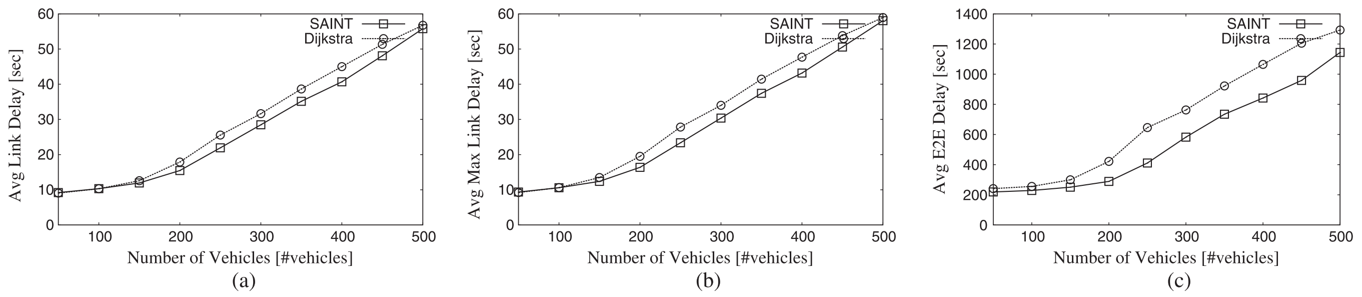


Fig. 11. Impact of vehicle number ($\alpha = 0.5$). (a) Link delay versus vehicle number. (b) Max link delay versus vehicle number. (c) E2E delay versus vehicle number.

speed limits for road segments. The simulation time is set to 2 h, and the simulations are repeated with different seeds (i.e., 10 or 30 seeds). The default values of the simulation are specified in Table I.

A. Navigation Behavior Comparison

This section compares the navigation behaviors of *SAINT* and *Dijkstra* with the cumulative distribution function (cdf) of E2E navigation delay. Fig. 10 shows the cdfs of *SAINT* and *Dijkstra* for the navigation delay in the road segment (i.e., link delay) and the navigation delay in the E2E travel path (i.e., E2E delay).

For link delay, as shown in Fig. 10(a), *SAINT* allows its cdf curve to be above *Dijkstra's* all the time, which means that *SAINT* has a higher cumulative distribution than Dijkstra at any given link delay in the horizontal line. As a result, *SAINT* allows its cdf to reach 1 at a much shorter delay (i.e., 34 s) than *Dijkstra*. In the case of *Dijkstra*, the cdf does not reach 1 even at the delay of 35.5 s.

For E2E delay, as shown in Fig. 10(b), *SAINT* allows its cdf curve to be equal to or above *Dijkstra's*. Moreover, *SAINT*'s cdf is increasing fast, reaching 1 at the delay of 670 s. On the other hand, *Dijkstra*'s cdf is slowly increasing, reaching 1 at the delay of 860 s, that is, *1.3 times* the *cdf completion point* of *SAINT* (i.e., 30% longer delay), which is the delay value with the cdf 1. Thus, *SAINT* has the E2E delay that is bounded within a narrow range of delay; however, *Dijkstra* has the E2E delay that is spread out within a larger range of delay.

Why does this performance difference happen between SAINT *and* Dijkstra? Fig. 8 explains this performance difference. As shown in Fig. 8(a), *SAINT* allows for **uniformly distributed traffic flow** to let vehicles bypass congested road segments with a high probability in the near future. On the other hand, *Dijkstra* allows vehicles to try to use road segments with light traffic at the moment of travel path computation in the *greedy per-vehicle basis* rather than the *road network's performance basis*. As a result, as shown in Fig. 8(b), some road segments are highly congested, but other road segments have almost zero traffic, leading to **partially congested traffic flow**. Therefore, by considering the overall road network performance, *SAINT* lets vehicles experience much shorter travel delay than *Dijkstra*.

B. Impact of Vehicle Number (N)

This section shows the impact of vehicle number on navigation performance. For link delay, max link delay, and E2E delay, as shown in Fig. 11(a)–(c), our *SAINT* outperforms *Dijkstra* in all the range of the number of vehicles. For light-traffic conditions from $N = 50$ to $N = 150$, the performance gap between *SAINT* and *Dijkstra* is small. On the other hand, for relatively heavy traffic conditions from $N = 200$ to $N = 500$, the performance gap between *SAINT* and *Dijkstra* is significantly large. In particular, in Fig. 11(c), for $N = 250$, *SAINT* has 64% of the E2E delay of *Dijkstra*. Even for the heaviest traffic condition of $N = 500$, *SAINT* has 89% of the E2E delay of *Dijkstra*, that is, reducing 11% of the E2E delay of *Dijkstra*. Thus, it can be seen that *SAINT* can support more effective navigation in heavy-traffic conditions, such as during rush hours.

C. Impact of Detour Constraint Parameter (α)

This section shows the impact of detour constraint parameter (α) on the performance along with the number of vehicles. In Section VII-B, a fixed value of $\alpha = 0.5$ was used. Note that the detour constraint parameter α determines the **delay bound** of a selected detour travel path to minimize the congestion contribution caused by the detour path in the road network. Thus, this means that each vehicle takes an α-increase travel path as a detour travel path whose travel delay is, at most, $(1 + \alpha)D_{uv}$, where D_{uv} is the travel delay of the time-wise shortest travel path p_{uv} from source position u to destination position v, as specified in Definition V.2. This section shows the best performance by changing the value of α from 0.1 to 1 by 0.1.

As shown in Fig. 12(a) and (b), *SAINT with α optimization* does not outperform *SAINT without α optimization* [as shown in Fig. 11(a) and (b)] in both the link delay and the max link delay. However, as shown in Fig. 12(c), for heavy-traffic conditions from $N = 300$ to $N = 500$, the performance improvement in E2E delay is significantly observable. In Fig. 12(c), for $N = 500$, *SAINT* with $\alpha = 0.6$ has 81% of the E2E delay of *Dijkstra*, that is, reducing 19% of the E2E delay of *Dijkstra*. Note that in Fig. 11(c), for $N = 500$, *SAINT* with $\alpha = 0.5$ has 89% of the E2E delay of *Dijkstra*, that is, reducing 11% of the E2E delay of *Dijkstra*. *SAINT* with $\alpha = 0.6$ can reduce 8% more for *Dijkstra* than *SAINT* with $\alpha = 0.5$. Thus, an optimal value selection of α can let *SAINT* achieve better performance. From our simulations, an optimal value of α is case by case for the vehicle number; hence, it is difficult to select an optimal value of α according to vehicle density. Fig. 13 shows that $\alpha = 0.6$

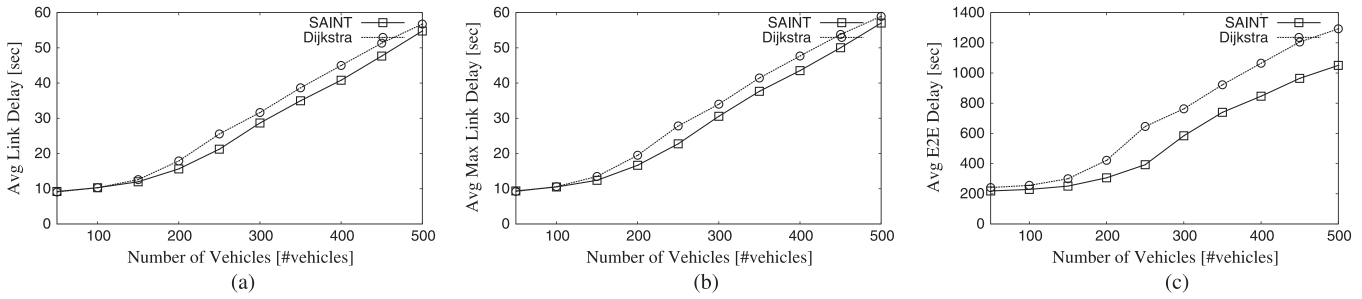


Fig. 12. Impact of optimal detour constraint parameter for vehicle number ($\alpha = 0.6$). (a) Link delay versus vehicle number. (b) Max link delay versus vehicle number. (c) E2E delay versus vehicle number.

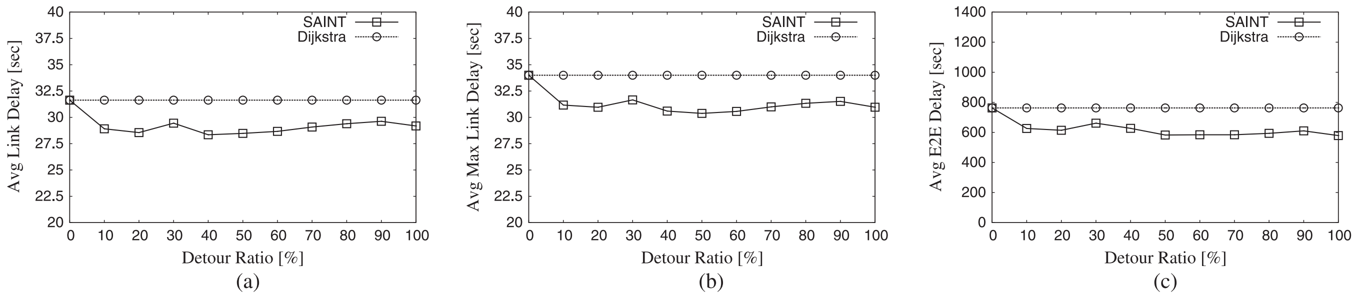


Fig. 13. Impact of detour constraint parameter. (a) Link delay versus detour ratio. (b) Max link delay versus detour ratio. (c) E2E delay versus detour ratio.

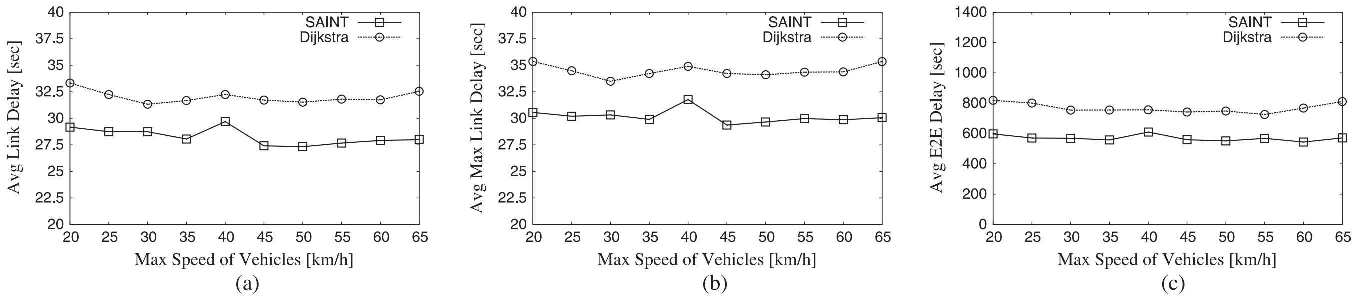


Fig. 14. Impact of max vehicle speed. (a) Link delay versus max vehicle speed. (b) Max link delay versus max vehicle speed. (c) E2E delay versus max vehicle speed.

achieves an overall good performance for three performance metrics in our simulation. However, since $\alpha = 0.5$ gives good performance for most vehicle numbers, we use 0.5 as the default of α for a given vehicle number rather than 0.6, which allows for good performance in our target road network, as shown in Fig. 7. The optimal value selection of α according to vehicle density is left as future work.

We can consider using different α values according to vehicles' travel distances. To relax the future congestion in a road area, a vehicle with a short travel distance uses a big α value, and another vehicle with a relatively long travel distance uses a small α value for fairness in terms of the travel distance. If these two vehicles use the same α value, the vehicle with a longer travel distance will take a longer detour than that with a shorter travel distance. This strategy requires a selection method of an optimal threshold to classify short travel distances and long travel distances. This threshold should consider both the short-term travel delay statistics (e.g., average for the last 10 min) and the current travel delay estimate (discussed in Section IV-B). The detailed investigation of the threshold selection and the α-value assignment to a given travel distance is left as future work.

D. Impact of Maximum Vehicle Speed $(v_{\max})$

This section shows the impact of maximum vehicle speed on the performance. Under a variety of speed limits from 20 to 65 km/h, *SAINT* allows vehicles to have better performance than *Dijkstra*, as shown in Fig. 14. In this figure, *SAINT* always has shorter travel than *Dijkstra* in the three metrics, such as link delay, max link delay, and E2E delay. From these results, we can conclude that *SAINT* can allow vehicles to travel faster than *Dijkstra* under any speed limit in road networks.

E. Impact of Vehicle Acceleration Profile (a_v)

This section shows the impact of the vehicle acceleration profile on the performance. Let a_v be the acceleration of each vehicle's speed. As shown in Fig. 15, for slowly accelerated

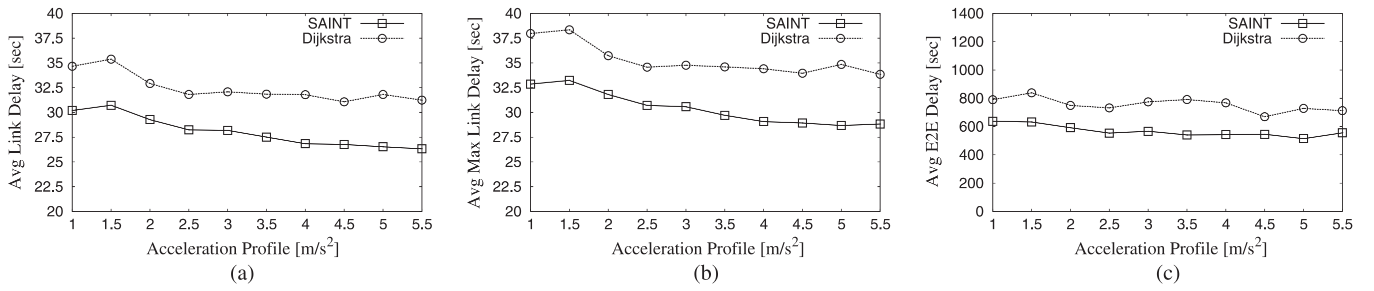


Fig. 15. Impact of acceleration profile. (a) Link delay versus acceleration profile. (b) Max link delay versus acceleration profile. (c) E2E delay versus acceleration profile.

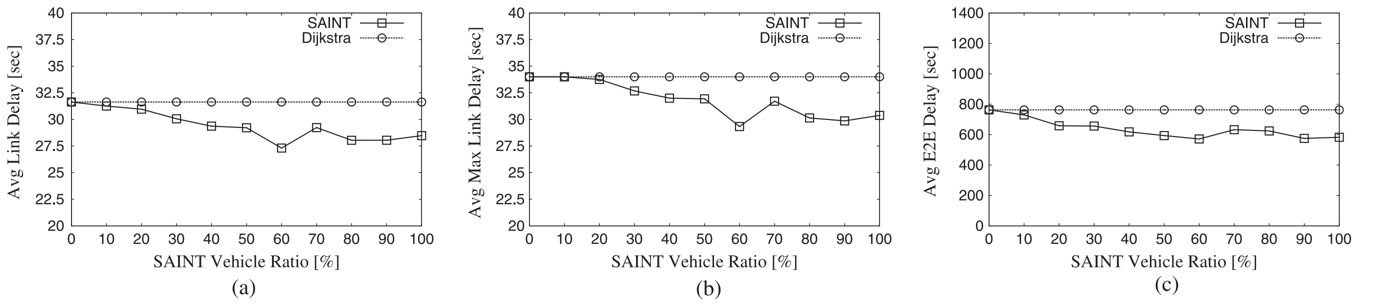


Fig. 16. Impact of SAINT vehicle ratio. (a) Link delay versus SAINT vehicle ratio. (b) Max link delay versus SAINT vehicle ratio. (c) E2E delay versus SAINT vehicle ratio.

vehicles from $a_v = 1.0$ m/s to $a_v = 2.0$ m/s, *SAINT* has, on average, 78% of the E2E delay of *Dijkstra*. For medium-accelerated vehicles from $a_v = 2.5$ m/s to $a_v = 4.0$ m/s, *SAINT* has, on average, 72% of the E2E delay of *Dijkstra*. Finally, for fast-accelerated vehicles from $a_v = 4.5$ m/s to $a_v = 5.5$ m/s, *SAINT* has, on average, 77% of the E2E delay of *Dijkstra*. From these results, it can be concluded that *SAINT* can allow vehicles to travel faster than *Dijkstra* under any acceleration profile in the target road network.

F. Impact of SAINT Vehicle Ratio (β)

This section shows the impact of *SAINT* vehicle ratio that is the percentage of vehicles using *SAINT* among the total vehicles in the target road network. We may raise the following question: *What happens if some vehicles use* SAINT *for their navigation and others use* Dijkstra *for their navigation?* This section answers this possible question.

Fig. 16 shows the impact of *SAINT* vehicle ratio, that is, the portion of vehicles using *SAINT* over the total number of vehicles. As shown in the figure, when at least 30% of vehicles use *SAINT*, the performance of *SAINT* is significantly better in all the three metrics than that of *Dijkstra*. This figure shows that *SAINT* performs ever better as the *SAINT* vehicle ratio increases. Therefore, even partial deployment of *SAINT* will be effective in real navigation scenarios during market penetration.

G. Impact of Nonlinear Formulation of Link Congestion Contribution

This section shows the impact of a *nonlinear formulation* for link congestion contribution on performance. We replace the linear formula in (3) with a nonlinear formula as follows:

$$c_i = 1 - \exp\left(-\frac{D_i}{D}\right) \tag{4}$$

where D_i is the travel time for the subpath from the starting intersection 1 to an intermediate intersection i along the vehicle trajectory, and D is the E2E travel time along the vehicle trajectory.

Fig. 17 shows that in most traffic conditions from $N = 50$ to $N = 500$, the performance gap between *SAINT* and *Dijkstra* is smaller than the performance gap between *SAINT* and *Dijkstra* in the case of the linear formulation, as shown in Fig. 11. From this result, it is seen that a simple nonlinear formulation for link congestion contribution is not more effective than the linear formulation. The investigation for a good nonlinear formula for congestion contribution is left as future work.

H. Impact of Dynamic Update of Link Congestion Contribution

This section shows the impact of the *dynamic update* of link congestion contributions (called *dynamic version*) for the remaining edges on the performance, while a vehicle arrives at each intersection along the vehicle trajectory. Note that the original *SAINT* lets a vehicle have a trajectory with static link congestion contributions at the beginning of navigation. That is, it lets the link congestion contributions for the edges on the trajectory be not updated during the vehicle travel along its trajectory, as discussed in Section V. This management of link congestion contributions is called the *static link congestion contribution method* (shortly *static version*).

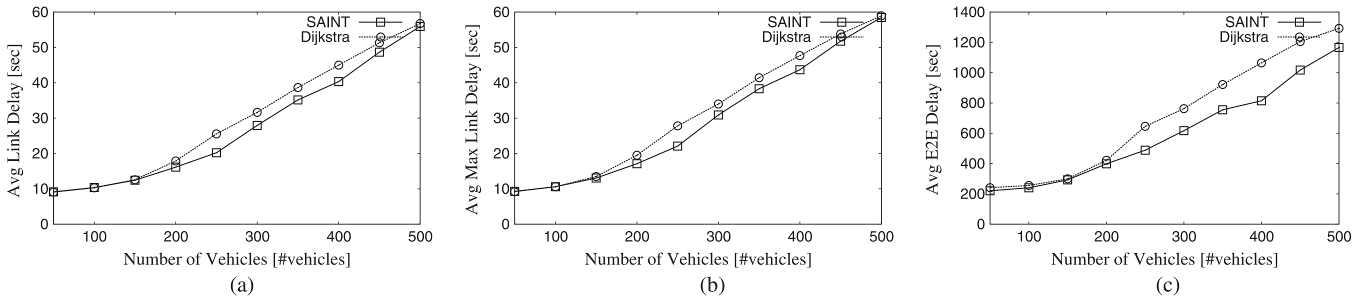


Fig. 17. Impact of nonlinear congestion contributions on vehicle number. (a) Link delay versus vehicle number. (b) Max link delay versus vehicle number. (c) E2E delay versus vehicle number.

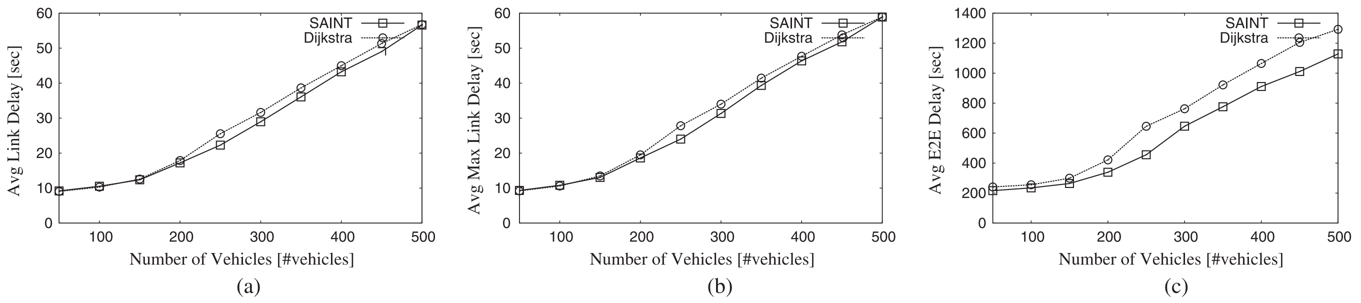


Fig. 18. Impact of dynamic update of link congestion contribution on vehicle number. (a) Link delay versus vehicle number. (b) Max link delay versus vehicle number. (c) E2E delay versus vehicle number.

As shown in Fig. 18, except for the lowest density $(N = 50)$ and the highest density $(N = 500)$, in most cases of density, the dynamic version has worse performance than the static version. This is because the dynamic version cannot provide the road capacity reservation consistently to avoid the traffic congestion better than the static version.

So far, we have showed that *SAINT* significantly outperforms *Dijkstra* under a variety of settings (i.e., vehicular traffic density, speed limit, and *SAINT* vehicle ratio) in the target road network for the performance metrics (i.e., link delay, max link delay, and E2E delay). Therefore, we conclude that *SAINT* will be a good foundation to enhance the cloud-based navigation in the future.

VIII. Conclusion

This paper has proposed the SAINT for cloud-based vehicular traffic optimization in road networks. With the increasing popularity of GPS-based navigation systems and mobile networks (e.g., DSRC and 4G-LTE communications), we believe that our SAINT will improve the traffic flow in road networks by providing vehicles with globally optimal navigation paths in the viewpoint of the *transportation network* rather than *individual vehicles* through vehicular cloud. This paper suggests a new metric called *congestion contribution* to estimate the near-future congestion level of each road segment. With this *congestion contribution metric*, our SAINT will set up a navigation path for each vehicle for global road traffic optimization. As future work, we will enhance the modeling of link congestion contribution by considering road capacity in terms of the number of lanes for better traffic optimization. Moreover, we will extend our SAINT for accident scenarios so that emergency vehicles (e.g., ambulance, police car, and fire engine) can reach an accident spot quickly, and other vehicles around the accident spot can navigate quickly toward their destination. In addition, we will investigate how to combine our SAINT with traffic light control systems so that traffic light scheduling at intersections can be dynamically adapted for better vehicular traffic optimization in a self-adaptive way.

References

[1] Garmin, Dedicated Navigator. [Online]. Available: http://www.garmin.com

[2] TomTom, Dedicated Navigator. [Online]. Available: http://www.tomtom.com

[3] Waze, Smartphone App for Navigator. [Online]. Available: https://www.waze.com

[4] Navfree, Free GPS Navigation for Android Smartphone. [Online]. Available: http://navfree.android.informer.com

[5] Q. Xu, R. Sengupta, and D. Jiang, "Design and analysis of highway safety communication protocol in 5.9 GHz dedicated short range communication spectrum," in *Proc. IEEE VTC Spring*, Apr. 2003, pp. 2451–2455.

[6] J. Zhao and G. Cao, "VADD: Vehicle-assisted data delivery in vehicular ad hoc networks," *IEEE Trans. Veh. Technol.*, vol. 57, no. 3, pp. 1910–1922, May 2008.

[7] Y. L. Morgan, "Notes on DSRC & WAVE standards suite: Its architecture, design, and characteristics," *IEEE Commun. Surveys Tuts.*, vol. 12, no. 4, pp. 504–518, 4th Quart. 2010.

[8] R. Kirui, "Accessing cloud computing resources over 4G LTE," Helsinki Metropolia Univ. Appl. Sci., Espoo, Finland, Tech. Rep., May 2014.

[9] C. D. Monfreid, "The LTE network architecture—A comprehensive tutorial," Alcatel-Lucent, Tech. Rep., 2009. [Online]. Available: http://www.cse.unt.edu/~rdantu/FALL_2013_WIRELESS_NETWORKS/LTE_Alcatel_White_Paper.pdf

[10] Traffic Control Center in Philadelphia Department of Transportation. [Online]. Available: http://philadelphia.pahighways.com/philadelphiatcc.html

[11] A. Abdrabou and W. Zhuang, "Probabilistic delay control and road side unit placement for vehicular ad hoc networks with disrupted connectivity," *IEEE J. Sel. Areas Commun.*, vol. 29, no. 1, pp. 129–139, Jan. 2011.

[12] Skobbler, Smartphone App for GPS Navigation and Maps. [Online]. Available: http://www.skobbler.com/apps/navigation/android

[13] Tmap, SKT Smartphone Navigator. [Online]. Available: http://www.tmap.co.kr/tmap2

[14] iNAVI, iNAVI Navigation System. [Online]. Available: http://www.inavi.com

[15] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. Cambridge, MA, USA: MIT Press, 2009.

[16] M. Wang *et al.*, "Real-time path planning based on hybrid-VANET-enhanced transportation system," *IEEE Trans. Veh. Technol.*, vol. 64, no. 5, pp. 1664–1678, May 2015.

[17] A. H. Khosroshahi, P. Keshavarzi, Z. D. KoozehKanani, and J. Sobhi, "Adaptive demand-driven multicast routing in multi-hop wireless ad hoc networks," in *Proc. IEEE Int. Conf. Comput., Control Ind. Eng.*, Wuhan, China, pp. 33–44, Aug. 2011.

[18] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 2nd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2002.

[19] C. Perkins, D. Johnson, and J. Arkko, Mobility support in IPv6, RFC6275, Jul. 2011.

[20] ETSI, DSRC Standardization. [Online]. Available: http://www.etsi.org/WebSite/Technologies/DSRC.aspx

[21] J. Jeong, T. He, and D. H. Du, "TMA: Trajectory-based Multi-anycast for multicast data delivery in vehicular networks," *Comput. Netw.*, vol. 57, no. 13, pp. 2549–2563, May 2013.

[22] J. Jeong, S. Guo, Y. Gu, T. He, and D. Du, "Trajectory-based data forwarding for light-traffic vehicular ad-hoc networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 5, pp. 743–757, May 2011.

[23] Y. Ding and L. Xiao, "SADV: Static-node-assisted adaptive data dissemination in vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 59, no. 5, pp. 2445–2455, Jun. 2010.

[24] J. Jeong, S. Guo, Y. Gu, T. He, and D. Du, "Trajectory-based statistical forwarding for multihop infrastructure-to-vehicle data delivery," *IEEE Trans. Mobile Comput.*, vol. 11, no. 10, pp. 1523–1537, Oct. 2012.

[25] J. Jeong and E. Lee, "Vehicular cyber-physical systems for smart road networks," *KICS Inf. Commun. Mag.*, vol. 31, no. 3, pp. 103–116, Mar. 2014.

[26] A. Polus, "A study of travel time and reliability on arterial routes," *Transportation*, vol. 8, no. 2, pp. 141–151, Jun. 1979.

[27] M. DeGroot and M. Schervish, *Probability and Statistics*, 3rd ed. Reading, MA, USA: Addison-Wesley, 2001.

[28] G. Dimitrakopoulos and P. Demestichas, "Intelligent transportation systems based on cognitive networking principles," *IEEE Veh. Technol. Mag.*, vol. 5, no. 1, pp. 77–84, Mar. 2010.

[29] Res. Innovative Technol. Admin. (RITA), IntelliDrive: Safer, Smarter and Greener. [Online]. Available: http://trid.trb.org/view.aspx?id=968739

[30] J. Y. Yen, "Finding the K shortest loopless paths in a network," *Manage. Sci.*, vol. 17, no. 11, pp. 712–716, Jul. 1971.

[31] SUMO, Simulation of Urban MObility. [Online]. Available: http://sumo-sim.org

[32] R. W. Rothery, "Car-following models," Transp. Res. Board, Washington, DC, USA, Revised Monograph on Traffic Flow Theory, Tech. Rep., 1997, [Online]. Available: http://www.tfhrc.gov/its/tft/chap4.pdf

[33] T. Camp, J. Boleng, and V. Davies, "A survey of mobility models for ad hoc network research," *Wireless Commun. Mobility Comput.*, vol. 2, no. 5, pp. 483–502, Aug. 2002.

Jaehoon (Paul) Jeong (M'09) received the B.S. degree from the Department of Information Engineering, Sungkyunkwan University, Suwon, Korea, in 1999; the M.S. degree from the School of Computer Science and Engineering, Seoul National University, in 2001; and the Ph.D. degree from the Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN, USA, in 2009.

He is an Assistant Professor with the Department of Software, Sungkyunkwan University. His research areas include vehicular networks, wireless sensor networks, and mobile ad hoc networks.

Dr. Jeong's two data-forwarding schemes (called TBD and TSF) for vehicular networks were selected as spotlight papers in the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS in 2011 and the IEEE TRANSACTIONS ON MOBILE COMPUTING in 2012, respectively. He is a member of the Association for Computing Machinery and the IEEE Computer Society.

Hohyeon Jeong (S'15) received the B.S. and M.S. degrees from the School of Electrical and Computer Engineering, Ajou University, Suwon, Korea, in 2012 and 2014, respectively. He is currently working toward the Ph.D. degree with the Department of Computer Science and Engineering, Sungkyunkwan University, Suwon, Korea.

His research area is self-adaptive software systems.

Eunseok Lee (M'15) received the B.S. degree from the Department of Electronic Engineering, Sungkyunkwan University, Suwon, Korea, in 1985 and the M.S. and Ph.D. degrees from the Department of Information Engineering, Tohoku University, Sendai, Japan, in 1988 and 1991, respectively.

He is a Professor with the Department of Computer Engineering, Sungkyunkwan University. He was an Assistant Professor with Tohoku University. Before that, he was a Research Scientist with Mitsubishi Electric Corporation. His research areas are self-adaptive software systems, software testing, and autonomic computing.

Tae (Tom) Oh (SM'15) received the B.S. degree from Texas Tech University, Lubbock, TX, USA, in 1991 and the M.S. and Ph.D. degrees from Southern Methodist University, Dallas, TX, USA, in 1995 and 2001, respectively, all in electrical engineering.

He obtained his degrees while working for telecommunication and defense companies. He is an Associate Professor with the Department of Information Sciences and Technologies and the Department of Computing Security, Rochester Institute of Technology, Rochester, NY, USA. He has published numerous technical articles. His research interests include mobile ad hoc networks, vehicle area networks, sensor networks, and mobile device security.

Dr. Oh is a Technical Editor of the IEEE COMMUNICATIONS MAGAZINE and an Associate Technical Editor of the IEEE NETWORKS. He is a member of the Association for Computing Machinery.

David H. C. Du (F'98) received the B.S. degree in mathematics from National Tsing Hua University, Hsinchu, Taiwan, in 1974 and the M.S. and Ph.D. degrees in computer science from the University of Washington, Seattle, WA, USA, in 1980 and 1981, respectively.

He is currently the Qwest Chair Professor with the Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN, USA. His research interests include cyber security, sensor networks, multimedia computing, storage systems, and high-speed networking.