CS4440
Emerging Database Technologies
Project Proposal
October 2, 2007

Bus Prediction Algorithm Evaluation

John Abraham
Galen Hussey
Matt Weber

1. Introduction

Providing accurate and timely information about bus arrival is important for Intelligent Transportation Systems (ITS) which use Automatic Vehicle Location (AVL). It drastically enhances the user experience while at the same time providing the operators with information about lagging or leading buses which can then take corrective action. Our objective in this project will be to utilize historical and current GPS records of the Georgia Tech Bus system to implement and compare algorithms for better bus arrival-time prediction.

2. Related Work

This project will attempt to build upon two existing projects, BuzzRoute and the SMS bus tracker. Here we briefly summarize both projects. We then discuss the related the current arrival time prediction service employed by GA Tech NextBus.

BuzzRoute is a project to provide a visual interface to the bus locations on a mobile device. BuzzRoute itself was built on a previous prototype that successfully implemented the mobile client and architecture, improving the architecture in some places, and adding a route detection component. Much of the code used in BuzzRoute is relevant to our bus prediction, so portions will be re-used in our application.[6]

SMS bus tracker was a project to provide an SMS-based query response interface to the NextBus (described later) predictions. Unfortunately, this project was not completed, but it has plenty of code relevant to our application. We hope to use this project as our interface to the bus prediction algorithm.

To our knowledge, the only service that attempts to predict the arrival time of a Georgia Tech bus is NextBus, which uses the existing schedule of the bus system, as determined by the transportation directory, and simply predicts the arrival time as the scheduled time of arrival. This system works well in cases where a given bus is either running early or on schedule, because the drivers are required to delay their progress to allow the schedules to catch up. This system's obvious flaw is in cases where the buses are running behind schedule, since there is little a bus driver can do to compensate for most causes of delay, such as traffic or large numbers of people entering or leaving the bus.
The bus schedules are broken into blocks for each route. Each bus on a single route is given its own block, which has static times for its arrival at each stop. The tables below show samples of the block schedule for the Green route, effective August 2007.

Block 1	Lv. TEP	Lv. CRC	Lv Hemphill & 10th St.	Ar. GTRI	Lv. GTRI	Lv. Hemphill & 10th St.	Lv. S. Cntr	Ar. TEP
Garage								
7:00 AM	7:15 AM	7:19 AM	7:21 AM	7:25 AM	7:30 AM	7:35 AM	7:38 AM	7:42 AM
1	7:45 AM	7:49 AM	7:51 AM	7:55 AM	8:00 AM	8:05 AM	8:08 AM	8:12 AM
1	8:15 AM	8:19 AM	8:21 AM	8:25 AM	8:30 AM	8:35 AM	8:38 AM	8:42 AM
1	8:45 AM	8:49 AM	8:51 AM	8:55 AM	9:00 AM	9:05 AM	9:08 AM	9:12 AM
1	9:15 AM	9:19 AM	9:21 AM	9:25 AM	9:30 AM	9:35 AM	9:38 AM	9:42 AM
1	9:45 AM	9:49 AM	9:51 AM	9:55 AM	10:00 AM	10:05 AM	10:08 AM	10:12 AM
1	10:15 AM	10:19 AM	10:21 AM	10:25 AM	10:30 AM	10:35 AM	10:38 AM	10:42 AM
1	10:45 AM	10:49 AM	10:51 AM	10:55 AM	11:00 AM	11:05 AM	11:08 AM	11:12 AM

Block 2	Lv. TEP	Lv. CRC	Lv Hemphill & 10th St.	Ar. GTRI	Lv. GTRI	Lv. Hemphill & 10th St.	Lv. S. Cntr	Ar. TEP
Garage								
8:00 AM					8:15 AM	8:20 AM	8:23 AM	8:27 AM
	8:30 AM	8:34 AM	8:36 AM	8:40 AM	8:45 AM	8:50 AM	8:53 AM	8:57 AM
2	9:00 AM	9:04 AM	9:06 AM	9:10 AM	9:15 AM	9:20 AM	9:23 AM	9:27 AM
2	9:30 AM	9:34 AM	9:36 AM	9:40 AM	9:45 AM	9:50 AM	9:53 AM	9:57 AM
2	10:00 AM	10:04 AM	10:06 AM	10:10 AM	10:15 AM	10:20 AM	10:23 AM	10:27 AM
2	10:30 AM	10:34 AM	10:36 AM	10:40 AM	10:45 AM	10:50 AM	10:53 AM	10:57 AM
2	11:00 AM	11:04 AM	11:06 AM	11:10 AM	11:15 AM	11:20 AM	11:23 AM	11:27 AM
2	11:30 AM	11:34 AM	11:36 AM	11:40 AM	11:45 AM	11:50 AM	11:53 AM	11:57 AM

Another flaw NextBus suffers from is in the bus system itself. In order for the route to be reported from the bus correctly, the bus driver must enter the route code into a keypad interface. Due to intermittent hardware problems, this is sometimes not accomplished. In this case, the bus location is still reported, but its route is not. When this happens, NextBus simply discards the updates for that bus, and predictions do not take that bus into account. It is worth noting that we will be using the same data stream for our predictions as NextBus.

3. Proposed Work

Upon completion, we hope to have evaluated two of the latest bus prediction algorithms and implemented one for the final application. To do this, we have acquired access to bus location update archives going back over 3 years, as well as source code for both the BuzzRoute and SMS bus tracker projects mentioned above. We have selected two bus-prediction algorithms to evaluate, one using a Kalman filter and the other a neural network. We have chosen to use MySQL as our spatial database solution and Java as our primary coding language. The rest of this section will explain these components in more detail.

Bus Location Archives:
Each bus, on the Georgia Tech transit system, is equipped with a GPS receiver and a cellular chip. Every 7 to 15 seconds the bus uploads its location to a server. This server combines the updates from all of the buses into a stream that NextBus accesses. In addition to being streamed to NextBus, the updates are also written to a daily log for archiving. As a result we have bus update records from as early as December of 2003.

This is a sample update with an explanation of each line:

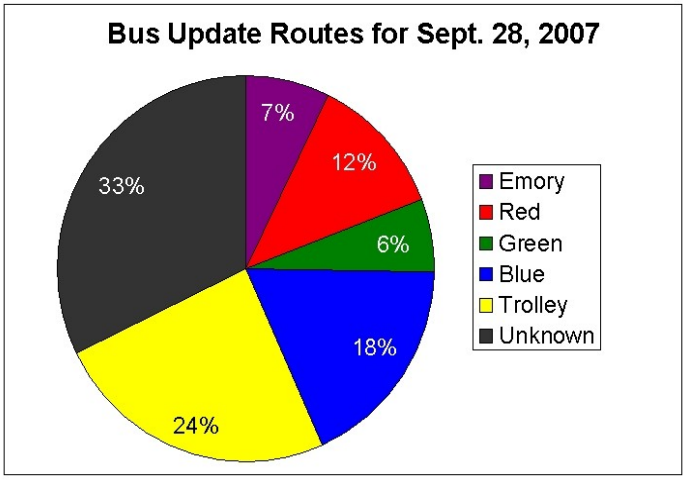
1. `$GPRMC,050010.923,A,3346.6145,N,08423.6269,W,0.00,90.58,030107,,*2E // NMEA sentence`
2. `I=1 // bus ignition status`
3. `U=0 // deprecated and will not be used`
4. `VID=810 // vehicle id`
5. `RSSI=-71 // cellular signal strength`
6. `NU=1:18:00 // time up on the network`
7. `ND=0 // times rebooted the network card`
8. `J=0202 // route id and block number`

```
Line one is a NMEA sentence used by most GPS receivers. It is broken down as follows:
RMC - Recommended Minimum Navigation Information
                                                      12
       1         2 3       4 5        6  7   8    9    10 11|  13
       |         | |       | |        |  |   |    |    |  | |   |
$GPRMC,hhmmss.ss,A,llll.ll,a,yyyyy.yy,a,x.x,x.x,xxxx,x.x,a,m,*hh<CR><LF>
Field Number:
  1) UTC Time
  2) Status, V=Navigation receiver warning A=Valid
  3) Latitude
  4) N or S
  5) Longitude
  6) E or W
  7) Speed over ground, knots
  8) Track made good, degrees true
  9) Date, ddmmyy
 10) Magnetic Variation, degrees
 11) E or W
 12) FAA mode indicator (NMEA 2.3 and later)
 13) Checksum
```

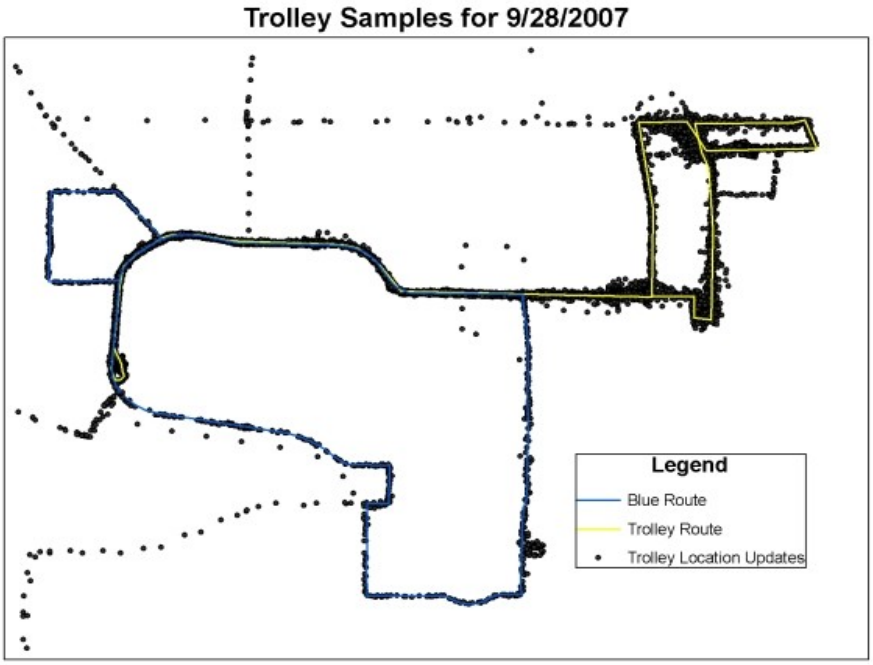
Issues:
There is ample information from each update to complete our project. However, there are a number of

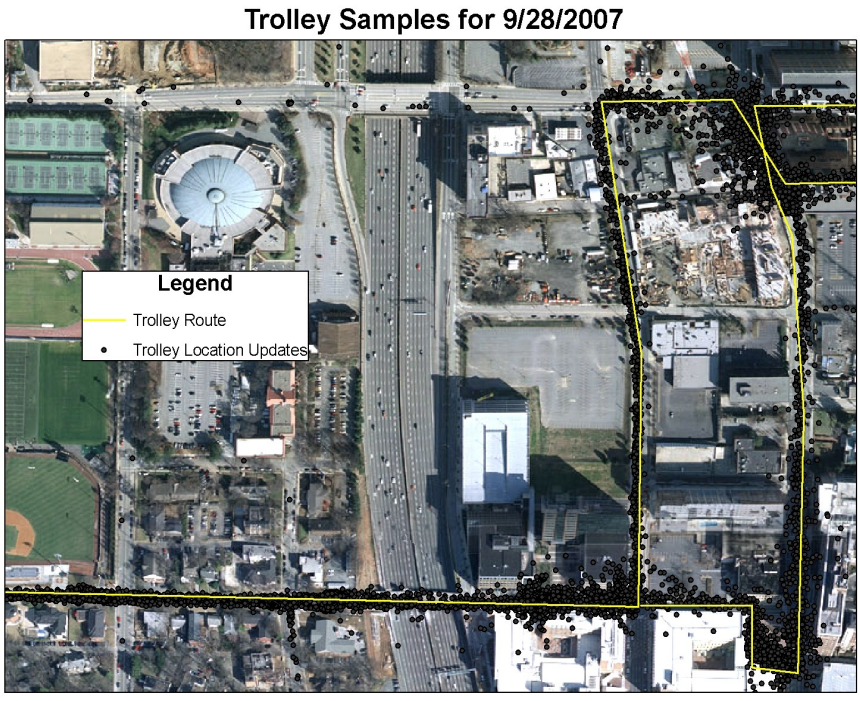
issues that must be addressed before the data can be effectively used for predictions.

The first problem we must overcome is with the route designations. As mentioned above, the route codes are not always entered, resulting in updates not associated with any route. NextBus' solution is to disregard these updates, but there are often a large number of updates that fall into this category. For example, the pie chart to the right depicts the percentages of updates associated for each route on Friday September 28, 2007. As illustrated, 33 percent of almost 90,000 updates were not associated with a route.



Another related problem concerns the wrong route being reported. If a bus' route is changed during the day, and the route code is not updated, it will report being on a route different than the one it is actually serving. For example, the image to the left shows all of the location updates associated with the Trolley route on Friday September 28, 2007. It is evident that not all of the updates were actually on the reported route, rather, it seems the bus was on the Blue route for a period of time. For the two previous problems, we intend to improve the dataset by developing a way to determine the routes these updates were on, based on their motion patterns.



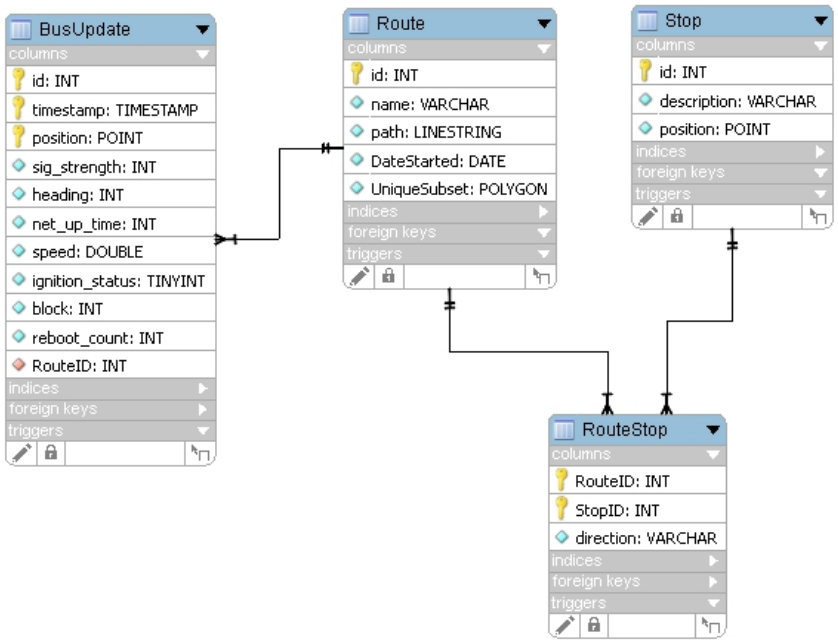


The next problem that must be dealt with is caused by GPS read errors. In most areas of the bus system, the accuracy of the samples appears to be within the Differential GPS error range of one to five meters, but in areas near tall buildings the error range becomes much larger. For example, the image to the left shows a portion of the Trolley route where this is the case. In the lower left region, we see that the samples are fairly accurate - this is the accuracy we have for most of the samples. As we follow the route near tall buildings, shown in the lower and upper right corners of the image, it is apparent that the accuracy decreases considerably. To deal with this problem, we intend to develop a way to snap the samples to the appropriate points on the route's path.

The final problem with the data is that much of the samples are not on any route. When the buses begin or end for the day, are driven to a maintenance shop, or are refueled, the updates do not stop. This can easily be dealt with by throwing out data not following the pattern of any route.

Database:

MySQL 5.0 supports spatial queries in the form of spatial extensions outlined in the MySQL manual.[5] We have designed our database using the basic spatial types to leverage this spatial query support in any way that may be necessary. We have also added fields to store all data found in the bus updates, regardless of its relevance to our project, so that it can be of use to future studies. To the right is our schema diagram. Our next step will be to modify the code from the BuzzRoute project, used to parse the data archives, to populate our database.



Kalman FIter:

a. What is a Kalman filter:

The Kalman filter is essentially a set of mathematical equations that implement a predictor-corrector type estimator that is *optimal* in the sense that it minimizes the estimated *error* covariance—when some presumed conditions are met.

b.Why choose a Kalman filter:

The Kalman filter attempts to use all information provided to it. It processes all available measurements, regardless of their precision, to estimate the current value of the variables of interest, with use of

- The knowledge of the system and measurement device dynamics.
- The statistical description of the system noises, measurement errors, and uncertainty in the dynamics model.
- any available information about the initial conditions of the variables of interest.

The Kalman filter, being recursive, does not require all previous data to be stored and reprocessed every time a new measurement is taken. This is important for quick predictions of Bus-arrival times.

c. Operation of a Kalman filter:

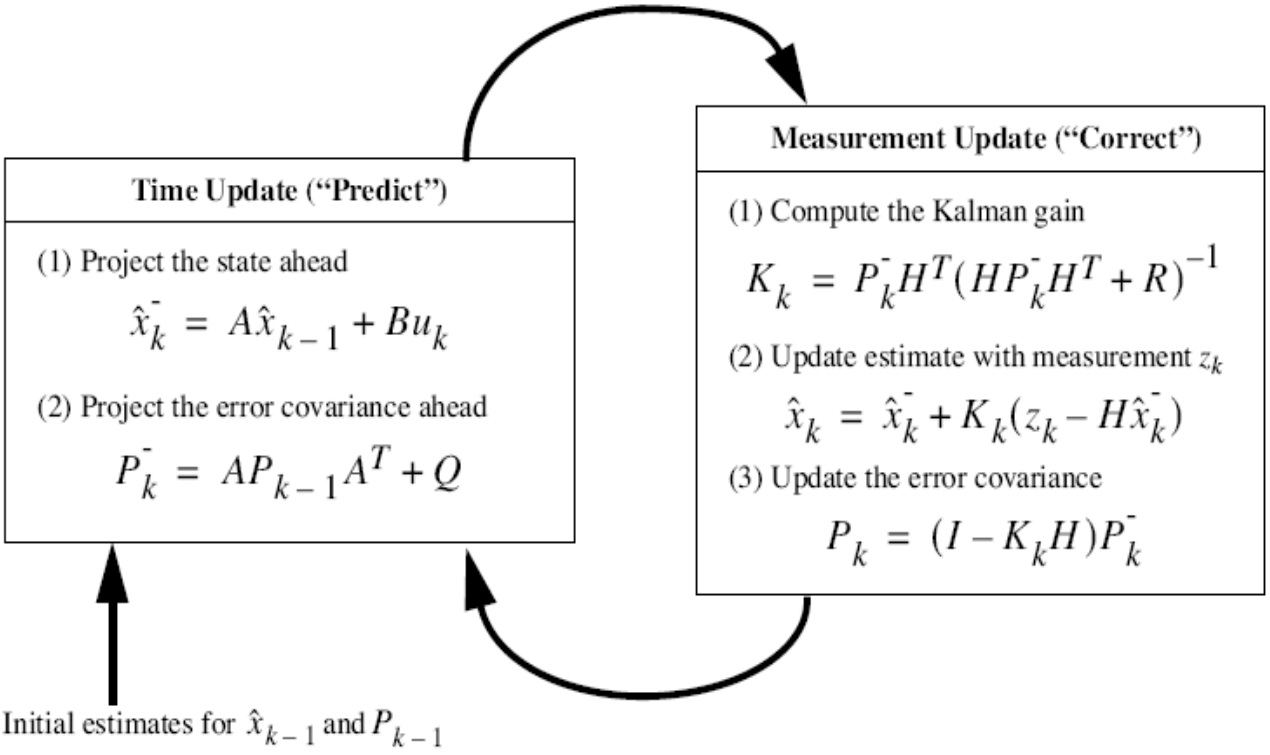
The Kalman filter estimates a process by using a form of feedback control: the filter estimates the process state at some time and then obtains feedback in the form of (noisy) measurements. As such, the equations for the Kalman filter fall into two groups: time update equations and measurement update equations. The time update equations are responsible for projecting forward (in time) the current state and error covariance estimates to obtain the a priori estimates for the next time step. The measurement update equations are responsible for the feedback—i.e. for incorporating a new measurement into the a priori estimate to obtain an improved a posteriori estimate. The time update equations can also be thought of as predictor equations, while the measurement update equations can be thought of as corrector equations.

The Kalman filter maintains the first two moments of the state distribution (the terms X_k,P_k are described later)

$$E[x_k] = \hat{x}_k$$

$$E[(x_k - \hat{x}_k)(x_k - \hat{x}_k)^T] = P_k$$

For brevity we represent the algorithm as a diagram



Where

$\hat{x}_k^-$ a priori state estimate at step k

$\hat{x}_k$ a posteriori state estimate at step k given measurement

$e_k^- \equiv x_k - \hat{x}_k^-$ a priori estimate error

$e_k \equiv x_k - \hat{x}_k$ a posteori estimate error

$P_k^- = E[e_k^- e_k^{-T}]$ a priori estimate error covariance

$P_k = E[e_k e_k^T]$ a posteori estimate error covariance

$(z_k - H\hat{x}_k^-)$ residual or innovation

A relates the state at the previous time step to the state at the current step

B relates the optional control input to the state x

Q process noise covariance

R measurement noise covariance

H relates the state to the measurement zk

u_k optional control input

K Gain or blending factor

K is chosen so as to minimize the a posteriori error covariance
The residual mentioned above indicates the discrepancy between the prediction and actual measurement.

d. Applying the Kalman filter, Experimental Parameters and Issues:

Most of our current understanding of the Kalman filter's feasibility is based on [1]. However, their assumptions maybe different from ours, for instance they assume that the vehicles move with constant speed over a limited distance, the variability of the process model is normally distributed at all times. At present we will use the approach they used, i.e, initially divide the route into shorter segments where linearity can be assumed and hence the time to arrival can be calculated linearly as Time = distance/velocity (Since the GPS records also indicate instantaneous velocity). This is obviously unrealistic and the actual measurements will deviate from the linear estimate. We aim to characterize this deviation as the "Process noise" distribution (from our historical data) rather than use normal distribution. Further ways to characterize process noise can be obtained from mining the distribution of the dwell time at the bus stops, positive or negative delay in meeting the bus schedules, etc. We may also take into account GPS read errors to characterize the measurement noise. Once we obtain a distribution for process and measurement noise we apply them to the Kalman Filter to obtain the time update and measurement update equations, as described in the diagram above. In our case , the state variables are vehicle location, time and time until arrival. The observables are the reported position, reported time of measurement and velocity. We contacted the authors of [1] to know how they characterized noise and are awaiting their response.

Neural Network:

a. What is a Neural net

Neural nets emulate the learning process of the human brain. They are good at pattern recognition, prediction, classification, etc. In essence, a neural net is a multi-layer set of feedback loops. Individual neurons, made up of simple equations, can perform very complex compensation/prediction calculations in a group.

b. Why choose a Neural net

The dynamic nature of the transportation environment is ideally suited to a neural network approach. Traffic delays, route changes, season, time of day - they are all factors that significantly influence transit and arrival times. The advantage of a neural net is that we can feed it all these many inputs, and, without necessarily devising equations to map all the relations of the inputs to arrival times, we can have the net figure out the mappings on its own.

c. Elements of a Neural net:

A typical neural net is composed of three layers - the input layer, the hidden layer, and the output layer. These layers are made with multiple neurons. An "oracle" is used to provide accuracy feedback on the net's predictions.

d. Applying a Neural net:

The nets are calibrated using two steps - training and testing. During the training stage, the net uses inductive learning principles to learn from a training data set. There are two types of learning techniques used in net development: unsupervised and supervised. In unsupervised learning, the network attempts to classify the training set data into different groups based on input patterns. In supervised learning, the desired output from output layer neurons is known, and the network adjusts the weight of connections between neurons to produce the desired output. During this process, the error in the output is propagated back from one layer to the previous layer by adjusting weights of the connections. Our implementation will probably involve supervised learning with archived bus data, and then we'll switch over to unsupervised to let the system handle streamed data. We'll input all the known variables and their current states, and use historical data to gauge the net's accuracy.

e. Experimental parameters and issues :

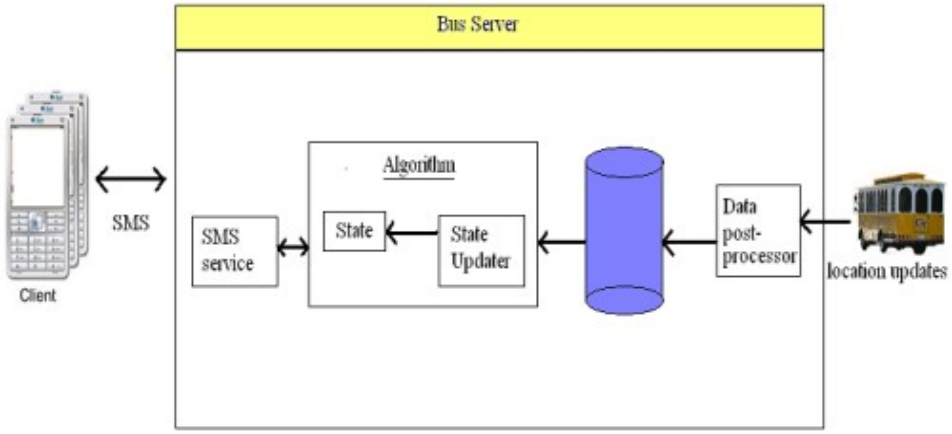
In the neural net prediction paper we examined[2], Automatic Vehicle Location (AVL) data collected in Houston, Texas was used for the test bed. The Houston data was collected by Houston Metro buses equipped with DGPS receivers at 5 second intervals. The data was collected over 6 months in 2000 (from June to November). The test bed was Route 60, which is highly congested in the morning

and afternoon peaks, and only the southbound direction was studied. This DGPS provides time, speed, heading, etc as well as bus location. The only points they considered in their prediction algorithms were the bus stations, which we think limits the effectiveness for routes that share pathways. The data stream we have is quite similar to theirs, although it sounds like we have more complex routes to consider.

The two major factors they attempted to measure were traffic congestion and dwell time at the stops. It's more difficult for us to get accurate dwell time measurements, but if we segment the bus routes properly, we should be able to get fairly detailed traffic congestion mappings. In the Houston case, heavy congestion made the 5-second updates more useful, since the slow speed would provide more measurement points per unit of distance. Our situation will not be so fortunate, as most of the bus routes do not typically remain in bumper-to-bumper traffic for long periods of time.

System Architecture:

Our system architecture is a modification of the architecture used in BuzzRoute. The modified architecture is illustrated to the right. Most of the components will reside on the bus server, gump.gatech.edu, hosted by OIT. As location updates arrive from the buses they will go through our post-processing algorithm to ensure they are helpful to our application. The prediction algorithm, that we ultimately choose, will periodically update its state from this database. The SMS bus tracking application, which will act as our user interface, will listen for SMS requests for a specific bus/stop pair, call the prediction algorithm, and return the results as a text message.



Milestones:

Assuming an eight week schedule, our milestones are as follows:

Week	Milestone
1	-Database population code complete
2	-Algorithm parameters modeled
3	-Data post-processor complete
4	-Algorithms implemented -SMS service up
5	-Simulation and evaluation planning period
6	-Evaluation complete
7	-Analysis of results -Presentation planning period
8	-Final presentation

Evaluation:

Evaluation of our results will be crucial to the project’s success. As we obtain a better understanding of the algorithms and their implementations, we will be better prepared to intelligently design our evaluation method. In general, we will compare results from each algorithm's prediction to actual bus arrival times and use the results as a metric of the algorithm’s accuracy. It is important to note that the evaluation is the primary objective of this project. The integration of the most accurate algorithm into the final application is secondary. As such, our evaluation will be entirely on the accuracy of the algorithms and not on the performance of the end application. We will consider this project a success when we are confident that we have correctly implemented the Kalman filter and the neural net, compared their performance, and declared a winner between the two.

Works Cited:

[1]. Dailey, D., Wall, Z., Mclean, S., Cathey, F., 2000. An algorithm and implementation to predict the arrival of transit vehicles. In: Proceedings of the
IEEE Intelligent Transportation Systems Conference.

[2]. Jeong, R. and L.R. Rilett. Bus arrival time prediction using artificial neural network model. in Proceedings - 7th International IEEE Conference on
Intelligent Transportation Systems, ITSC 2004. 2004. Washington, DC, United States.

[3]. Welch, Greg, and Gary Bishop. Introduction. An Introduction to the Kalman Filter. Chapel Hill: University of North Carolina At Chapel Hill, 2006. 2 Oct.
2007

[4]. Lin, Wei-Hua and Zeng, Jian " Experimental Study of Real-Time Bus Arrival Time Prediction with GPS Data". Transportation Research Record 1666, PP 101-109,
1999

[5]. "MySQL 5.0 Reference Manual :: 17 Spatial Extensions." MySQL 5.0 Reference Manual. 2 Oct. 2007 <http://dev.mysql.com/doc/refman/5.0/en/spatial-extensions.html>.

[6]. "BuzzRoute." 2 Oct. 2007 <http://gump.gatech.edu/buzzroute/doc_root/>