Improving Public Transportation Through Crowd-Sourcing

Anirudh Vemula, Nikhil Patil, Vivek Paharia, Aneesh Bansal, Megha Chaudhary,
Naveen Aggarwal, Divya Bansal, K. K. Ramakrishnan, Bhaskaran Raman

Abstract: Commuting on roads in densely populated cities of the developing world is fraught with high delays and uncertainties. Wide use of public transportation can ease the load on the road infrastructure, but such use is not convenient, partly due to the unpredictable nature. In this work, our goal is to improve the usability of public transportation, through better information. Such information can lead to better planning and predictability for commuters. We take a crowd-sourced approach where information about transportation units as well as road conditions is crowd-sourced from commuters. The information is then processed and made available to other commuters. In this context, this paper presents a naming framework we have developed, which will enable flexible and scalable content-driven data gathering and dissemination. Based on a preliminary implementation of the framework, we present various field-experiment results which shed light on the practicality of the proposed approach as well as on technical issues which need further careful addressing.

I. Introduction and Motivation

Commuting on roads, especially in peak hours, is not only unpleasant but also wasteful of time and fuel. While this is true of most places in the world, it is especially true in densely populated cities of the developing world where the stress on road space is especially high. Use of public transportation (buses, commuter trains) can alleviate the situation, if only a large section of population were to use it. But we currently are in a vicious cycle: most who can afford private vehicles shy away from public transportation, due to their poor usability and predictability; and this further stresses the already stressed road infrastructure.

In this work, termed CARTS (Communication Assisted Road Transportation Systems), we seek to improve the usability of public transportation systems, primarily through better availability of real-time information. Our hope is that this will contribute to breaking the vicious cycle, by providing better planned and more predictable public transportation commutes.

Our approach is to crowd-source information and public transportation units, as well as other relevant information such as real-time road conditions, from commuters themselves. For this, we use commuters' smart-phones. Information is crowd-sourced in an automated fashion (e.g. speed of movement through GPS sensor) as well as through manual input (e.g. "I have boarded bus number 356 now"). Subsequently, we disseminate the processed information to interested commuters, to answer queries such as "When is the next bus number 25 expected at my bus stop?", "When can I expect to reach my destination by boarding that bus?".

In this paper, we make a beginning toward supporting the above crowd-sourced information framework. We present the design of a naming scheme (Sec. III) which supports flexible and potentially scalable means of data gathering and dissemination. The naming scheme enables content-driven information collection and dissemination. We have a preliminary implementation of the crowd-sourced framework on the Android platform. Prior to a pilot deployment, we have collected several field measurements, results from which are presented in Sec. IV. Our overall observations are that using GPS connectivity in buses is not always reliable, likely due to physical obstructions and surrounding people. We find that bus-stop dwell times do not contribute significantly to travel time. Finally, we also find that automated detection of bus stops is not easy, as there is significant variability along various dimensions: where exactly a bus stops, and how long it stops, from one trip to another.

II. Related Work

We now present related prior work which has looked at crowd-sourced approaches for getting road traffic related information.

VTrack [3] is a system which dynamically tracks location of vehicles. The focus here is on lowering power consumption of the smart-phones involved, by using Wi-Fi based positioning in place of GPS when possible. CTrack [4] furthers this work by considering GSM cellular signal-based positioning as well. Nericell [2] develops techniques to get information about road conditions (e.g. congested, noisy) using smart-phone sensors. The work in [5] uses GPS data to deduce information about road conditions, e.g. congestion level in each road segment. The techniques in the above works are complementary and orthogonal to our goal of crowd-sourcing public transit information.

EasyTracker [1] deals with prediction of arrival time of a transportation unit. It however assumes dedicated on-vehicle sensors while our focus is on crowd-sourced information.

978-1-4799-8439-8/15/$31.00 ©2015 IEEE

Zhou et. al. [6] develop a bus arrival time prediction algorithm using crowd-sourced information. However the work assumes presence of a specific bus ticketing system, and specific sounds made by the card readers of the ticketing system. These sounds are used to detect when a commuter has boarded a bus. Clearly, the same mechanism will not carry over to bus systems which do not have such a ticketing system. Likewise, the bus classification approach in [6] works based on partial bus route estimates. This will not work in situations where different bus routes have large common sub-routes, a common situation in many cities.

Apart from research work, several popular paid/free smart-phones applications have also been developed toward crowd-sourced traffic information. RAC's mobile traffic application (www.rac.co.uk/travel/mobile-apps/) crowd-sources information about road incidents and delays. It also has a trip planner, but specific to the UK and to cars.

Google Maps (maps.google.com) uses crowd-sourcing to get real-time road traffic information. HERE (here.com) has similar features. Waze (www.waze.com), in addition, includes facilities to crowd-source changes to the road map itself. But the above systems do not include means to crowd-source public transportation specific information.

Moovit (www.moovitapp.com) and Tiramisu (tiramisutransit.com) are the closest to our goal in terms of crowd-sourcing information specific to public transit. Both have crowd-sourcing of information about specific routes, crowd-level in specific buses, cleanliness, feedback on general experience, etc. Tiramisu has explicit start/stop buttons for the commuter to share real-time information about the bus's movement.

Open Technical Issues: Although the above work exists in the space of crowd-sourcing road traffic information, much work remains to be done. Most of the above work has been done in developed countries. Several aspects are different on city roads in developing countries like India. First, static information about bus transit schedules (which are used in the absence of real-time crowd-sourced information) are generally hard to come by and are unreliable even when available. Such static information is used by most of the above systems today. Second, it is unclear how well location determination will work, since auxiliary information such as Wi-Fi based location is mostly unavailable. Third, installation of infrastructure (such as a GPS unit) on buses has been tried and found to be impractical in the long term due to maintenance issues.

With respect to the currently available applications, listed above, their performance in terms of accuracy of information is unclear as there is no publicly reported evaluation of the system.

In this work, we have designed a naming framework specific to crowd-sourcing of public transit information. We also present several on-field measurements from two Indian cities (Mumbai, Chandigarh), which represent a beginning toward understanding the practicality of crowd-sourcing public transit information.

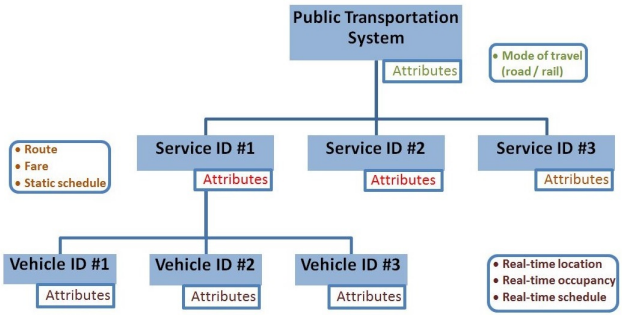

Fig. 1: Public transportation hierarchy

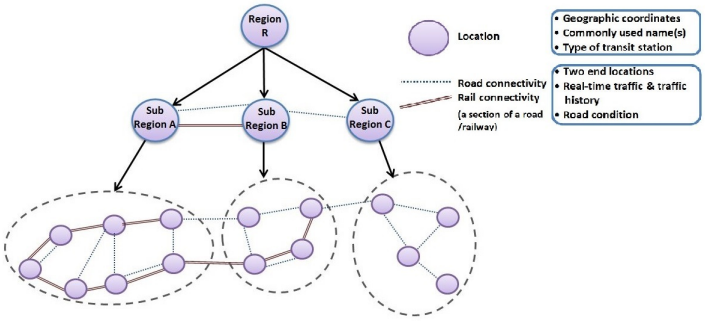

Fig. 2: Location Hierarchy

III. Naming Scheme for a Content-Centric Framework

In this section, we discuss the content naming mechanism in CARTS that enables name-based storage and retrieval of relevant information. It makes the access operations on the content easier by creating a hierarchical structure. For the CARTS application domain, we initially develop a naming framework that supports two classes of entities: public transportation systems, and locations. We have designed a separate naming hierarchy for each, as shown in Fig. 1 and Fig. 2 respectively. All the information relevant to private transportation systems is available in the location hierarchy. Public transportation systems need both the hierarchies. The public transportation hierarchy refers to the location hierarchy for determining information about its routes. In each case, a node in the hierarchy names a relevant entity and has attributes providing additional information about the entity, as described below.

Public transportation hierarchy

The public transportation hierarchy is composed of three levels (Fig. 1). The specific public transportation system is at the top. Services offered (e.g., different routes) form the middle layer, and individual transport vehicles which may be used for a given service, and potentially in transit currently, form the bottom layer.

Among the major attributes associated the top layer (public transportation system) are: name of the authority

running the public transport, mode of transport (road or rail), type of transit (city bus, local trains, monorail, etc.). These can be used by users while querying, filtering out irrelevant information.

The main attributes associated with the middle layer (services run by the system) can be broken up into two classes: static attributes and historical attributes. Static attributes of a service are obtained from the transportation authority, and do not change unless there is a change in the actual service offering. Historical attributes provide the history of the service so that it can be used in the absence of real-time information.

The attributes are as follows. (a) *Service name* – a unique name that can be used by users to identify the service, e.g., bus route number 422. (b) *Route* is a sequence of locations (serving as transit stations) It may be specified as a sequence of latitude and longitude values or as transit points in the location hierarchy. (c) *Fare* is specified for each pair of transit stations on the route. (d) *Schedule* is the predefined arrival and/or departure times for each transit station on the route (it could be specified as a frequency). (e) *History:* historical data about the actual arrival and/or departure times for transit stations is stored, which can be used for example to estimate attributes (e.g., arrival times of a bus), in the absence of real-time information. This could be organized based on time of day, day of the week, etc. (f) *Updates:* could be used to reflect temporary changes (e.g., schedule changes).

Finally, the attributes of the entities in the bottom layer (transit vehicles offering the service) include the following: (a) real-time location, speed and direction, (b) occupancy (i.e., level of crowding in bus), (c) real-time schedule, which is the actual arrival and/or departure times for the stations that have been passed up to now, and the forecast times for the remaining stations (e.g., based on current traffic and/or history of traffic available in the location hierarchy).

Location hierarchy

The location hierarchy (Fig. 2) is essentially a geographic hierarchy with transport connectivity information embedded into it. The location hierarchy is also used to represent the road and rail connectivity as edges between the nodes in the hierarchy. Note that the connectivity edges themselves are not part of the naming hierarchy. In Fig. 2, region R contains sub-regions A, B and C which further contains sub-sub-regions.

The attributes of a *location* entity include the geographic coordinates and the commonly used names of the location (e.g., *chor bazaar*). Also, if the location is a transit station, information about public transport services available at the location is specified.

The attributes of the *transport connectivity edges* include the two end locations of the edge and type of edge (road or railway). If the edge represents a section of a road, the following attributes are also stored. (a) *Real-time traffic state:* this can be binary (congested versus free-flowing traffic) or a multivalued attribute. (b) *Road condition:* how smooth or bumpy the road section is. (c) *Traffic history:* is maintained which can be used to estimate traffic state in the absence of real-time information.

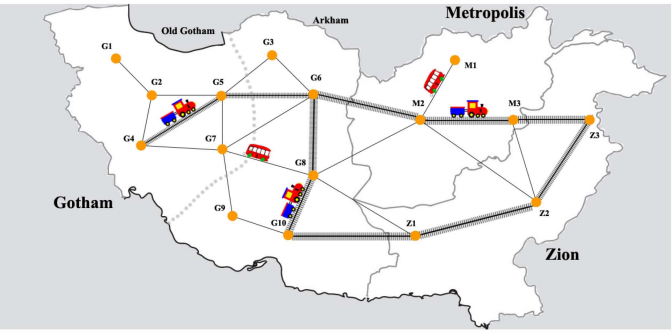

Fig. 3: Map of a fictional Atlantis island

Note the interconnecting linkages between the public transportation hierarchy and the location hierarchy. Also note that the attributes in the hierarchy can change in real-time.

Such a hierarchical representation gives flexibility in content-based query-response or publish-subscribe. For instance, a commuter may query for any public transport (a top-layer entity) destined to location X, or may query for bus service number 422 (a middle layer entity), or may query for expected arrival time at destination X of the bus she has just boarded (a bottom layer entity).

Examples of Content Naming Hierarchies

Consider a fictional island - Atlantis with three cities viz. Gotham, Metropolis and Zion. Gotham city has two sub-regions viz. Old Gotham city and Arkham. Each of these cities further contains locations which are rail or bus stations, or famous places in the city (named as G1, G2, M1, Z1, etc. for simplicity). These locations are defined by unique latitude and longitude i.e., these cannot further contain any locations. Atlantis runs railway and bus services across these cities. The map of Atlantis along with its road and rail network in shown in Fig. 3.

Location hierarchy for Atlantis:: The location hierarchy for Atlantis is shown in Fig. 4. As it can be seen from figure, location hierarchy maintains a parent-child relation between Atlantis and its three constituent cities, Gotham and its constituent two sub-regions, and also between each region and various locations contained in it.

We only need to maintain the transport connectivity information among the lowest level locations, and using these links, connectivity information among the higher level locations can be easily computed. Also, the information about roads is also represented in this hierarchy. For instance, rail stations located on a particular road can be easily found out using this hierarchy.

The information related to roads, such as distance covered, traffic, road condition is also stored in the location

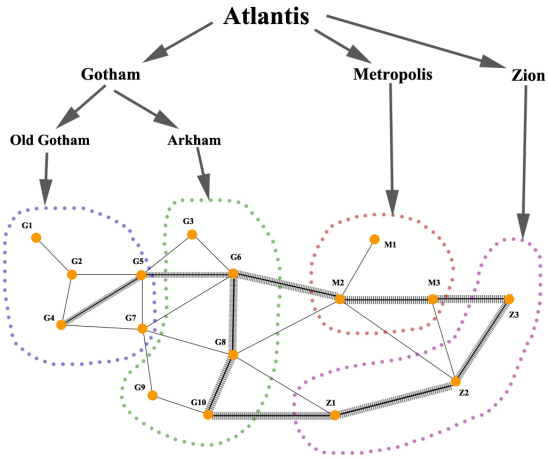

Fig. 4: Location hierarchy for Atlantis

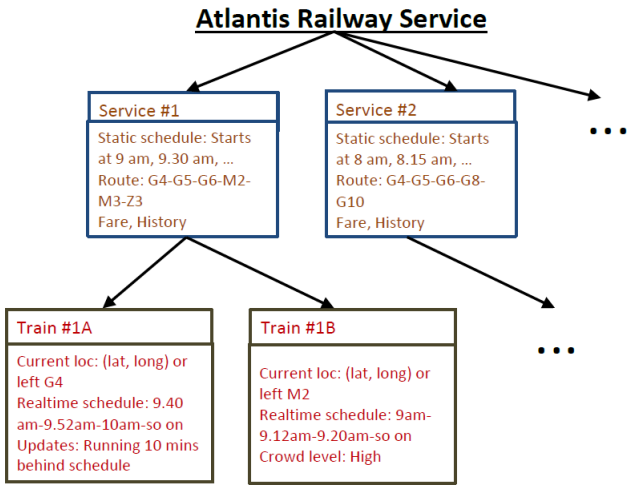

Fig. 5: Atlantis public transp. (rail) hierarchy

hierarchy. For instance, if a user requests for available routes to go from G2 to G6 in Gotham, location hierarchy gives three options (G2-G5-G3-G6, G2-G5-G7-G6 and G2-G4-G7-G6). Now based on the user preference, routes with least traffic or routes with least distance may be provided first in response to the user query. Traffic and distance information is available with the location hierarchy.

Location hierarchy not only represents geographical hierarchy and its transport networks, but also stores various attributes relevant to its elements enabling the system to resolve various user queries.

Public transportation hierarchies for Atlantis:: The information about public transportation system in Atlantis is maintained in two hierarchies - rail and bus transportation hierarchies. Figure **??** shows the rail transportation hierarchy for Atlantis. The bus public transportation system has a similar hierarchy.

In the middle layer of hierarchy, static information about all available services is stored. For example, the middle layer stores the static schedule, fare details, running history, etc. These details are common for all the children (lowest level elements) of a service. The lowest layer in the hierarchy represents the vehicles in transit and stores the real-time information about these vehicles, such as their current location, speed, crowding, real-time schedule, etc.

When any user submits journey details, the location of the train/bus is computed and updated in lowest level of the public transportation hierarchy. The user queries about the schedule of trains/buses, etc. can be handled using this information.

Fig. 3 also happens to show a train service #1 and two trains #1A and #1B serving this route. The details of the train service #1 and its two trains are shown in the public transportation hierarchy.

IV. Field Experimentation Results

We have developed a prototype implementation of the CARTS system. This includes the above hierarchical content naming framework, which is split across a client-side Android application, and a server-side data collection and query handling framework. Prior to a pilot deployment of the application, we wished to collect preliminary field data to validate the feasibility of the overall approach. We turn our attention to this next.

	Mumbai	Chandigarh
Trip description	IIT main gate from/to Mulund bus stop	ISBT 17 to High Court to PEC to PGI to ISBT 43 to ISBT 17
Trip distance	8.5 km (1-way)	6.45 km (circular)
# trips	12 (one-way)	12 (circular)
# bus stops	23	11
# traffic lights	11	8

TABLE I: Data collection at Mumbai, Chandigarh

A. Data Collection

We have collected data along specific routes in two cities: Mumbai, Chandigarh. We installed our Android application on a smart-phone, and started the data collection at the time of boarding a bus by pressing a button on the application (much like it would happen in real use). GPS and accelerometer sensor readings were recorded using the Android API.

We enhanced the application to manually mark the ground truth of bus stop and traffic light locations (through button press on the application), and used specific trips to collect this information at either city.

Overall information about the data collection at either city is given in Table I.

B. GPS Connectivity

We now characterize the availability of GPS connectivity in our data. Fig. 6 shows the percentage of time as well

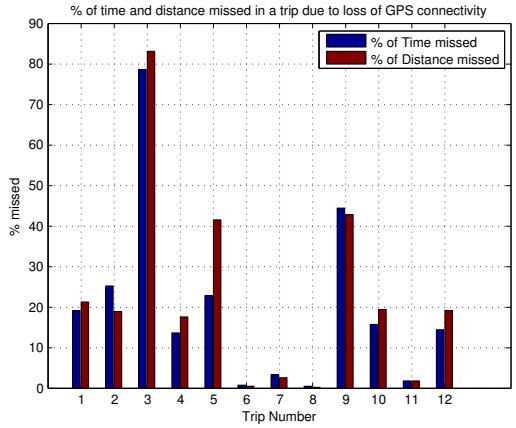

Fig. 6: Loss of GPS Connectivity

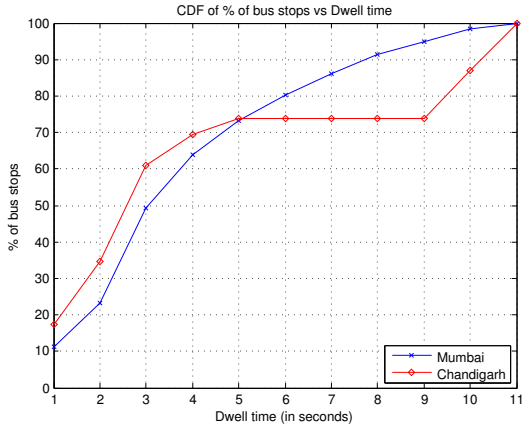

Fig. 7: CDF of dwell time at bus-stops

as distance for which GPS connectivity was missed, in our Mumbai trips. Overall, we see that GPS connectivity can be lost for a significant time. The mean loss in GPS connectivity is 18.4% of the entire journey time, and it is as high as about 80% for one of the trips.

In separate experiments, we found that this is not due to any specific model of smart-phone, or due to a specific cellular network provider (for assisted GPS). Nor did we find any specific correlation with the location at which GPS connectivity is lost. However, we did find a pattern in terms of *when* GPS connectivity is lost. We observed that GPS connectivity has a high chance of getting lost soon after the commuter has boarded a bus. To characterize this, we find that for 7 of 6x2=12 trips (58.3%) GPS connectivity is lost within the first 1 km (lost for a distance over 100 m) of the trip. Note that this *not* indicative of correlation with location, as we do not have similar loss of connectivity toward the *end* of the trip, in a round-trip data collection. The mean time of loss of GPS connectivity in the first 1 km is about 65 sec.

We observed a similar pattern in our Chandigarh data as well, although the loss of GPS connectivity was a bit lesser. The mean loss in GPS connectivity was for 9.6% of the overall journey time (standard deviation 11.1%). For 5 of the 12 circular trips (41.7%) GPS connectivity was lost within the first 1 km (lost for a distance over 100 m). And the mean time for which GPS connectivity was lost in the first 1 km was about 19 sec.

A plausible reason for loss of GPS connectivity at the beginning of a bus trip is that we suddenly move from an outdoor location to inside the bus, which somewhat resembles an indoor location, with poor line-of-sight to GPS satellites. We note that our data was collected at times of relative less bus crowd. If a bus were crowded, then the loss of GPS connectivity could be worse.

The overall implication of the above result is that we cannot rely entirely upon GPS connectivity for our data collection and subsequent algorithms (e.g. prediction of bus arrival time). We must explore possibilities such as: sharing of location information across commuters of the same bus, using GSM network based location when GPS is not available (WiFi-based location cannot yet be used in most places in India as WiFi deployment is not yet as prevalent).

C. Dwell Time at Bus Stops

We next characterize the dwell time of the bus at each of its bus stops (how long it stops). For this, we use the manually marked ground truth of the bus stop locations. We then get the dwell time metric by using the exact location of the bus stop from the ground truth data and then observing how long the bus stays stationary (accounting any error in GPS readings) in that location during the data collection trips. Fig. 7 plots the CDF of the dwell times, across all the bus stops. Separate lines are shown for Mumbai and Chandigarh.

The mean dwell time at Mumbai was 5.9 sec (standard deviation 2.9 sec), while the mean dwell time at Chandigarh was a similar value of 5.1 sec (standard deviation 3.6 sec). Such small numbers imply that even ignoring the dwell time from any bus arrival time prediction algorithm would not result in significant error. While it is true that we collected data during relatively less busy times, and that dwell times could be higher at peak times, it is also likely that they would anyway be over-shadowed by the travel time, which will also be higher at peak times.

D. Bus Stop Detection

We now explore using our data, the feasibility of an automated bus-stop detection mechanism. The work in [5] uses GPS traces to deduce road segments automatically. Likewise, given a collection of GPS traces over multiple trips, is it possible to deduce the locations of the bus-stops? We do a preliminary analysis toward this question now. To mark a particular instance of stationary-bus as a bus stop, we first apply a lower bound of movement under 20 m within 5 sec. We do this since even when the bus is

NumTrips used in algorithm	Precision	Recall	Recall for traffic lights
Fuzzy-Intersection, Mumbai			
1	0.20	0.61	0.36
2	0.31	0.65	0.36
3	0.40	0.26	0.09
4	0.40	0.09	0.09
Fuzzy-Union, Mumbai			
1	0.20	0.61	0.36
2	0.11	0.74	0.73
3	0.08	0.87	0.73
4	0.06	0.91	0.73
Fuzzy-Intersection, Chandigarh			
1	0.30	0.72	0.58
2	0.34	0.66	0.58
3	0.37	0.60	0.46
4	0.38	0.51	0.36
Fuzzy-Union, Chandigarh			
1	0.30	0.72	0.58
2	0.12	0.90	0.75
3	0.08	0.90	0.75
4	0.06	0.90	0.75

TABLE II: Precision and recall for the fuzzy-intersection and fuzzy-union algorithms (*NumTrips* is an algorithm parameter)

stationary, we could get an updated GPS reading from the Android API, due to inherent GPS error.

Just using automated bus stop markings from one trip, using the above mechanism could be error-prone, as after all, the bus remains stationary at various locations during a commute: at (various points in the queue behind) traffic signals, as well as during any traffic congestion. To combine information across trips, we consider two broad approaches: a fuzzy intersection, and a fuzzy union. The adjective "fuzzy" refers to the fact that across trips, we consider two locations marked as bus stops to be the same, if they are within 30m of each other. We now evaluate these two methods, using data from the two cities' trips.

Table II shows the precision as well as recall for the two algorithms, for the case of Mumbai as well as for Chandigarh. In the fuzzy-intersection and fuzzy-union algorithms, the following parameter is used: *NumTrips*, the number of trips over which the intersection or union is applied. The table shows the algorithm behaviour for values 1-4 of this parameter. In our ground truth, we had also collected information about traffic light locations along the routes we travelled. For comparison, the table also reports the recall in detection of traffic lights using the same two algorithms.

We see from the table that the behaviour of fuzzy-intersection and fuzzy-union with respect to the *NumTrips* parameter is the reverse of one another. With fuzzy-intersection, the precision improves with increase in *NumTrips*, while the recall drops. With fuzzy-union, the behaviour is the reverse. This is what we would expect.

We further observe that neither algorithm has a reasonable value of precision as well as recall, for any value of *NumTrips*. That is, both algorithms perform poorly at least in one metric in each case. This holds for the Mumbai data and the Chandigarh data. For fuzzy-intersection, the recall in the case of traffic lights is poor as well.

While the fuzzy-intersection and fuzzy-union represent only preliminary explorations, the overall implication of the above result is that automated detection of bus stops appears to be a tough problem, given the various factors introducing noise in the system: GPS errors, bus-stops getting confused with other traffic related stops, etc.

V. Conclusion

In this work we have considered the use of crowd-sourcing for collecting information about public transportation units in a road network, from commuters. We designed an intuitive and hierarchical naming framework to refer to the various entities involved. We believe that this will lead to a flexible and scalable implementation in practice. We have developed an early prototype of the system. The paper also presented analysis of data collected on roads from two cities, Mumbai and Chandigarh. Our overall results indicate that using GPS connectivity alone can be error prone in several situations due to loss of GPS connectivity. Next, we find that bus-stop dwell times are mostly under 5-10 sec. Finally, we presented results which point toward the challenge in automated detection of bus-stops from GPS traces. A more comprehensive implementation and real-user evaluation of the system is our immediate future work.

Acknowledgement: This work is undertaken as part of the project titled "CARTS: Communication Assisted Road Transportation Systems", supported by ITRA, Media Lab Asia.

References

[1] J. Biagioni, T. Gerlich, T. Merrifield, and J. Eriksson. EasyTracker: Automatic Transit Tracking, Mapping, and Arrival Time Prediction Using Smartphones. In *SenSys*, Nov 2011.

[2] P. Mohan, V. Padmanabhan, and R. Ramjee. Nericell: Rich Monitoring of Road and Traffic Conditions Using Mobile Smartphones. In *SenSys*, 2008.

[3] 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 *SenSys*, Nov 2009.

[4] A. Thiagarajan, L. S. Ravindranath, H. Balakrishnan, S. Madden, and L. Girod. Accurate, Low-Energy Trajectory Mapping for Mobile Devices. In *8th USENIX Symp. on Networked Systems Design and Implementation (NSDI)*, Boston, MA, March 2011.

[5] J. Yoon, B. Noble, and M. Liu. Surface street traffic estimation. In *MobiSys*, 2007.

[6] P. Zhou, Y. Zheng, and M. Li. How Long to Wait?: Predicting Bus Arrival Time with Mobile Phone based Participatory Sensing. In *MobiSys*, 2012.