

Matt Weber
mattweb@gatech.edu

May 2nd, 2008
CP8900 Final Report
Dynamic Public Transit Routing

Introduction

As the demand for public transportation increases with the price of gas and concerns about our environment, one might wonder why the demand was so low to begin with. Atlanta has long been a leader in high commute times and congestion in the US and the available public transportation is capable of caring users from metro areas, in all directions, to Atlanta and back. One possible reason for this could be the ease of use. Though the transportation networks are connected well enough for reasonable transitions from one provider to another, ie: from Cobb to MARTA, the maps are maintained by different agencies and often in different formats. This makes it difficult to determine an itinerary that spans multiple transportation providers. For example a route from Woodstock to the CGIS first requires a drive to the park-and-ride on Busbee parkway in Woodstock where a Cobb County Express bus will lead to MARTA which can then be taken to the Tech Trolley. Of course the Cobb Express buses only run ever two hours so synchronizing a schedule between these providers, to reach the CGIS by a specific time, requires even more research. While this route may be worth deriving manually for a daily commute, for a single trip it is simply not worth it. The freedom of a vehicle always seems to prevail.

Now personal GPS-based navigation devices have made driving even more convenient, by giving the user real-time directions to their destinations, and even correcting them dynamically when a wrong turn is taken. There is no technological barrier that prevents this same technology from being applied to public transportation. With such an application in place users could enjoy an enhanced sense of freedom and confidence when taking public transportation. This is the motivation for the project described below.

As mentioned in the proposal, such an application is too complex to implement in a single semester. However two fundamental components are not, a mobile remote tracking application and an integrated transit routing application. The progress, current state, and future work for these components is described in this report.

Remote Tracking

Summary

The remote tracking application will allow the user to periodically upload their location to a remote server from a mobile device. The application has three major components, a mobile app, a web service, and database. The mobile app is written in J2ME, mobile Java, and employs a distance based dead-reckoning (DR) bandwidth reduction scheme. The web service is written in Java and was tested on an Apache Tomcat servlet engine. The database is written in MySQL.

Database

Figure 1 shows the ER diagram for the database. This was implemented on MySQL 5.0 using spatial extensions. Users must be created before uploads can be sent from the mobile

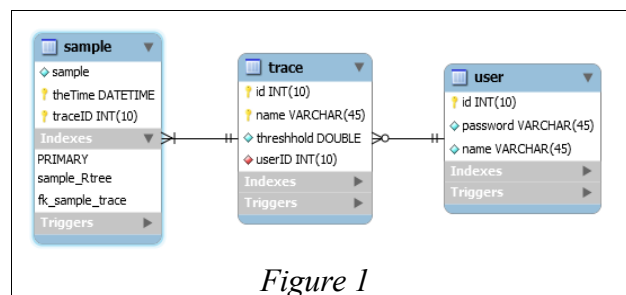


Figure 1

client which populates the trace and sample tables. Each sample is a location update from the client and each trace is a list of samples. The current web service only populates these tables but how an implementation could query them to retrieve the predicted location of the user is discussed later in the DR section.

Mobile Client

The mobile client application was written in J2ME and tested on Sun's Wireless Toolkit emulator. It requires JSR-172 (web service package) and JSR-179 (location package) in addition to standard mobile Java virtual machine. It has a minimal user interface as it is a tracking component of larger application, yet to be implemented. Figure 2 shows the only screen worth noting. The trace name field is the name of the location history that the user would like to store in the remote Db, the threshold is the DR error tolerance parameter discussed later, and the user name and password are those stored on the remote Db.

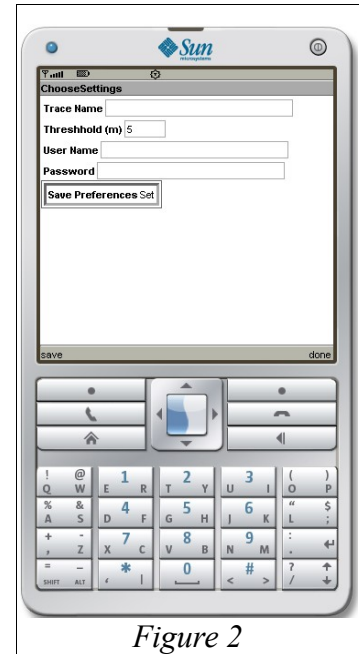


Figure 2

Web Service

The web service is a basic sample upload interface for the mobile app to send location updates to the Db. It does no complex calculations and offers only a basic authentication scheme. Figure 3 shows the methods exposed to the web. Both take a user

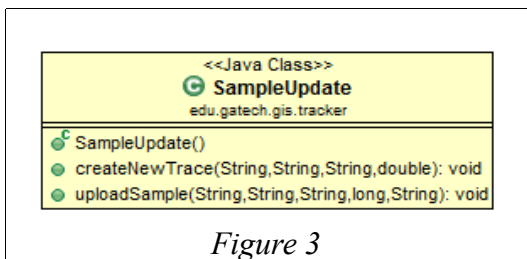


Figure 3

name and password. It is important to note that these are clear text and

should not be considered secure. The current implementation is over HTTP but should be migrated to HTTPS, or a custom encryption scheme implemented, before this service is published to a server with ports open to an untrusted network.

Dead Reckoning

With DR, the client and server agree on a user location prediction scheme with a given error tolerance. Each time the client receives a location update it makes the same prediction about the user's location that the server would make at that time, it then compares that prediction to the actual location. If the prediction is off by more than a given error tolerance, the client uploads that position, otherwise it knows that the server's prediction is close enough. In general, this approach reduces bandwidth use considerably at the expense of processor load, even with a low error tolerance. It also allows any trace, or user's location history, to be stored with much less memory. With DR the server is able to query the user's location at regular intervals, with a bounded accuracy.

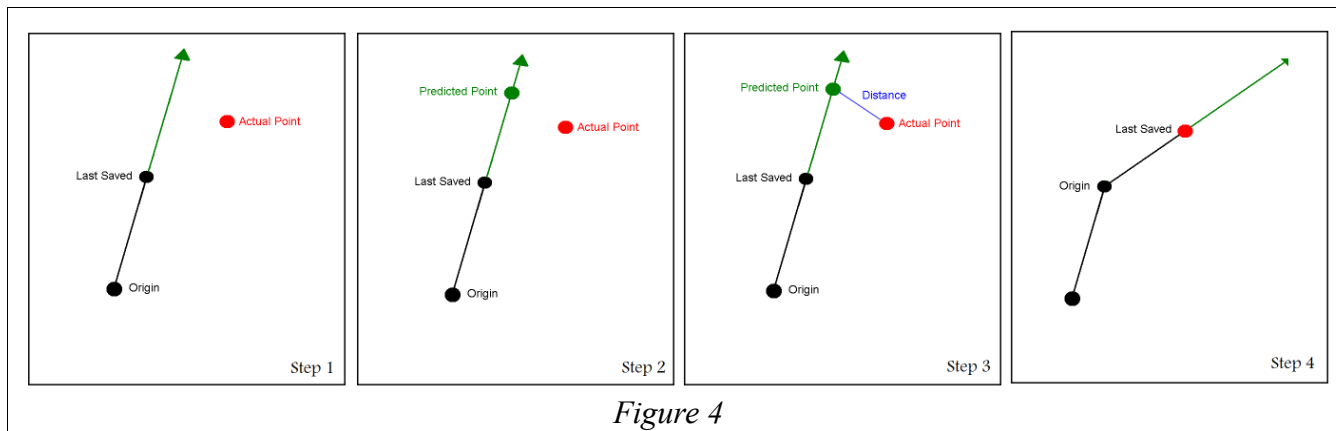


Figure 4

The current implementation is a distance based approach. Figure 4 Shows the four basic steps. Initially the first two samples of the trace are uploaded and stored in memory on the client side and a vector is drawn from the first to the second point, shown as a green arrow in step one. On the client, each time a location update is received, from GPS in this case, the location of the user is predicted as being where they would be, if they had maintained a constant speed and direction, at the same time that the actual GPS update was received, shown in step two. The distance between the predicted point and the GPS point is then measured, shown in step 3. Finally, if that distance is greater than a pre-defined error threshold, mentioned above, then the actual location is uploaded and the state of the algorithm is updated locally, shown in step 4, otherwise the GPS point is discarded and nothing is updated. This approach can also be thought of as a real-time line simplification algorithm.

The main advantage to DR, aside from bandwidth reduction and battery conservation, is that at any time, the location of the user can be predicted by the server, using the same method as the client, and we know the prediction is within the defined threshold of accuracy. It should also be noted that even at very low threshold values, considerable savings can be achieved with bandwidth and storage cost of the entire trace.

Transit Routing

Achieving the multi-modal routing component was more difficult than previously anticipated. Actually getting a prototype system is not difficult, but designing it to persist the frequently changing transportation routes among the various transportation departments has been a challenge. Because of this two avenues were explored, adding the GA Tech bus routes to Google Transit and building a network dataset to be published from our own ArcGIS server. The details of each follow.

Google Transit

Google Transit <<http://www.google.com/transit>> is an existing application that offers public transit routing. According to Charles Fleming of GRTA, MARTA is in the process of conforming to the Google Transit format so that it can be published to this service. GRTA is therefore doing the same with the metro systems. If GA Tech were to also conform to Google's standard we

could publish our transit service and it would automatically be integrated into all of the other services that exist in Google's service. This would prevent us from having to maintain updated versions of datasets, other than our own here on campus. This is the clear solution to the persistence problem however the downfall is whether or not Google will publish this as a web service that other developer's can access, which would be a requirement for a customized real-time mobile transit routing application. Regardless, it is worth attempting and it would at least provide a web based client to give GA Tech users access to the routing functionality that exists.

David Williamson is an assistant director in the Parking office and is in charge of the bus routes. He has agreed to allow our data to be published to Google and has provided the latest versions in his format. Google has also responded and has provided the necessary steps to accomplish this. The current challenge is to convert his data into the format required by Google. This will be difficult as it is currently maintained with spreadsheets, including the routes which are embedded images.

The other avenue is related and should be done as a step to converting to Google Transit. That is building the network dataset, for the GA Tech routes, to be published on ArcGIS Server. Doing this allows us to have full control of the routing web service, at least for our own transit system, and puts the data in a better format to be converted for Google. The approach taken, to this end, was to first build the network by hand, with ArcMap, and then find a way to automate the transition once a better idea of what the data would look like was acquired. The first step of this is complete, for the most part, and ideas on how to automate the process are outlined below, as is the current state of the hand-built dataset.

As mentioned, the network dataset was built using ArcMap and the routing functionality was tested using the Network Analyst extension. Each segment of each route is represented as a line connected by nodes. The direction of each segment must also be represented. When ArcMap generates the network from the base shapefiles it assumes that each edge is bidirectional unless a field is added to indicate otherwise. When such a field, "oneway" is the default recognized by ArcMap, exists the direction is set to flow in the direction indicated by that field, based on the order in which line is digitized. All of this can be overridden to manually specify these parameters, however the defaults are adequate if the data is prepared accordingly. This is confirmed with the hand-built dataset. Figure 5 shows a sample of the bus stop file maintained by David Williamson. This file is, conceivably, the only file needed, from Parking, to build the routing dataset; of course this does not include the schedule.

Master Bus Stop File												
ID#	Stop Name	Route	Direction	Loc	HS	TPnt	Stop Order	GPS		WGS84		Status
								Lat 1	Long 1	Lat 2	Long 2	
1	10th St./Peachtree	Emory	To Emory				3	N33 46.886	W84 23.043	33.781433	-84.384050	Active
2	25th St./ Fowler	Blue	Counterclockwise	NW			12	N33 46.621	W84 23.645	33.777017	-84.394083	Active
3	25th St./ Fowler	Trolley	To CRC	NW			4	N33 46.621	W84 23.645	33.777017	-84.394083	Active
4	35th St./ Fowler	Red	Clockwise	SW			12	N33 46.612	W84 23.604	33.776867	-84.393400	Active
5	35th St./ Fowler	Trolley	To MARTA	SW			4	N33 46.612	W84 23.604	33.776867	-84.393400	Active

Figure 5

With the stop locations, the next thing needed is the actual line data of the route. The data provided by Parking is in image format, so another way to automate the creation of the lines is needed. Since access to the real-time bus location trace is available, this is not entirely difficult and options to go about it are discussed later. For now the line data was created as a subset of the Fulton county centerline road network manually. It is important to note that segments of the routes that follow the same road must be identical for this particular approach to work. Once the route lines have been created the next step is to snap the location of the stops to the closest point along their respective routes. Figure 6 Shows an example of the result for the blue route. The labels represent the order number column shown in Figure 5 . With the order number and the stops placed along the closest point on the route, the next step is to segment the route into lines cut by the stop locations, ensuring to digitize the line in the direction of the increasing order number. Finally a field named “oneway” should be added to the file with “FT” as the entry for each row. Once this is done, for all routes, a properly noded network dataset can be generated with the network analyst extension. Figure 7 shows an example of the verified results produced by Network Analyst in ArcMap.

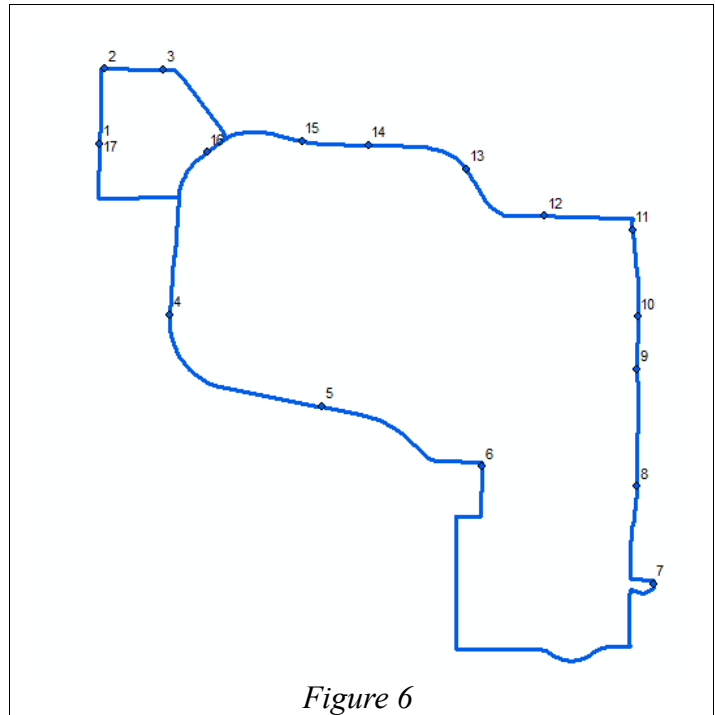


Figure 6

The question of how to automate the building of the route line data, when the routes are changed, still remains. As mentioned above this can be accomplished with the bus location history using a technique known as Map Matching (MM). MM is the process of matching samples recorded by a mobile object to lines on a given road network. A MM algorithm can be employed to periodically check the actual bus' path and verify that it still follows the same route. If a new route is detected the path can be presented to the transportation director for approval. Once approved it could be automatically integrated into the dataset.

- Advantages
 - Excellent user experience, this approach would require the transportation director to simply tell his driver the new route and wait for the change

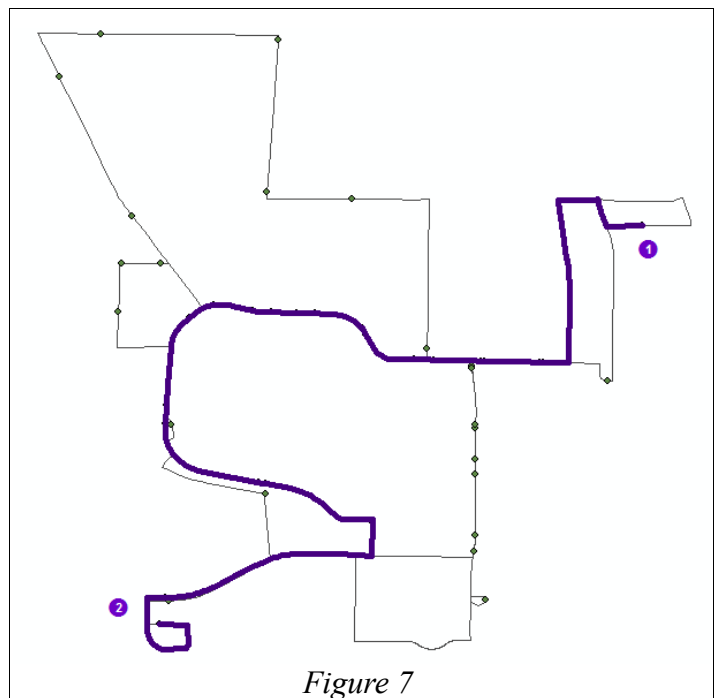


Figure 7

to be automatically reported to him.

- If put in place this approach would also prevent drivers from having to manually enter the route they are servicing, something they don't always do anyway.
-
- Disadvantages
 - This requires an accurate road map to be matched to. If a road on the route doesn't exist on the map, this algorithm would fail possibly causing frequent erroneous route change detections, or if improperly tweaked, could conversely result in route changes going unreported.
 - Fairly complex algorithm. Few people know what MM is so this might result in a persistence solution that might not persist beyond the original implementation, when maintenance is required.

Another option is to create a better route change interface for the transportation director. With a user friendly interface, the transportation director might be convinced to switch his current method. This would allow the changes to be immediately updated in the system. In turn they could be exported in the format of the director's choice.

- Advantages
 - Updates of the routes would be added the system immediately.
 - Would be easier to maintain than a more complicated approach.
- Disadvantages
 - Requires the transportation director to learn a new method of updating routes which could result in a bad user experience if the interface or export functionality are poorly implemented.
 - Is not a fully automated solution.

Future Work

Now that the current state of the routing component has been explained, it is probably apparent that much must still be done before the overall application is ready for users. The work that must and/or could be done is listed below.

- Basic
 - The routing algorithm needs a more dynamic edge weighting scheme. It currently uses length of the route segment. This causes the shortest route to sometimes require the user to leave a bus unnecessarily, ie: step1 = get on Trolley, step2 = get on Red, step3 = get back on trolley.
 - Crossing the street needs to be integrated into the network. This could either be

done with walkpaths or by adjusting the node structure. Currently the network goes far out of the way in some cases where the user could just cross the street. There is a walkpath network dataset, in a compatible format, currently being hosted by the CGIS.

- There are no temporal values in place. Basic schedules need to be incorporated so that routes aren't considered when they are not in service. In addition the green night route takes a different course which has also yet to be incorporated.
- **Advanced**
 - Beginning stops and route transitions could be calculated based on the actual position of the bus and/or the predicted time of arrival (not necessarily the same thing). For example if a user is only 500 meters farther from stop A than stop B, and she would miss the bus at stop A but not at B, she should be directed to B. Also if there are multiple itineraries from one stop to another one could be weighted by actual bus locations rather than just best average time or distance. For example, if taking the green bus to the red bus is shorter than taking the green bus to the blue but the next blue bus is in a location that would result in a faster arrival time, the green-blue route would be given.
 - Hardware is being installed, this Summer, to count people as they enter the buses. This information could be used to adjust itineraries in cases where buses might be packed. In addition, if the seating capacity is known for each bus, routes could be recommended based on whether or not the user requires, or prefers, an empty seat.

Conclusion

With the remote tracking component finished and the integrated transit routing solution underway for Atlanta, this application is well on its way to a prototype. GA Tech has a unique combination of resources needed to accomplish this. Though this application is by no means entirely novel, interesting challenges still exist. Using these resources to stay ahead of the curve on both the research and implementation fronts, is well worth the time of both City Planning and Computer Science students.