Online Grid-Based Dynamic Arrival Time Prediction Using GPS Locations

Naveen Nandan

***Abstract*—Transportation modes are aplenty in today's urban environment. Commuters use public transport such as buses, trains, taxis, personal motor vehicles, walking, bicycles, etc. to travel between places. One of the major concerns for the people who rely on public transportation is the unavailability or inaccuracy of systems that predict the estimated arrival time or the schedule based on the current location of vehicles and the traffic situation. With the advent of technology, a large set of urban transportation operators have begun to use location reporting systems such as GPS devices on-board their fleet, with the primary purpose of monitoring and managing their fleet. This paper describes methods for predicting the arrival time taking advantage of the location reports from such devices. The system pipeline developed is based on a complex event processing engine within which an algorithm is implemented to continuously predict in real-time the estimated arrival time in an online fashion. The developed system in first phase is evaluated using a vehicle simulator that generates vehicle trajectories along real public transportation routes.**

***Index Terms*—Complex event processing, data stream mining, real-time distributed systems, spatial data mining.**

I. Introduction

Public transportation is one of the key enablers of city operations. Majority of the population in cities rely on different modes of public transportation for their daily commute. To quote the Mayor of the city of Bogota, Gustavo Petro [1], "A developed country is not a place where the poor have cars. It's where the rich use public transportation", which gives a clear indication that planners see the need to invest resources and time for building efficient transportation systems. Transportation operators and planners look towards technology and infrastructure, both hardware and software, for this purpose. This is not only to make their systems efficient, but also to improve the experience of the citizens.

It is agreeable that in most mega cities the transportation systems are quite well-developed. In fact, most public transportation operators already make available the timetables or schedule of their services for commuters on the web, through mobile apps or display boards at stations. But, it is often a case that over the day, the dynamics of the city change and there are unforeseen delays. Long waiting time tends to cause bad experience for commuters as they are either unaware of where the vehicles are currently, or there are not very accurate methods that use location reported from existing hardware such as GPS devices, mobile networks, etc. to predict the estimated arrival or travel time. Advancements in GPS technology enable location reporting with a high degree of accuracy, and this can be exploited to predict the arrival time.

Technology such as GPS devices are already being used by various operators for monitoring and managing their fleet in cities like Singapore, London, etc. This enables systems to collect the location information of vehicles in near real-time if sampled at higher rates. This tilts the spotlight towards being able to process and analyze all of this data on the fly. The paradigm of extracting knowledge from continuous, rapid, high-volume, highly-variable data streams is referred to as "Real-time Data Stream Mining". Through this paper, we discuss high-velocity data stream mining that can be applied to data generated from location reporting devices such as GPS within the context of urban transportation.

The system presented in this paper can perform real-time prediction of the arrival time of vehicles based on location data, which is processed by an event stream processing engine. The system is tested using a simulator built that uses real bus routes in Singapore to model the movement of vehicles and streams in the location information of these vehicles at high sampling rates.

In this paper, a novel technique to estimate the arrival time or travel time of vehicles in an online fashion is discussed. The system is designed to handle multiple streams of GPS data and continuously predict the arrival time for each vehicle.

II. Related Work

Zhou *et al.* [2] present a system that predicts bus arrival time using mobile phone signals. They use techniques such as cell-tower sequence matching that rely on the cell configuration of GSM network operators to compare the position of the bus to the actual bus route. This method is quite effective assuming there is not much handover between cell-towers, which is not a common case in urban environments where there is quite a high density of subscribers.

Yu *et al.* [3] discuss a prediction model for bus travel time based on support vector machine regression method. This approach introduces a forgetting factor to assign weights based on the recent data due to the bus running time-based variable quantities. They use Grubbs' test method to remove outliers from the input location data.

Pu *et al.* [4] discuss various literature that use artificial neural networks to estimate/predict travel time. They use a segment-based approach by dividing the street into smaller snippets and computing the average speed of the vehicle. This method suggests using buses as probes to estimate

Manuscript received September 20, 2013; revised November 19, 2013. This research was supported with the funding from the Economic Development Board and the National Research Foundation of Singapore.

Naveen Nandan is with SAP Research & Innovation, Singapore (e-mail: naveen.nandan@sap.com).

travel time along segments.

Another method makes use of GPS data to dynamically predict the travel time. This takes into account the travel time between two consecutive stops to estimate the arrival time at the next stop in sequence. Kieu *et al.* [5] discuss how Bluetooth and RFID can be used as a traffic data source.

There are many systems that, in practice, estimate traffic density based on loop detectors over various road segments and predict arrival time based on the estimates. This paper discusses a grid-based approach that uses GPS locations that help in reaching the granularity of road segments by limiting the area per cell or size of the cell.

III. Data Stream Mining

A lot of the research focus, effort and money are being invested in the area of data stream mining by both industry and academia to address the pressing needs of architecting real-time or close-to real-time systems.

STREAM: The Stanford Data Stream Management System [6] was built on top of contextual query language (CQL) focused on query optimization for memory management. The system made use of "synopses" to approximate results based on summarized information. TelegraphCQ developed at Berkeley [7] was an extension of PostgreSQL using a continuous querying mechanism of CQL type. This system addressed a key requirement of being able to add new queries dynamically. Aurora (superseded by Medusa, then Borealis) [8] brought in the new dimension of scalability through "distributed stream processing". Other systems that were similar in nature include NiagaraCQ [9] which is a scalable continuous query system for web-based (XML) databases, StatStream [10] for statistical monitoring of multiple data streams, StreamMiner [11] which is a classifier ensemble-based data mining engine, Gigascope – a Stream database [12] and Hancock (a C programming language variant) from AT&T was developed for mass surveillance of their massive networks.

The interesting thing about most of the above systems is that they gained attention from the big names in the market and hence, further development was shifted to the industrial research labs. Streambase is one such system that spun-off from Aurora and focuses on high-performance Complex Event Processing (CEP). Coral8 another famous CEP tool, designed on top of a publish-subscribe architecture, has been used widely by the likes of Microsoft and IBM. The engine that evolved from STREAM is maintained by SAP as part of the SAP Sybase Event Stream Processor. TelegraphCQ became widely popular with the telecommunication network sector, and was integrated by Cisco into their network management platform – Cisco Prime.

By convention, real-time data stream mining systems are built either as batch processing systems or, the more recent, continuous (real-time) processing frameworks.

Batch processing systems, such as "Hadoop" (from Apache Software Foundation and Cloudera), store/buffer raw data streams (files, web, images, etc.) over a period of time and process them at frequent intervals. This in turn is done in two stages where the input (files) is distributed over multiple machines and then a Map-Reduce operation is performed. The results are then pushed to a data-store or a subscribed client. To achieve close to real-time processing, the regular interval in which the input is distributed is narrowed down.

A couple of years ago, BackType (now acquired by Twitter) came up with a promising continuous distributed real-time processing framework called "Storm" which addresses most of the requirements for a system of its genre, such as reliability, robustness, fault-tolerance, scalability, etc. Pachube (now Cosm) is another system that was developed as a platform to address the idea of "The Internet of Things". This platform acquires data from various external devices and handles them in real-time. DataSift provides a social media data platform which brings in multiple data streams together and aids in performing analysis over them through the steps of extraction and reduction of data. Most of these systems use ZeroMQ and Apache Zookeeper for message queuing and coordination respectively. Other comparable systems are Esper, Streambase, Hstreaming and Yahoo S4. A system such as that of DataSift's has an estimated data volume that adds up to 1 TB each day.

For this system, we make use of the SAP Sybase Event Stream Processor for running the queries on the simulated vehicles GPS data streams.

IV. Online Grid-Based Dynamic Arrival Time Prediction Algorithm

The online grid-based dynamic arrival time prediction algorithm can be broken down into the following functional steps:-

1) Initializing the bound (global or local)
2) Splitting the grid (rows × columns)
3) Fitting the GPS location to a cell
4) Computing the time spent in a cell
5) Estimation of arrival time

A. Initializing the Bound (Global or Local)

One of the required inputs into the model is the bounds of the grid. As GPS locations can be seen as latitude/longitude points on the map, the grid can be bounded either by global constraints ($-180 < lon < 180$; $-180 < lat < 180$) or locally ($lon_1 < lon < lon_2$; $lat_1 < lat < lat_2$). For example, in the case of Singapore, the grid is locally bounded to ($103.62 < lon < 104.02$; $1.22 < lat < 1.48$) the geographic boundaries of the territory, as illustrated in Fig. 1.

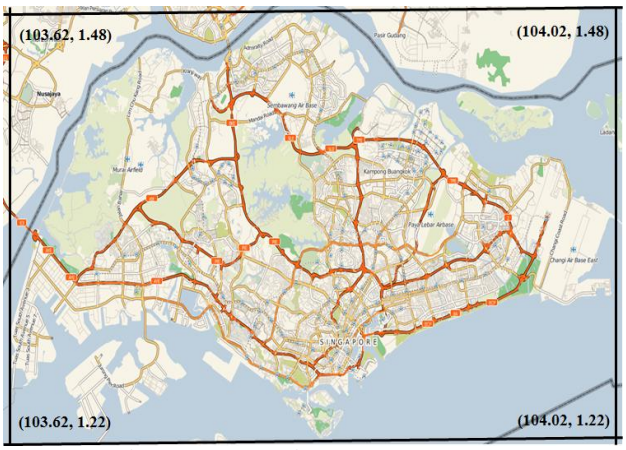
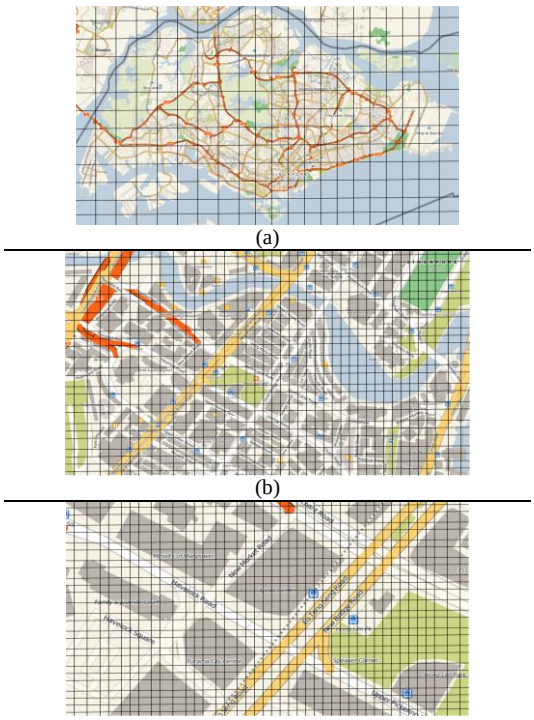


Fig. 1. Defining the bounds of the grid as the geographic extent.

B. Splitting the Grid (Rows × Columns)

The number of rows and columns are to be selected to split the grid into cells. The larger number of cells will result in higher granularity, thereby, resulting in more frequent updates in the predicted time.

As illustrated in Fig. 2, based on larger number of rows and columns, the coverage per cell is limited to a smaller area which gives higher precision or detail, almost up to the road segment level.



(a) (b) (c)

Fig. 2. Grid-view based on different splits. As the split results in larger number of cells, the level of detail increases.

C. Fitting the GPS Location to a Cell

With the input stream containing the GPS location of the vehicle, the current position (cell number) within the grid can be computed, as shown in Fig. 3, using the following method:

$$gridX = floor[\{(lat - lat_1) * rows\} / (lat_2 - lat_1)]$$
$$gridY = floor[\{(lon - lon_1) * columns\} / (lon_2 - lon_1)]$$
$$cell\_number = ((columns * (gridX - 1)) + gridY)$$

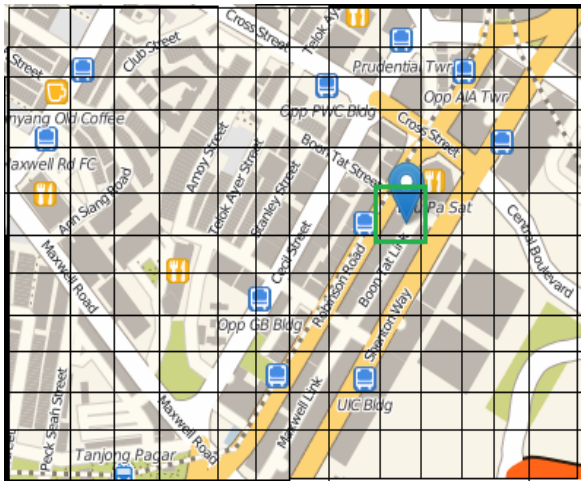


Fig. 3. Fitting the GPS location, indicated by the marker, to a cell.

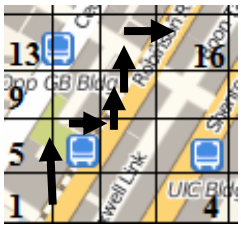
D. Computing the Time Spent in a Cell

Using the window operator of the complex event processing engine, we can accurately measure the duration of cell transitions, i.e. the time spent by a vehicle within a cell before moving to an adjacent cell. Let each cell transition from 'm' to 'n', as shown in Fig. 4, be denoted as '$C_{m \to n}$', the time taken for this transition be denoted as '$t_{m \to n}$' and the average time of all vehicles transitioning from '$m \to n$' be denoted as '$T_{m \to n}$'.

Fig. 4. Time windows are used to measure the duration for traversing from one cell to another ($C_{m \to n}$, $t_{m \to n}$).

E. Estimation of Arrival Time

The arrival time is computed and output every time the vehicle makes a cell transition. If the grid splitting is good (i.e. highly granular or contains a large number of cells), the transition is rather soon and therefore, the prediction algorithm returns the arrival time in a dynamic fashion.



2	6	7	11	15	16

Series of cell transitions

Fig. 5. Transition between cells as the vehicle moves along the route.

In any given grid, if the cells are numbered in a sequential fashion starting from the lower left, as shown in Fig. 5, the arrival time (T_{est}) is estimated using the given algorithm:

$T_{est} \leftarrow T_{route}$ *(scheduled arrival time)*
$C_{2 \to 6}: T_{est} \leftarrow T_{route} + (T_{2 \to 6} - t_{2 \to 6})$
$C_{6 \to 7}: T_{est} \leftarrow T_{est} + (T_{6 \to 7} - t_{6 \to 7})$
$C_{7 \to 11}: T_{est} \leftarrow T_{est} + (T_{7 \to 11} - t_{7 \to 11})$
$C_{11 \to 15}: T_{est} \leftarrow T_{est} + (T_{11 \to 15} - t_{11 \to 15})$
$C_{15 \to 16}: T_{est} \leftarrow T_{est} + (T_{15 \to 16} - t_{15 \to 16})$

To generalize,

$$C_{m \to n}: T_{est} \leftarrow T_{est} + (T_{m \to n} - t_{m \to n})$$

The stream processing engine sees the transitions between cells as an event, and the operator to estimate the arrival time is applied every time the transition event occurs. Hence, the system continuously predicts the estimated arrival time for the vehicle. The initialization of the estimate can be done either through scheduled arrival time (T_{route}), as illustrated above, if the schedule is available, or by computing the average for the specific route over a window of time by aggregating the time spent by all vehicles following a similar route or that made a similar cell transition.

V. Conclusion

In this paper an online grid-based algorithm for dynamically predicting the arrival time of public transport is presented. The initial system design and implementation is done in a complex event processing engine to ensure scalability when multiple GPS data streams are used in the future. The simulator built serves as a good test of the algorithm, but as an improvement the intention is to run and test the system with actual live GPS streams from vehicles on the road, when available. In future, the prediction accuracy could be improved by also analyzing historical trajectory data. As an extension, the system would be built in a generic manner to be able to handle spatial data streams from multiple different sources.

Acknowledgment

We would like to thank the Land Transport Authority (LTA) of Singapore for making the bus routes available through their Data Mall which has been used as a basis to model the vehicle simulator; the Economic Development Board and National Research Foundation of Singapore for supporting this research which can be extended and applied to any generic geo-spatial real-time application domains; my colleague Dr. Daniel Dahlmeier for pair programming on the vehicle simulator.

References

[1] Gustavo Petro. [Online]. Available: http://www.changemakrs.com/GustavoPetro

[2] P. Zhou, Y. Zheng, and M. Li, "How long to wait: Predicting bus arrival time with mobile phone based participatory sensing," *ACM MobiSys'12*, vol. 4, no. 3, pp. 33-39, June 2012.

[3] B. Yu, T. Ye, X. Tian, G. Ning, and S. Zhong, "Bus travel-time prediction with forgetting factor," *Journal of Computing in Civil Engineering*, November 2012.

[4] W. Pu, J. J. Lin, and L. Long, "Real-time estimation of urban street segment travel time using buses as speed probes," *Transportation Research Record*, January 2010.

[5] L. M. Kieu, A. Bhaskar, and E. Chung, "Bus and car travel time on urban networks: integrating bluetooth and bus vehicle identification data," in *Proc. 25th ARRB Conf.*, 2012.

[6] A. Arasu, B. Babcock, S. Babu, J. Cieslewicz, M. Datar, K. Ito, R. Motwani, U. Srivastava, and J. Widom, "STREAM: The Stanford data stream management system," Technical Report, Stanford InfoLab, 2004.

[7] S. Chandrasekaran, O. Cooper, A. Deshpande, M. J. Franklin, J. M. Hellerstein, W. Hong, S. Krishnamurthy, S. Madden, V. Raman, F. Reiss, and M. Shah, "TelegraphCQ: Continuous dataflow processing for an uncertain world," in *Proc. 1st Biennial Conf. on Innovative Data Systems Research*, Asilomar, CA, January 2003.

[8] M. Cherniack, H. Balakrishnan, M. Balazinska, D. Carney, U. Cetintemel, Y. Xing, and S. Zdonik, "Scalable distributed stream processing," in *Proc. 1st Biennial Conf. on Innovative Data Systems Research*, Asilomar, CA, January 2003.

[9] J. Chen, D. J. DeWitt, F. Tian, and Y. Wang, "NiagaraCQ: A scalable continuous query system for internet databases," *ACM SIGMOD Record*, pp. 379-390, May 2000.

[10] Y. Zhu and D. Shasha, "Statstream: Statistical monitoring of thousands of data streams in real time," in *Proc. 28th VLDB Conf.*, August 2002, pp. 358–369.

[11] W. Fan, "Streamminer: a classifier ensemble-based engine to mine concept-drifting data streams," in *Proc. 30th VLDB Conf.*, August 2004, pp. 1257–1260.

[12] C. Cranor, T. Johnson, O. Spatscheck, and V. Shkapenyuk, "Gigascope: A stream database for network applications," *ACM SIGMOD Record*, pp. 647-651, June 2003.

Naveen Nandan is from India and was born on March 6, 1988. He received a bachelor of Technology in Electronics and Communication Engineering from Amrita Vishwa Vidyapeetham, Coimbatore in 2009. He continued to pursue Master of Science at the School of Computer Engineering, Nanyang Technological University, Singapore where his thesis work was on "Multi-class Emotion Detection from Suicide Notes" applying techniques from natural language processing for feature extraction, association rule mining for feature selection and machine learning for classification.

He started off his career as a systems engineer with the Education & Research Department at Infosys, Mysore. He spent a brief stint as a systems engineer with JamiQ.com, a social media intelligence startup in Singapore, after which, he worked as a software engineer (PLATFORM) on the LIVE Singapore project with the MIT SENSEable City Lab at the Singapore-MIT Alliance for Research and Technology under the Future Urban Mobility program. He currently works for SAP Research & Innovation, Singapore as a research & development engineer. His current research focus is in the areas of distributed & real-time systems, machine learning and spatial data mining.