

# TrafficView: A Driver Assistant Device for Traffic Monitoring based on Car-to-Car Communication

Tamer Nadeem, Sasan Dashtinezhad, Chunyuan Liao  
 Department of Computer Science  
 University of Maryland  
 College Park, MD 20742

{nadeem,sasan,liaomay}@cs.umd.edu Cristian Borcea, Porlin Kang, Liviu Iftode  
 Department of Computer Science  
 Rutgers, the State University of New Jersey  
 Piscataway, NJ 08854  
 {borcea, kangp, iftode}@cs.rutgers.edu

## I. EXTENDED ABSTRACT

Vehicles are part of people's life in modern society, into which more and more high-tech devices are integrated. Most of the current research focuses on the functionalities of individual vehicles, and less attention has been paid to the cooperation among vehicles and road facilities, which forms the whole transportation system. Moreover, a common platform for inter-vehicle communication is necessary to realize an intelligent transportation system supporting safe driving, dynamic route scheduling, emergency message dissemination, traffic condition monitoring, etc. In this paper, we present TrafficView, a system that the next generation of vehicles can be equipped with to provide the driver with a real time view of the traffic on the road far beyond what he can see. Equipped with a TrafficView, each vehicle on the road gathers and disseminates information about other vehicles in a peer-to-peer fashion, with no need for central server or special infrastructure on the highways. This support can be particularly useful in difficult driving situations such as foggy weather or traffic jams, allowing the driver to make driving decisions based on full knowledge about the road ahead.

## TrafficView Overview

A TrafficView system consists of four components: a computer embedded in the vehicle, a GPS receiver, a short-range wireless network interface, and a display. The computer is connected to the GPS receiver from which it gets the geographical position of the vehicle, the speed of the vehicle, and the global time. Using the wireless network interfaces, the computers embedded in the vehicles can form mobile ad hoc networks to dynamically exchange traffic information over multiple hops. Each computer aggregates the traffic information received from other vehicles, update its own information, forwards it to the neighbor vehicles, and present it on the display (Figure 1).

The main important feature of TrafficView is the ability to present the driver with a real time view of the traffic on the road independent of the current visibility conditions. In doing so, the TrafficView system does not need any specific

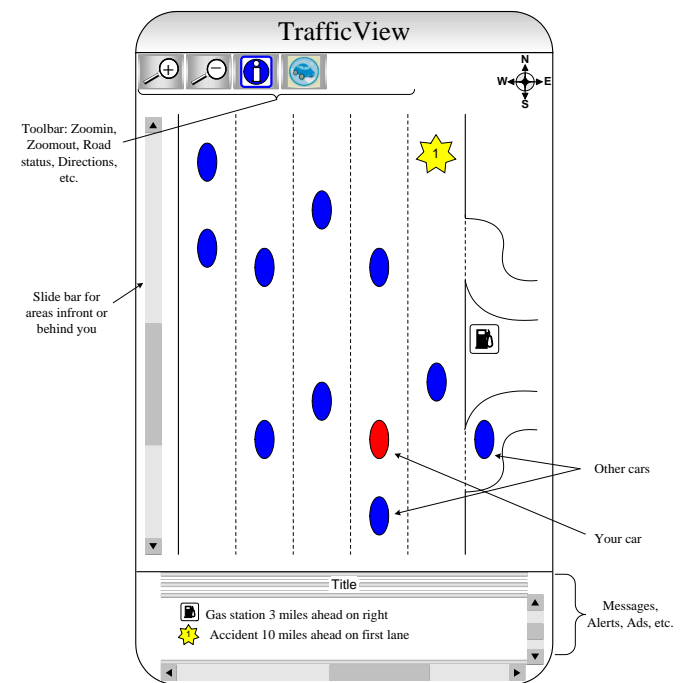


Fig. 1. Example of TrafficView Display Snapshot.

infrastructure installed on the roads. Vehicles construct on-the-fly an ad hoc network which changes dynamically to reflect the current traffic situation. Unlike infrastructure-based services for which scalability is a difficult problem, TrafficView is inherently scalable due to its completely decentralized design. Given the tradeoff between the ability to present the driver with as much information as possible and the limited wireless bandwidth, TrafficView uses data aggregation mechanisms to reduce the size of the data sent out over the network. An aggregation algorithm reduces the size of the information while taking the semantics of the data into account. For instance, if a vehicle has two records about two vehicles which are close to each other and are moving with relatively the same speed, an aggregation algorithm can replace those two records with one record representing both vehicles. We have developed

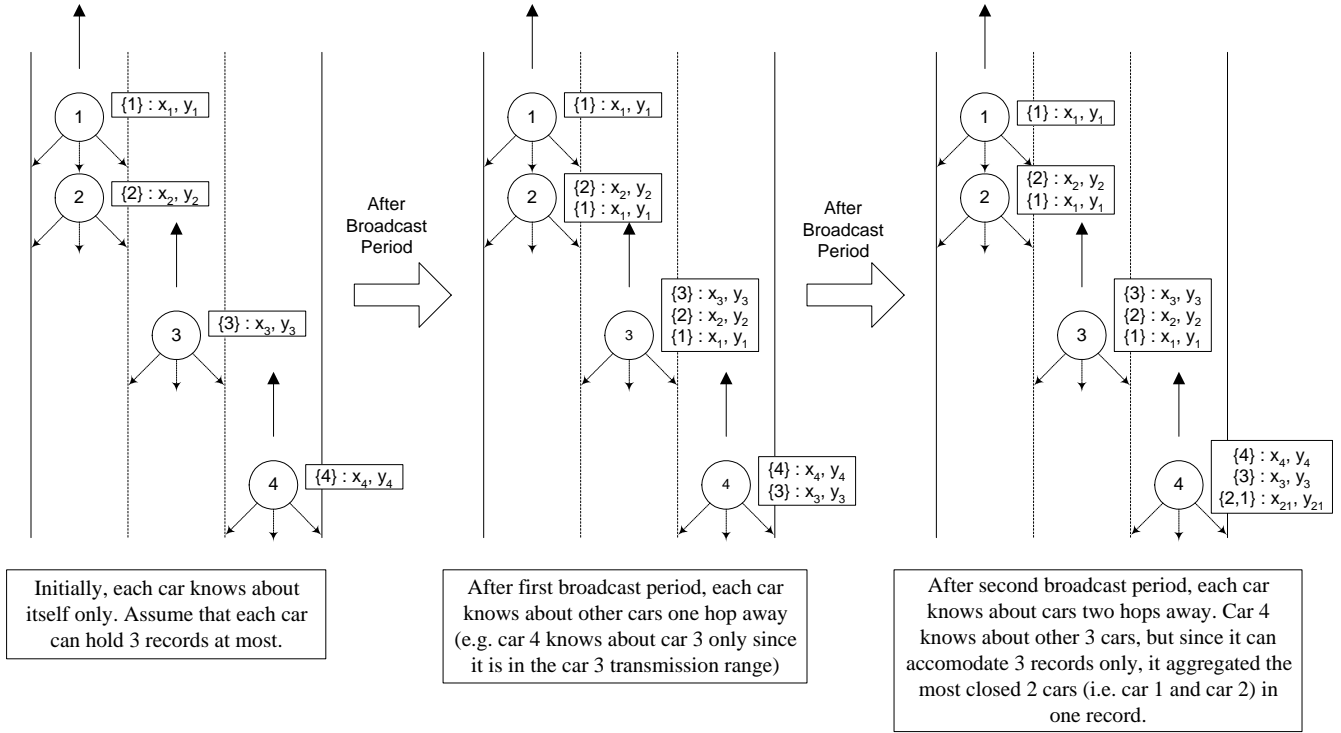


Fig. 2. TrafficView Scenario.

novel scalable aggregation algorithms that will be used by the TrafficView devices [3]. An aggregation example is presented in Figure 2.

In addition to passively receiving traffic information (push mode), TrafficView enables the driver to query information about specific targets (pull mode). For example, a vehicle can query about the average vehicle speed in a region (e.g., Exit 9 on I95 highway).

## Performance Evaluation

In a previous paper [3] we measured the performance and scalability of the system, a simulation version of a TrafficView-equipped vehicle network has been built into NS-2 network simulator. We have developed our own traffic scenario generator that given the road parameters such as length of the road, number of lanes, average vehicle speed, and average gap between vehicles, generates traffic scenarios that are representations of vehicle movements on a road. These traffic scenarios were used to assess the performance of the system.

We ran several simulations with 355, 580, and 960 nodes (corresponding to the number of vehicles on the road). Three metrics were used to evaluate the system's performance: (1) accuracy, which is measured by the error in position of different vehicles, (2) visibility, which is the length of the road in front of a vehicle that the vehicle has information about, and (3) coverage, which denotes the percentage of the vehicles on different regions of the road that the vehicle have information about. The simulation results for different

scenarios show that the system achieves good coverage and visibility while maintaining acceptable errors. The visibility almost equals the length of the road. The coverage is close to 100% for closer regions and decreases for farther away regions.

## Prototype Implementation

In this paper we present the TrafficView prototype we implemented to validate our design. The prototype consists of a portable device HP iPAQ with Linux Familiar distribution augmented with a Global Positioning System (GPS) and a IEEE 802.11b wireless card. The iPAQ device runs the TrafficView software that displays road traffic information and is responsible for the interaction with other vehicles. We assume that each vehicle has a unique identification number (ID). The vehicle identification number (VIN) or drivers driving license number are good candidates for the ID.

Figure 3 shows the software components (modules) of a vehicle in the system. Each vehicle stores records about other vehicles in its local data sets. When the record is first received in a broadcast message, it is stored in the non-validated data set, since it might contain outdated or conflicting information. After these records are examined for validity, they are moved and merged with the validated data set.

The current TrafficView communication and aggregation protocol is implemented in Java. In the paper, we also describe a second implementation of TrafficView using Smart Messages, a Java-based distributed computing platform that we have recently developed for networks of embedded sys-

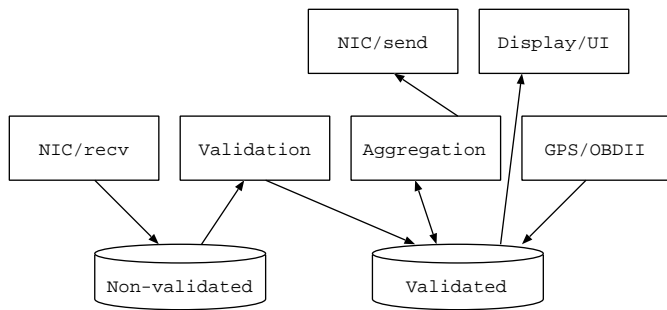


Fig. 3. The structure of a vehicle in TrafficView

tems [?]. Smart Messages are execution units which migrate through the network to execute on nodes of interest (named by content) and route themselves at each node in the path toward a node of interest. With Smart Messages, the TrafficView protocol can be dynamically uploaded and routing can be tailored to specific traffic conditions.

Also in this paper we describe the privacy and the security problems. Privacy is an important issue in such a system. Different privacy levels should be available from which the drivers can select. One level of privacy is to completely hide any information about the vehicle while it continues to participate in relaying other vehicles' information. Another level is to allow others to gain information about the vehicle without being able to identify it. For example, vehicles on the road may know about a group of vehicles, including the target one which is located in an area on the road without being able to identify the exact location of any vehicle including the target.

Security and trust are two other important issues in such a system. A fraudulent vehicle could disseminate information about nonexistent vehicles, or broadcast bogus information about existing vehicles. Different mechanisms is required to prevent this and to identify those fraudulent vehicles and avoid accepting their packets.

## II. GPS DATA PROCESSING

Given a set of cars carrying devices connected in the TrafficView network we need to know for each car what are the relative positions of, distances to, and directions of the rest of the cars with respect to that car. In order to solve this problem, we need to know for each car, at any given time, on what road the car is driven, what is the closest mapped point on that road, and in what direction on that road. Once we have this information we can compute the relative distances and moving directions among the cars.

We will employ a devices and related software to read a stream of real time Global Positioning System (GPS) location readings. The software module processes the output from a Garmin GPS device [5] using the Garmin Simple Text Output Protocol (GSTOP) [6]. When the device output is enabled for this protocol, the output is a set of lines, each line containing a sampled GPS coordinate consisting of information about the current date and time, the latitude and longitude coordinates (in degrees as integer numbers and minutes as float number),

speed, type of GPS algorithm to compute the coordinates based on the information received from satellites, and error in meters. In order for the device to function properly in the GSTOP mode, the device's software must be updated to the latest version offered by Garmin on the company website. An example of a GSTOP reading is given in Figure 4.

```
@031222181226N4031717W07428034G014+00043E012.3N0031U0001
```

Fig. 4. Example of a character line read from the output of a Garmin GPS device using Garmin Simple Text Output Protocol.

This reading has the following translation: start of GPS reading line (@), year of 2003, month 12, day 22, 18 hours, 12 minutes, 26 seconds, North hemisphere 40 degrees and 31.7171 minutes latitude, West hemisphere 074 degrees and 28.034 longitude, reading was obtained using 3D (latitude, longitude, and altitude) GPS positioning method (using information from at least 4 satellites), 14 m horizontal position error, positive altitude of 43 meters above the sea level, East velocity of 012.3 m/s, North of velocity 003.1 m/s, Upwards vertical velocity 00.01 m/s, end of GPS reading line.

Because the number of meters per latitude or longitude degree differs dramatically with the latitude, all the computations involving distance are done in meters, using conversion tables from latitude and longitude degrees to meters [7]. The corresponding meter values for degree values between two consecutive latitude or longitude degrees are computed using interpolation.

The road graph is created using the Record Type 1 (RT1) and Record Type 2 (RT2) files [8] from the data files offered by the US Census Bureau through the 2000 Tiger Line database [9]. Each road is recorded as a set of reference points that represent extremities of segments in a chain. Segments of roads are parts of GT-polygons (GT stands for geometry and topology). RT1 files contain information about the roads like name, type, direction, and start and end points. RT2 files contain information about intermediary reference points on the roads described in RT1 files. More map area information is recorded in other Record Type files (i.e., 3-9, A, C, H, I, P, R, S, Z), and it is not used by the software module. There is a set of RT files for each county in each state of US [10].

The illustration (taken from [8]) in Figure 5 shows an example for a generalized block of a Tiger Line map recorded in RT1 and RT2 files. It consists of three GT polygons and contains a point landmark (*Parkside School*) inside GT polygon 2 and an area landmark (*Friendship Park*) that is coextensive with GT-polygon 3.

Identifying the location on the map based on a sampled GPS position involves identifying the closest reference map points (recorded in RT files) to it. This implies the need for some sort of clustering of the map points that is minimal space-wise and search-wise due to the limited memory and computing resources of a PDA.

A solution is using *simple Peano keys* [8] (page 3-44), similar with the Peano keys invented by the 19th century Italian mathematician Giuseppe Peano, who proved that a 2D coordinate system can be collapsed into a 1D coordinate

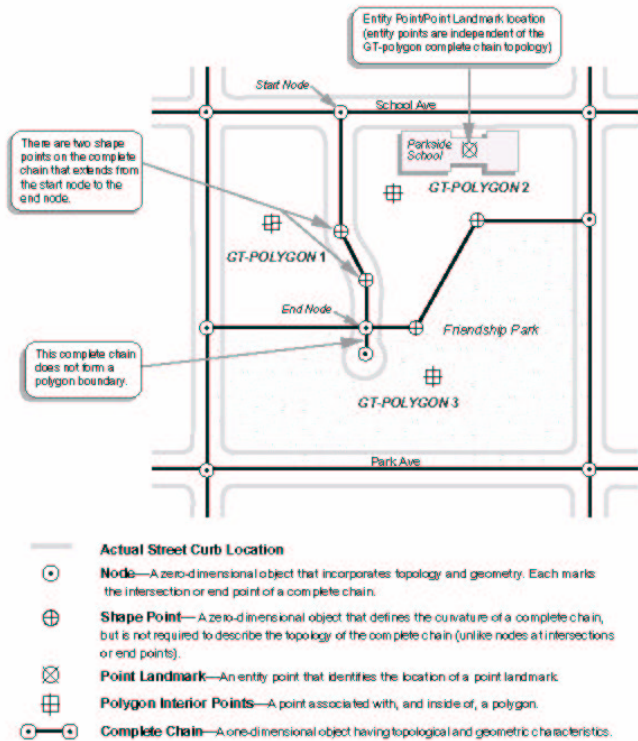


Fig. 5. Map recording in Tiger Line Files of type RT1 and RT2

system (i.e., considered as a one dimensional line). To generate a simple Peano key we interleaving the digits of the numbers that give the x and y coordinates of each point in 2D. If we create simple Peano keys for each point in a map and put them in a sorted array in lexicographic order, the adjacent keys represent map points that are close to each other on the map.

Considering the previous GPS reading at North hemisphere 40 degrees and 31.7171 minutes latitude, West hemisphere 074 degrees and 28.034 longitude, it can be represented by the pair of numbers (in degrees) (+40.5286111, -074.4672222), from which a Peano key suitable for our problem can be generated as -+04704542687621212 (decimal points are discarded). The -+ signs actually indicate that the GPS point is in the NW quadrant.

Let us assume that we are doing mapping in a county of a state. The software module will first pre build a road map using the RT1 and RT2 Tiger Lines files from the US Census Bureau. First, it builds a list of all roads in the county and each road will contain the list of mapped reference points. Then it computes all the Peano keys for the reference points using their latitude and longitude coordinates and stores the keys into a sorted array. Now a map has been constructed.

Let us assume that we have a GPS sampled point and we want to find out on what road it is located if it was recorded on a road, or what is the closest road to it if it was recorded near a road (see Figure 6). The software module computes the Peano key corresponding to the coordinates of the GPS sample point and executes a binary search with this key in the sorted array of keys. Obviously, the key is not in the array,

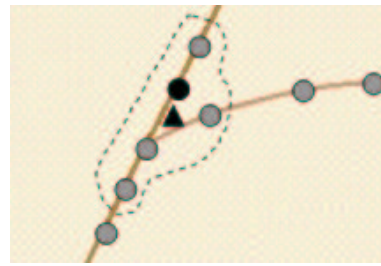


Fig. 6. Identifying the closest road to a GPS location. The dots (gray or black) represent exact map points generated from Tiger Line information. The triangle represents a GPS location. The 5 dots in the enclosure are the map points that have the closest simple Peano keys to the simple Peano key of the GPS location for a radius  $r = 2$ . The map point that is closest in space to the GPS location is represented as a black dot. The road containing the black road is considered to be the closest road.

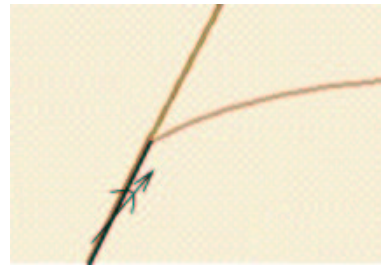


Fig. 7. Matching a GPS segment (bounded by two successive GPS location readings) with a road segment (recorded on a map created based on Tiger Line information) by computing the angle between the two segments.

but the last key from the sorted array used in comparison by the binary search is the closest Peano key and corresponds to one of the very close reference points. We need the closest point, so we consider all the Peano keys in the array before and after the closest Peano key that are in a given radius  $r$  (e.g., we consider the  $r$  keys before the closest key, the closest key, and  $r$  keys after the closest key in the array). This gives us  $(2r + 1)$  close keys, from which we chose the one that corresponds to a reference point that is the closest (i.e., situated at the minimum distance in space) to the GPS sampled point. That reference point gives us the closest road to the sampled location.

Identifying the closest road does not mean that we know for sure the road we are actually on. The following issue needs to be solved first. Let us assume now that we are processing a continuous output of sampled GPS points recorded from a GPS devices while driving on a route consisting in multiple roads. How do we decide that we are on a certain road and when do we decide that we are moving on a different road? For each consecutive and distinct closest Peano keys corresponding to two (distinct) GPS sampled points, the software module considers the two corresponding reference map points. If the two points are on the same road, the software module computes the angle between that segment and the segment bounded by the two GPS sampled points, as shown in Figure 7 (where the road segment is shown as a thick segment and the segment bounded by the two GPS sampled points is shown as an arrow).

If the angle is less than 15 degrees, the decision is that it is

highly probable that we are on or very near to the road that contains the two reference map points. How do we decide on each road we are exactly? The software module uses a smoothing veto technique similar with a short-term memory. The decision on which road we are is taken choosing the road that appears most frequently in the last  $m$  identified roads.

One more refinement has to be added. First, the GPS measuring accuracy is between 3 and 30 meters, and therefore for a given GPS sample point we can be anywhere inside the circle with a radius between 3 and 30 meters and centered in the real location. Second, the reference points given by the Tiger Lines RT1 and RT1 files can be at quite large distances (up to 200 meters). Therefore, the algorithm described here may miss identifying the correct road if no reference point is close to the current GPS sample point, but other reference points from roads that intersect our road are nearby. To drastically reduce road identification errors due to the two facts described, when the road map is built in the beginning, each Tiger Line road segment is split in sub segments (i.e., sub sampled) creating equally distant reference points on it through interpolation, such that any of the segments that form a road have length less than a predetermined value  $d$  (e.g., 20 meters).

The method described here is highly accurate. For example, on a set of 2628 GPS sample points recorded at a rate of about 2 per second from a GARMIN GPS device while driving on 22 roads over a total distance of about 15.5 miles, for about 22 minutes (including waiting for traffic lights), with the average speed of about 43 mph, all the roads were accurately identified. The tuning parameters (mentioned before) were  $d = 60$ ,  $r = 4$ , and  $m = 10$ . The software is implemented in C language. On a Pentium III processor at 1 MHz, the processing speed for a recorded log of 2628 GPS sampled points is under 1 second once the road map (of the Middlesex county of the state of New Jersey, 34689 roads and 231294 reference points) has been created from original Tiger Line Files in about 13 seconds. For the given  $d = 60$  there was no road segment sub-sampling needed. The same result was obtained for  $d = 10$ ; in that case 807215 reference map points were created in 14 seconds and the log processing remained under 1 second. The reference points are created once only at the startup of the software module. Most of the startup time is spent with sorting the array of simple Peano keys.

Figure 8 shows how the road identification software module works: GPS coordinates are read from a GPS device and the road name and the name of and distance to the next crossroad are computed and input in a text to speech engine and a graphic mapping module that generates a map.

For each road, the road points are stored in an array. Once two consecutive road points are identified, the direction in which the car is moving on that road can be computed by looking at the direction in which the indices of the points in the array are decreasing or increasing. Therefore, for a set of cars connected in the Traffic View network, we can tell what cars are moving in one direction and what cars are moving in opposite directions (see Figure 9). We can also compute the relative movement between two cars i.e., if they move in the same direction or in opposite directions. From the closest

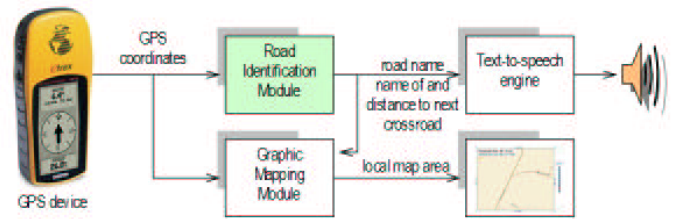


Fig. 8. Integration of the road identification software module in a complete GPS system.

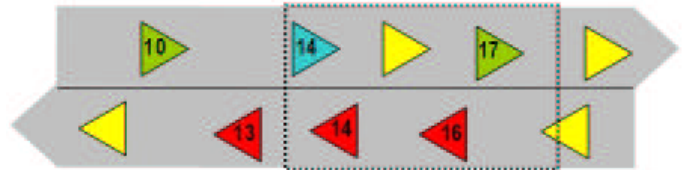


Fig. 9. A set of cars (symbolized as triangles) connected through the Traffic View network (symbolized as red, green, and blue triangles with road point indices inside) on a section of a 2 way road and moving in both directions (given by the triangle orientations). The blue car only needs to know which cars are in front of it (shown inside the dotted rectangle, greater indices), in which directions they are moving relatively to it (same direction in green, opposite direction in red), and what are the current distances from it to those cars.

mapped points can also approximate the distances between cars. From the values of the indices corresponding to those points in that point array of the road and moving directions we can compute the relative positions (*in front*, *near to*, and *behind*) between any two cars.

An implemented extension of this software module also identifies the crossroads along a GPS recorded route. The sorted array of simple Peano keys is used to determine the crossroads during the module initialization.

## REFERENCES

- [1] I. Chisalita, N. Shahmehri, "A peer-to-peer approach to vehicular communication for the support of traffic safety applications," *5th IEEE Conference on Intelligent Transportation Systems*, pp. 336–341, Singapore, Sept. 2002.
- [2] E. Welsh, P. Murphy, P. Frantz, "A Mobile Testbed for GPS-Based ITS/IVC and Ad Hoc Routing Experimentation," *International Symposium on Wireless Personal Multimedia Communications (WPMC)*, Honolulu, HI, Oct. 2002.
- [3] T. Nadeem, S. Dashtinezhad, C. liaomay, L. Iftode, "TrafficView: A Scalable Traffic Monitoring System," *2004 IEEE International Conference on Mobile Data Management (MDM)*, Berkeley, CA, Jan. 2004.
- [4] Cristian Borcea, Deepa Iyer, Porlin Kang, Akhilesh Saxena, Liviu Iftode, "Cooperative Computing for Distributed Embedded Systems," *Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS)*, pp. 227–236, Vienna, July 2002.
- [5] "Garmin devices," <http://www.Garmin.com>, retrieved 1/2004.
- [6] "Garmin, Simple Text Output Protocol," [http://www.Garmin.com/support/text\\_out.html](http://www.Garmin.com/support/text_out.html), retrieved 1/2004.
- [7] Humerfelt, S., "The Earth according to WGS 84," [http://home.online.no/~sigurdhu/Grid1\\_deg.htm](http://home.online.no/~sigurdhu/Grid1_deg.htm), retrieved 1/2004.
- [8] "US Census Bureau, Tiger Line Technical Documentation," [http://www.census.gov/geo/www/tiger/rd\\_2ktiger/tgrrd2k.pdf](http://www.census.gov/geo/www/tiger/rd_2ktiger/tgrrd2k.pdf), retrieved 1/2004.
- [9] "US Census Bureau, Topologically Integrated Geographic Encoding and Referencing system (Tiger Line)," <http://www.census.gov/geo/www/tiger>, retrieved 1/2004.
- [10] "US Census Bureau, 2002 Tiger/Line Files by state," <http://www.census.gov/geo/www/tiger/tiger2002/tgr2002.html>, retrieved 1/2004.