Borokhov, P., Blandin, S., Samaranayake, S., Goldschmidt, and Bayen,A., "An Adaptive Routing System for Location-Aware Mobile Devices on the Road Network," In IEEE Conference on Intelligent Transportation Systems (ITSC), Washington, DC, pp. 1839-1845, October 2011, doi:10.1109/ITSC.2011.6083021.

An adaptive routing system for location-aware mobile devices on the road network

Paul Borokhov, Sebastien Blandin, Samitha Samaranayake, Olivier Goldschmidt, and Alexandre Bayen

***Abstract*— As congestion problems become a greater concern and negatively impact society, solutions which alleviate them are needed to improve the performance of the transportation system. Routing systems which take into account the travel-time experienced by the driver have been largely unexplored in the domain of adaptive routing. In this article, we present a system which enables users of smartphones to obtain directions generated using an algorithm which provides an optimal routing policy for *reliable on-time arrival*; that is, directions which seek to maximize the probability of arriving to the destination within a given time budget, rather than to minimize the travel time based on posted speed limits. Our work leverages the geolocation capabilities of smartphones to provide optimal routing directions along the route dependent on the realized (experienced) travel time. The adaptive routing scheme we implement allows for significant power savings and improved driver safety compared to classical routing algorithms; special attention is paid to minimizing driver distraction by emphasizing aural and graphical components over textual elements during route guidance. Finally, we illustrate system performance and design choices on synthetic examples and real traffic data from the *Mobile Millennium* system in San Francisco.**

I. INTRODUCTION

For the past decade, congestion throughout the United States has become an increasing problem. While in recent years, the average percentage of vehicle-miles traveled under congested conditions, as well as the resulting additional time needed to complete a trip, has not changed significantly, the number of annual hours of delay per capita has continued to grow. Annually, the average American is delayed by nearly 22 hours as a result of congestion [1]. Solutions which provide an opportunity to improve these conditions, reducing the amount of delay drivers experience, can potentially facilitate better commuting experiences and reduce the negative impacts on society arising from congestion.

Historically, traffic monitoring systems have been used nationwide by transportation agencies for formulating policy and aiding planning measures, as well as research, but the data itself, and consequently traffic information, was not easily accessible to the general public. Traffic information was generally obtained through the radio, which in turn obtained its information using traffic helicopters and firsthand accounts. Seasoned commuters knew a number of routes between which they could choose depending on traffic conditions, but such measures were always reactive and only became available after significant experience traveling on a particular route. Such regimens changed dramatically, however, with the widespread availability of web-enabled smartphones and portable GPS navigation systems with real-time traffic information and rerouting capabilities.

Smartphone penetration among mobile phone users has continually increased in recent years, especially with the introduction of Apple's iPhone in 2007 [2], reaching US market share rates of nearly 30% in late 2010 [3]. Most of these devices feature built-in GPS receivers, which make them extremely suitable to routing and navigation applications.

However, the navigation solutions bundled with these phones do not actively or continually use the user's current location, available as a result of these GPS capabilities. For example, Apple's iOS (including the most recent release, 4.3.5) does not provide dynamic route updates once directions have been calculated. Google's Navigation Beta for Android provides location-adaptive routing but is only available for the newest devices, with the most recent version attempting to route users around current bottlenecks. While some commercial products, such as TomTom, take traffic conditions into consideration when routing drivers, their relatively high upfront costs, along with uncertain benefits over free alternatives, reduce their adoption rates. Furthermore, none of these systems seem to take into account the general stochastic nature of travel-times arising out of congestion as well as different drivers exhibiting different driving styles. Such systems could potentially require recomputation in the case of unexpected individual events, such as flat tires, which in a stochastic setting are already accounted for as congestion.

An important issue when designing software for mobile devices is power consumption. Users cannot be reasonably expected to change their charging behavior to suit the power consumption needs of a particular application, unless the user clearly expects a significant battery drain to occur (such as when playing a graphically detailed game for an extended amount of time). Therefore, any alternative navigation solutions must be cognizant of power consumption issues and seek to minimize the amount of power they need for operation, especially given that a navigation application will be used for the entire duration of a route.

In this work, we describe an algorithm and introduce a novel system design which seeks to address the aforementioned issues currently present in mobile device nav-

P. Borokhov is the corresponding author and can be reached at pborokhov@berkeley.edu

P. Borokhov, S. Blandin, S. Samaranayake are with Department of Civil and Environmental Engineering; A. Bayen is with the Department of Electrical Engineering and Computer Sciences and the Department of Civil and Environmental Engineering, University of California Berkeley, Berkeley, CA 94720, USA

O. Goldschmidt works at OPNET Technologies, Bethesda, MD 20814, USA

igation systems. The *Stochastic On-Time Arrival (SOTA) algorithm* [4] computes time-varying routing policies between an origin and destination given a travel-time budget. A routing policy provides directions to the user which depend on the experienced travel time up to the current location in the network, in contrast to static *a priori* solutions that determine the entire path before departure. Because SOTA policies are downloaded once and contain information for the entire trip, they allow us to reduce power consumption while providing adaptive and time-varying directions to the user. Push-based policy recalculation maintains the relevancy of the SOTA policies in changing traffic conditions in a power-conserving manner. Simple, aural directions reduce driver distraction; their simplicity also eliminates the need for a text-to-speech engine and consequently results in lower battery drain. In our implementation, historical travel-time distributions used by the SOTA algorithm are obtained from the *Mobile Millennium* system (`http://traffic.berkeley.edu`), which fuses millions of datapoints from multiple sources of probe data to obtain link travel-time distributions. The utility of our mobile application allows us to use it as a sensor to provide one additional source of input data to this system.

This article is organized as follows. Section II describes the SOTA algorithm. Section III provides an overview of the routing system, including a description of the server and client communications. Section IV describes the mobile phone client in detail, including design decisions made specifically for mobile devices.

II. STOCHASTIC ON-TIME ARRIVAL PROBLEM

Link travel-times on road networks exhibit variability patterns due to individual driving behaviors, varying demands and exogenous factors. In the context of traffic information systems, the network of roadways can be modeled as a directed graph with stochastic link travel-times. For typical trips, commuters are mainly interested in a path which minimizes the expected travel-time between their origin and their destination. This *least expected time* (LET) path is usually the route recommended by classical routing directions providers. While these routes are suitable for many situations, a significant number of practical scenarios require reliability guarantees that LET paths do not provide. For example, a user driving to the airport is interested in the path that maximizes his probability of catching his flight. In this case, a route with higher reliability would be preferred over a route with a lower expected travel-time but lower reliability.

One of the first formulations of the shortest path problem which incorporates reliability guarantees was introduced by Frank [5], who defined the optimal routing strategy as the strategy that maximizes the probability of realizing a travel-time that is less than a given time constant. This problem, known as the *stochastic on-time arrival* (SOTA) problem, implicitly defines the optimality criterion considered by our routing system, as described next.

A. Problem statement

Consider a directed graph $G(N, A)$ with $|N| = n$ nodes and $|A| = m$ links. The weight of each link $(i, j) \in A$ is a random variable t_{ij} with probability density function $p_{ij}(\cdot)$ that represents the travel-time on link (i, j). Given a time budget T, an optimal routing strategy is defined to be a policy that maximizes the probability of arriving at a destination node s within time T. Given a node $i \in N$ and a time budget t, $u_i(t)$ denotes the probability of reaching node s from node i in less than time t when following the optimal policy. At each node i, the traveler is advised to pick the link (i, j) that maximizes the probability of arriving on time at the destination. If j is the next node being visited after node i and ω is the time spent on link (i, j), the traveler starting at node i with a time budget t has a time budget of $t - \omega$ to travel from j to the destination, as described in equation (1).

Definition 1: The optimal routing policy for the SOTA problem is defined as follows:

$$\begin{aligned} u_i(t) &= \max_j \int_0^t p_{ij}(\omega)\, u_j(t-\omega)\, d\omega \\ &\qquad \forall i \in N,\ \ i \neq s,\ \ (i,j) \in A,\ \ 0 \le t \le T \qquad (1) \\ u_s(t) &= 1 \qquad 0 \le t \le T \end{aligned}$$

where $p_{ij}(\cdot)$ is the travel-time distribution on link (i, j). In the context of traffic information systems, the link travel-time distribution functions $p_{ij}(\cdot)$ are assumed to be known and can for example be obtained using historical distributions or real-time commuter information. Section II-C describes the travel-time distributions used in our application.

B. Solution algorithm

The general solution to the *a priori* SOTA problem does not satisfy Bellman principle of optimality [6] and does not yield efficient solutions [7] [8] [9].

However, under its adaptive form, the problem can be approached as a stochastic optimal control problem that produces an optimal *policy* as opposed to an optimal *path*. An optimal policy is an adaptive node-based decision rule that defines the optimal path from a given node to the destination conditioned on the realized travel-time. Instead of being assigned a fixed travel route when leaving the departure point, the commuter is adaptively guided across the stochastic network given the realized travel-time at each intermediate point.

The stochastic optimal control problem has been solved using successive approximations [10] and discrete convolutions [11], but these solution methods are too computationally intensive for practical real-time routing. A recent solution presented in [4] provides a more tractable solution using the *Fast Fourier Transform* (FFT) to compute the convolutions and a preprocessing algorithm designed to minimize the computational complexity of computing them. This algorithm leverages the existence of a minimal link travel-time on the road network to prove the existence of a lower-bound on the computation time of the optimal policy. The routing algorithm used by our mobile application is the first operational implementation of the provably optimal SOTA algorithm introduced in [4].

C. Travel-time distributions

The SOTA algorithm is agnostic to the estimation methods producing the link travel-time distributions. In its current form, the system described in this article uses travel-time distributions obtained from the *Mobile Millennium* traffic information system.

Mobile Millennium is a research project launched in November 2008 at the University of California, Berkeley, jointly with Nokia and NAVTEQ. The system collects millions of raw traffic data points from loop detectors, radars, taxi fleets, and commuters' GPS devices daily and estimates link travel-time distributions at a 30-second granularity on the arterial and highway network of Northern California in real time. Specific estimation methods are used for each network: highway and arterial.

On highways, following classical macroscopic traffic flow theory, traffic dynamics are modeled by the *Lighthill-Whitham-Richards* partial differential equation [12] [13]. Due to the nonlinearity and non-differentiability of the dynamics, real-time estimation is achieved with an ensemble Kalman filtering method [14] on the PDE expressed in the velocity variable (see [15] for more details). Every 30 seconds, the most likely estimate given the model is combined with collected measurements from the road network to produce the variance-minimizing estimate of the current traffic state given the model and the real-time observations.

On arterial roads, the sparsity of real-time data combined with the complexity of driving behaviors makes this approach unfeasible. A purely statistical model of link travel-time described in [16] is used to learn the traffic patterns from historical knowledge and streaming data. A significant difficulty on arterial roads is the amount of measurement noise in GPS data points. A dedicated algorithm [17] [18] based on a Hidden Markov Model (HMM) is used to provide accurate map-matched GPS measurements which can then be fed to the statistical model.

The complete interaction of the estimation algorithms producing the travel-time distributions, and the routing system introduced in this article, is presented in the following section.

III. SYSTEM OVERVIEW

The complete *routing system*, presented in Figure 1, consists of a web server for accepting user routing requests and sending the routing policies, a routing server for running the SOTA algorithm, a push server for notifying clients about significant policy changes, and a mobile phone client which communicates with these servers. The mobile client provides an interface to specify the origin and destination, as well as a time budget. It also serves as a traffic sensor, collecting location and speed data as the user progresses along her route and sending this data back to the *Mobile Millennium* system for improving estimates of travel-time distributions; this is described in more detail in section IV-D.

One distinguishing feature of the mobile client application is the ability to specify a desired travel (time) budget, allowing the SOTA algorithm to calculate the optimal policy for arriving at the destination on time with the highest probability.

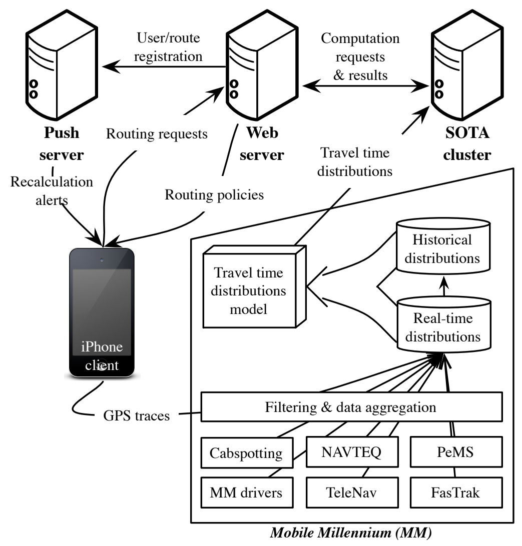


Fig. 1: **SOTA routing system**: the *web server* and *SOTA cluster* are described in section III-A, the *push server* in section III-B, and the client in section III-C.

Once the trip begins, the budget is also used to determine how much time remains until the driver must reach his destination, and therefore which policy decision to follow at any given intersection. The maximum probability of completing the trip in the specified budget is also provided to the user, a reliability metric that is not common in contemporary navigation systems but could be very useful to travelers.

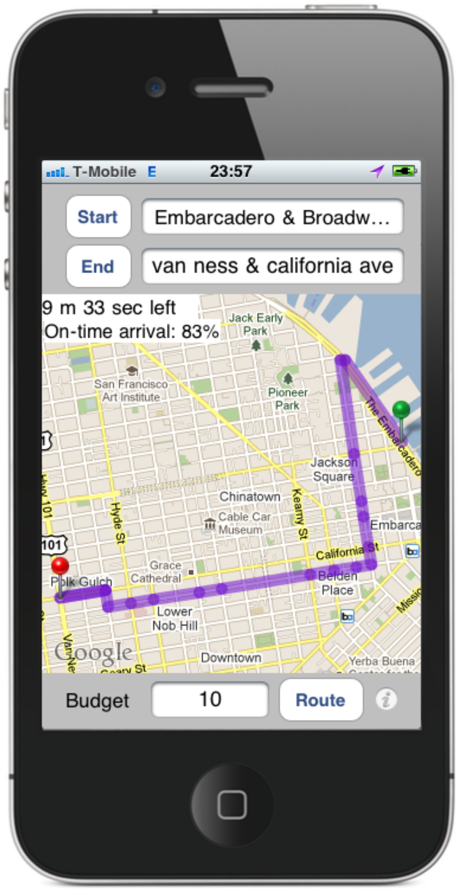
We describe the server components below; the mobile phone client is described in greater detail in Section IV.

A. Routing and web server

An Apache Tomcat server hosting a self-contained *Web application ARchive* (WAR) file receives application requests for routing policies. A Java Servlet handles HTTP requests and requests policies from the SOTA computational component. All communication between the application and web server takes place over standard HTTP protocols, automatically guaranteeing that connections between the client and server are reliable.

B. Push server

The push server, described in detail in section IV-B, maintains the relevance of downloaded policies in situations in which the real-time travel-time distributions on the network change significantly to the extent that the originally downloaded policies are no longer valid. This is accomplished as follows: first, the server monitors which areas of the network have recently had significant changes in their travel-time distributions; then, it maintains a list of which mobile clients are currently en-route and need to be alerted of these changes. This component is almost completely independent from the routing server: the sole dependency is that the routing server needs to inform the push server when a client has initiated a route. The push server will then register the client in its list of current drivers and keep it there until the client has completed



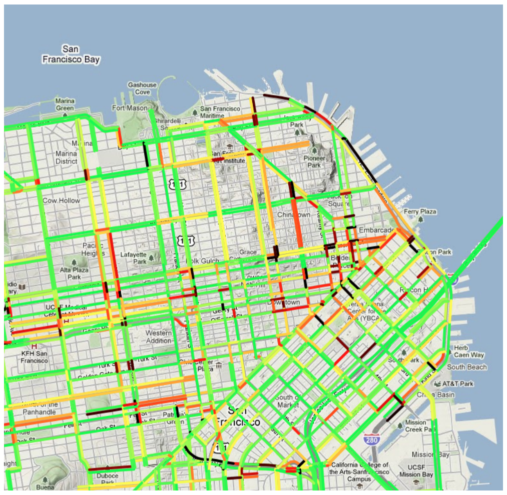


Fig. 2: **Top:** Screenshot of a sample route, as shown initially before guidance commences, through San Francisco. Once guidance begins, the map zooms to show the next turn and the rest of route updates automatically as conditions change. **Bottom:** A sample of the arterial distributions in the *Mobile Millennium* system for the arterial network in San Francisco used to the generate the route on the left.

the route or the budget time has elapsed, whichever comes first. Communication with the push server is mediated by Apple and is not guaranteed to be successful at all times [19].

The push server's relative independence allows it to scale independently of the other components. Furthermore, the push server can go down completely without significantly affecting the rest of the system. Finally, its existence allows for significant power savings on the mobile device while still providing a near-real-time policy experience to users, as described later.

C. Mobile phone client

Traditional navigation products primarily provide textual directions along with a visual representation of the route on the map. The nature of the optimal SOTA policy - namely, that at any given time, the only "direction" known is the one for where to go at the next *intersection* - means that providing textual directions might be futile, confusing, and potentially dangerous (as users will keep trying to read the direction text as it changes). At the same time, due to the significant amount of existing experience most users will have with traditional navigational software, some sample route must be drawn to give the driver an idea of where they should go for the next few blocks. As a result, our application uses a hybrid approach which eliminates textual directions entirely while still providing usable and understandable navigational guidance.

- **Graphical path**: to provide a visual cue of the potential route to the driver, we plot the most likely path given that *mean travel-time* is achieved over *all* the road links of the route, or in essence, the *expected path*. However, we believe that users will mostly be focused on driving, and therefore this visual representation is always an approximation beyond the next road link.
- **Aural directions**: for directions, we provide *simplified* audio instructions, using standard English recordings of the direction in which the driver should go at the next link junction. Since the roadway network used by the algorithm does not count every road intersection as a link junction, we also announce the ordinality of the intersection to which the direction applies. For example, the application could announce *Continue straight at the third intersection*. When a road link corresponds to a single street block, the directions would read, for example, *Turn right at the next intersection*.

This model accomplishes a number of seemingly divergent goals simultaneously.

- Its computational requirements are adapted for mobile devices because all the sound samples are prerecorded, small in file size, and downloaded once with the application bundle, requiring minimal computational power (and thus battery expenditure) to play back.
- It is appropriate for the algorithm used, because it does not confuse the user with a set of textual directions which have the potential to continually change as the driver progresses through his route.
- It is highly fitting for the context in which it is being used, since simple directions like *turn right at the second*

intersection and *make a sharp left turn at the next intersection* impose significantly lower cognitive loads on the user and do not require him to distract himself by attempting to read signs indicating the names of cross-streets [20].

Thus, our solution has the potential to be safer than existing navigational products, while also providing power savings (described next) and improved on-time arrival reliability.

IV. Mobile Device Adaptations

Our implementation attempts to fully exploit the smartphone platform while at the same time being subject to its constraints. The application takes advantage of location awareness, a ubiquitous internet connection, and the phone's capability as a sensing platform while subject to a limited amount of available power, a limited user attention span, and a need to preserve user privacy. Some of our solutions to the constraints imposed by a mobile platform arise naturally due to the output that the SOTA algorithm generates. Others are specific design decisions we have made in architecting the system and the mobile phone client.

In achieving reductions in power consumption, previous work has focused on minimizing usage of the GPS radio [21]. However, we believe that providing true GPS location to the driver results in a better quality of service and therefore leave GPS on for the entire length of navigational guidance. Our algorithm, and the design of the system as a whole, provides a number of different approaches to reducing power consumption, as detailed below.

A. Time- and location-adaptive features

As the user progresses through the route, the application keeps track of the current position on the route and monitors whether the decision rule output for the upcoming node, which had been chosen previously while generating the *expected path* (as described in section III-C), differs from the decision rule's current output. This procedure does not require any additional network communication on the part of the application. Once a policy has been downloaded to the phone, it contains all the information necessary to navigate to the destination with any time remaining from any intersection in the network reachable from the origin in the specified budget. While this results in a larger policy size, and thus a larger download, we believe it is offset by the reduced network communication during the route. As time elapses and the remaining time budget is reduced, the only operation the application has to perform is a lookup in a local hashtable, which stores a representation of the policy, and then announce its output to the user. Furthermore, the user is able to see the expected time remaining, the budgeted time remaining, and the likelihood of arriving on time to the destination. This is in contrast to the native iOS solution, which requires manual route recalculation to determine the time needed to reach the destination and provides no on-time arrival probabilities.

B. Push-based policy recalculation

In most cases, the application does not need to request a new policy after its initial download, since policies account for time-varying traffic patterns. This holds as long as travel-time distributions on the network are sufficiently similar to those used in computing the original policy. However, it is possible that some network event takes place along the user's route which results in travel-times which are significantly different from the distributions used by the algorithm in generating its policy. For example, the policies generated for a driver who leaves home at a normally uncongested time of the day and another who leaves during rush hour are very likely to be different. If an accident occurs during the uncongested time of day, the uncongested policy is likely to no longer be optimal. However, the likelihood of such an incident happening on *every* trip is very low. Therefore, it would be inappropriate to requery the routing server for a new policy on a regular basis (for example, every 5 minutes), to guarantee the optimality of the policy for potentially adverse traffic conditions. Figure 3 shows that battery life degrades dramatically if a policy is retrieved on a periodic basis. At the same time, downloading the policy only once leaves the user susceptible to using a suboptimal routing policy.

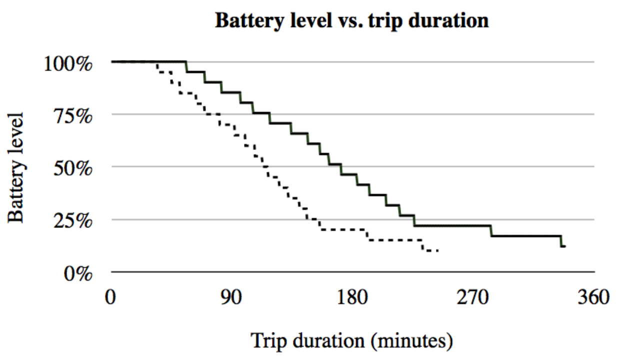


Fig. 3: The push-based policy recalculation provides a significant power savings to the user. In a "periodic refresh" scenario, shown by the dashed line, the battery completely discharges after a 4-hour trip. The "single download" scenario, shown by the solid line, is enabled by the existence of a push server, and enables the battery to last nearly 5.5 hours before full depletion, a 37% improvement.

As a result, we have designed a *push-based* system for determining when a downloaded SOTA policy has become stale. Using the Apple Push Service, built into iOS 3.0 and above, the server can signal to the application when a potentially policy-changing event, such as a serious accident, has taken place along the user's route. A similar push mechanism exists in Android 2.2 and above. The application, upon receiving such a notification, asks for the SOTA server to recalculate a new policy taking this road hazard into account and immediately starts using this policy once it is received. This allows us to provide users with accurate, real-time policies without needing a polling mechanism, which would create a needless battery drain on their devices.

C. Simplified and prerecorded voice directions

As described earlier, our application makes use of prerecorded audio directions for navigational guidance. The benefit

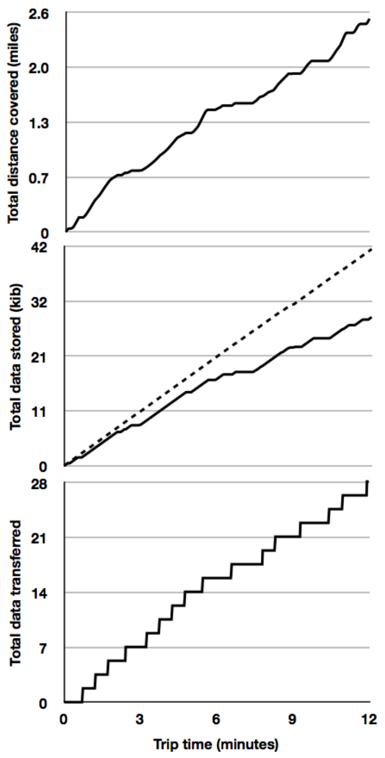


Fig. 4: GPS data collection rules, described in detail in section IV-D, try to strike a balance between providing real-time speed data for the generation of travel-time distributions on the server while minimizing power consumption. **Top:** time-space diagram for a user of the app. **Middle:** Amount of data recorded (in kibibytes) for the trip. Dashed line is the benchmark, recording all points, while solid line is using the rules described, saving 13 kib for this 12-minute trip. **Bottom:** Cumulative amount of data transferred over the network.

of this is twofold: power consumption is reduced and driver distraction is minimized.

One of the problems with providing natural voice directions on mobile devices is that significant computational resources are necessary for synthesizing arbitrary text on-the-fly. For example, no systemwide text-to-speech API is currently provided for iOS devices. Our use of pre-made recordings for the limited set of voice directions required by the application when providing directions allows us to keep power usage low and the application bundle size small.

Significant attention has been recently devoted to the impacts of driver distractions on accidents and traffic safety. Providing a navigation system without aural directions puts the driver and those around him at a significant risk, as he must take his eyes off the road to consult his phone on where to go next [20]. Automatically rerouting the user and notifying him of any changes aurally keeps driver distractions to a minimum and allows the user to fully take advantage of the system without compromising safety.

D. Client as sensor

When a route is in progress, the application collects a GPS trace of the user's location (we address the privacy issues arising from this in section IV-E). Following a set of power-conscious rules (described below), this recorded trace is sent to logging servers, which can then provide this raw data to the real-time travel-time distribution models for processing.

- We eliminate low-speed data points after a stop because this data provides no additional information to the travel-time estimation algorithms: a simple interpolation between the time when $0\ m/s$ was reached and $1\ m/s$ was reached is sufficient for observing that a vehicle was effectively stopped for t seconds. Therefore,
  - Tracing only begins once GPS speed exceeds $0\ m/s$ after the trip has been "started" by the user
  - After speed reaches $0\ m/s$, points are not recorded until:
    - $*$ Speed $> 1\ m/s$, or
    - $*$ Distance from last recorded point $> 20\ m$, or
    - $*$ Time from last recorded point $> 5\ min$
- When the destination is reached, we assume that the user stops driving and leaves the vehicle. Therefore, tracing ends as soon as the driver reaches the road link which contains his destination. It is also possible that the user "abandons" the route without completing it, so traces are terminated when $elapsed\ time > 2 \cdot budget$.
- Since initiating GPRS connections is expensive from a power perspective, it would be inefficient to send GPS coordinates as soon as they are obtained. At the same time, real-time travel-time estimation relies on having up-to-the-minute speed data. To balance these two opposing goals, we send data only if more than 30 points have been recorded or more than 5 minutes have elapsed since the last batch of data has been sent. Usually, this means that data will be sent every 30 seconds. If a vehicle is not moving for over 5 minutes, this allows the server to differentiate between clients that have "disappeared" and those that are simply in congested traffic.

The effect these rules have on data generated and data transmitted is shown in Figure 4.

E. Privacy and estimate fine-tuning

User privacy is maintained in two ways. On the server (data collection) side, all routes from the same device are tied to a single identifier. This identifier, however, is a universally unique identifier (UUID) which is generated on the phone on first application launch but provides no way of tracking which phone generated it. Furthermore, a new UUID is generated every day the application is used; if a trip is in progress at midnight, the new UUID is generated once the trip is completed. On the client (application) side, virtually no location data is stored locally, so a stolen device would provide no information about the user's location to a potential thief.

Accelerometers built into most modern smartphones allow us to combine acceleration and jerk data with GPS speed data to create a model of driver behavior [22] locally on the phone and adjust travel-time estimates, the probability of on-time arrival, and the routing policy based on driver behavior. The benefit of doing this analysis on the phone is twofold. First, there is no reliance on the server to carry out some computations, which allows for immediate adjustments and power savings since no network data is transferred. Furthermore, this approach preserves user privacy, as all behavioral data is stored locally on the device.

V. CONCLUSIONS AND FUTURE WORK

The widespread use of smartphones today has opened up numerous new possibilities for mobile computing and mobile sensing. Navigation and geolocation services have been an integral part of the smartphone experience for many years; however, the navigation experience has largely remained unchanged and in some ways unadapted to needs and environments that are uniquely mobile. Our development of the SOTA routing system seeks to address what we perceive as deficiencies in most current mobile navigation systems while providing additional value to the user and taking advantage of the mobile phone's sensing capabilities to gather traffic data and further improve the experience.

In the near future, empirical travel-time data collected from users utilizing our routing system will allow us to validate the policies provided by the algorithm as well as the amount of travel-time/probe data users will provide when using a routing application such as ours. This data will also allow us to evaluate how much more reliable SOTA's policies are compared to shortest- or fastest-path policies using empirical measurements.

ACKNOWLEDGMENTS

The authors would like to thank the staff at California Center for Innovative Transportation (CCIT) for providing us with continual support and the infrastructure on which we can run this system.

REFERENCES

[1] U.S. Department of Transportation, Federal Highway Administration, *2008 Status of the nation's highways, bridges, and transit: Conditions & Performance. Report to Congress*, ch. 4. USDOT, 2009.

[2] IDC, "Worldwide converged mobile device (smartphone) market grows 56.7% year over year in first quarter of 2010, says IDC." http://www.idc.com/getdoc.jsp?containerId=prUS22333410, 2010.

[3] The Nielsen Company, "Apple edges out Research in Motion in U.S. smartphone market share." http://www.macrumors.com/2010/12/01/apple-edges-out-research-in-motion-in-u-s-smartphone-market-share/, December 2010.

[4] S. Samaranayake, S. Blandin, and A. Bayen, "A tractable class of algorithms for reliable routing in stochastic networks," in *International Symposium on Transportation and Traffic Theory*, vol. 17, (Berkeley, CA), pp. 341 – 363, July 18-20 2011.

[5] H. Frank, "Shortest paths in probabilistic graphs," *Operations Research*, vol. 17, pp. 583–599, 1969.

[6] R. Bellman, *Quarterly of applied mathematics*, ch. On a routing problem, pp. 87–90. No. 16, Brown University, Division of Applied Mathematics, 1958.

[7] E. Nikolova and M. Brand, "Optimal route planning under uncertainty," in *Proceedings of International Conference on Automated Planning and Scheduling*, 2006.

[8] E. Nikolova, J. Kelner, and M. Brand, "Stochastic shortest paths via quasi-convex maximization," *Algorithms–ESA 2006*, 2006.

[9] Y. Nie and X. Wu, "Shortest path problem considering on-time arrival probability," *Transportation Research Part B*, vol. 43, no. 6, pp. 597–613, 2009.

[10] Y. Y. Fan and Y. Nie, "Optimal routing for maximizing travel time reliability," *Networks and Spatial Economics*, vol. 3, no. 6, pp. 333–344, 2006.

[11] Y. Nie and Y. Fan, "Arriving-on-time problem: discrete algorithm that ensures convergence," *Transportation Research Record: Journal of the Transportation Research Board*, no. 1964, pp. 193–200, 2006.

[12] M. Lighthill and G. Whitham, "On kinematic waves II a theory of traffic flow on long crowded roads," in *Proceedings of the Royal Society of London*, vol. 229, pp. 317–345, 1956.

[13] P. Richards, "Shock waves on the highway," *Operations Research*, vol. 4, no. 1, pp. 42–51, 1956.

[14] G. Evensen, "The ensemble Kalman filter: Theoretical formulation and practical implementation," *Ocean dynamics*, vol. 53, no. 4, pp. 343–367, 2003.

[15] D. Work, S. Blandin, O.-P. Tossavainen, B. Piccoli, and A. Bayen, "A traffic model for velocity data assimilation," *Applied Mathematics Research eXpress*, vol. 2010, no. 1, pp. 1–35, 2010.

[16] R. Herring, A. Hofleitner, S. Amin, T. A. Nasr, A. A. Khalek, P. Abbeel, and A. Bayen, "Using mobile phones to forecast arterial traffic through statistical learning," in *89th Transportation Research Board Annual Meeting*, (Washington, D.C.), January 10-14 2010.

[17] T. Hunter, R. Herring, P. Abbeel, and A. Bayen, "Path and travel time inference from GPS probe vehicle data," *Proceedings of the Neural Information Processing Systems foundation (NIPS), Vancouver, Canada*, 2009.

[18] T. Hunter, T. Moldovan, M. Zaharia, S. Merzgui, J. Ma, A. Bayen, and M. Franklin, "Scaling the mobile millennium system in the cloud," in *Proceedings of the ACM Symposium on Cloud Computing (SOCC)*, October 2011.

[19] Apple Computer, "The notification payload." http://goo.gl/ZiuAH.

[20] K. Kishi and S. Sugiura, "Human factors considerations for voice route guidance," in *Automotive display systems and IVHS*, pp. 99–109, 1993.

[21] A. Thiagarajan, L. Ravindranath, K. LaCurts, S. Madden, H. Balakrishnan, S. Toledo, and J. Eriksson, "Vtrack: accurate, energy-aware road traffic delay estimation using mobile phones," in *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, pp. 85–98, ACM, 2009.

[22] A. Doshi and M. M. Trivedi, "Examining the impact of driving style on the predictability and responsiveness of the driver: Real-world and simulator analysis," in *Intelligent Vehicles Symposium (IV)*, (San Diego, CA), pp. 232–237, IEEE, June 21-24 2010.