# Communication Reduction for Floating Car Data-based Traffic Information Systems

Daniel Ayala*, Jie Lin†, Ouri Wolfson*, Naphtali Rishe‡ and Masaaki Tanizaki*
*Department of Computer Science
University of Illinois at Chicago, Chicago, IL USA
Email: dayala@uic.edu, {tanizaki,wolfson}@cs.uic.edu
†Department of Civil and Materials Engineering
University of Illinois at Chicago, Chicago, IL USA
Email: janelin@uic.edu
‡School of Computing and Information Sciences
Florida International University, Miami, FL USA
Email: rishen@cis.fiu.edu

***Abstract*—In this paper, we consider a traffic information sharing system based on Floating Car Data (FCD). FCD is one of the methods used to gather traffic information. It uses vehicles as sensor nodes that transmit their speed to the server. The server then broadcasts the updated speed data to all vehicles on the road so that vehicles can compute optimal travel routes based on current traffic data. The cost of communication and load on the server are issues on this system since many vehicles on the road can generate a significant amount of communication between the vehicles and the server. A server update policy is necessary to reduce the communication load on the server and to maintain the integrity of the data that will be broadcasted to the cars. We propose an efficient way for this client/server architecture to be implemented and present an update policy that outperforms previously studied server update policies. This is shown by way of simulation using real traffic data from Chicago highways.**

***Keywords*-Floating Car Data, randomization, client/server architecture**

## I. INTRODUCTION

### A. FCD (Floating Car Data)

In order to improve the efficiency of urban transportation, Floating Car Data (FCD) techniques have been researched and developed [1]. FCD is a traffic information gathering method where cars act as mobile sensor nodes that are equipped with a location detecting device such as a GPS unit, and a communication device such as a cellular phone. These traffic-information services based on FCD are already in use in Japan and the EU [2], [3].

In traffic information systems that are based on FCD techniques, vehicles send a measured velocity to a server. This server manages a spatial database which is a representation of the road network. This database stores real-time speed information for road segments in the road network. Upon receiving measured velocities from the floating cars, the server updates the database and broadcasts the updated values to all the vehicles travelling in the road network. The vehicles that receive these updated speed values can then recalculate a minimum-time route to their destinations based on the updated real-time values received.

### B. Problem and Proposed Solution

To update the traffic information database that resides on the server, a significant amount of transmissions from the vehicles are needed. However, in metropolitan areas with millions of vehicles there are problems of the communication cost and load on the server to process all received transmissions. These problems make this traffic information sharing system based on FCD difficult to expand. One obvious solution is to reduce the number of transmissions from the floating vehicles to the server. But reducing the transmissions indiscriminately will cause the accuracy of the data that the server will provide to suffer.

Transmissions to the server can be reduced by using randomization. With randomization each vehicle would have a probability of transmitting the speed reports to the server. The effectiveness of this *transmission probability* will hinge on its construction and applicability to the server model, i.e. how the server goes about updating its link-speed database.

In this paper we present a server update policy to be used by each vehicle that reduces communication cost and increases the accuracy of the updated average speeds compared to previous server update methods.

### C. Related Work

There are different ways to build a traffic information sharing system based on FCD. First, from the point of view of system architecture, one way uses a client/server architecture, where vehicles as clients send the velocity that they measure to their server individually. This informs the server of the real time traffic condition of each road segment. Then the server broadcasts updated velocities of each road segment to all vehicles.

The other way is based on a mobile peer to peer (MP2P) architecture, where vehicles send velocities of road segments to each other without server facilities. In this paper we consider the client/server architecture.

Kerner et. al. [4] have developed a traffic information sharing system using a client/server architecture, where the server broadcasts velocities with threshold values; each vehicle sends an update to the server when its measured speed on a road segment differs from the broadcasted speed by an amount larger than the threshold. Tanizaki and Wolfson [5] introduced a randomized update policy to reduce the communication costs to the server using this threshold as well. In this randomized policy vehicles transmit the measured velocities to the server with some probability smaller than 1. In other words, to determine whether to send a measured speed to the server, each vehicle will toss a coin where the probability of getting 'head' is $p$, and then will send the measured speed only if a 'head' comes up. These techniques are not suitable for methods where the server is aggregating multiple velocities. The server will receive potentially skewed data since there's a portion of the data that will not be represented in the sample received at the server.

Goel et. al. [6] have considered and prototyped a system based on a mobile P2P architecture, where each vehicle sends measured velocities to other vehicles, updating each other using a wide range peer to peer communication method like SMS.

Shinkawa et. al. [7] have considered a traffic information sharing system based on mobile P2P extended by using buses running along fixed routes as ferries which transport the traffic information to disconnected groups of clients.

Then there are other approaches that reduce the communication cost by reducing the transmission frequency or the data volume. For example, Shinya et. al. [8] have developed a compression method of trajectories based on the decomposition of temporal and spatial component, and discrete wavelet transformation of the temporal component. Civilis et. al. [9] have developed a data reduction method based on the extraction of velocity change-points; it uses a profiled pattern of acceleration changes on a routine route. Basically, this research proposes compression methods of trajectories. Even though they deal with tracking trajectories and not traffic information, this research still does not provide a metric to indicate how much the reduction of transmission impacts the accuracy of information.

The rest of the paper is organized as follows. In section 2, we present the system specification in which we discuss the server model and the actions taken by clients to report data to the server. In section 3, an overview of past server update policies is presented and the flow-based policy is proposed. In section 4 we describe the simulation and results to compare the flow-based policy to the information cost based and deterministic policies. In section 5 we present a conclusion of our work.

## II. System Specification

We assume that the vehicles have a location detecting device such as GPS, a transmission device such as a cellular phone, and a broadcast receiver such as a radio tuner. The location detecting device is used for determining its position on one of the road segments in the map data.

The server and vehicles have the same map database that consists of road segments, identified individually by id. A road segment is defined as the section of road between two intersections or between two highway exits on a highway.

At each point in time, each road segment has a velocity $v_m$ that represents the current speed on the segment. $v_m$ is computed by simply dividing the length of the road segment by the time it took the vehicle to traverse the segment. Each vehicle has a relation storing the velocity of each road segment and so does the server. Vehicles traversing a segment compute $v_m$ when they reach the end of the segment, and inform the server so it can broadcast it to all vehicles. Clearly, this velocity is of interest to vehicles that are not on the road segment.

However, a speed value from one car on a segment is not sufficient evidence to update the speed on that segment in the server's database. Traffic speed data has variance from vehicle to vehicle so then multiple data points are needed to compute a true value of the speed conditions on the segment. The server needs to collect multiple data points for each road segment to update the database with some confidence. Then the service period will be divided into constant time intervals called *collection periods*.

During each *collection period* the server will expect to receive multiple speed reports, from vehicles on the road for each road segment. Then at the end of each collection period the server will average the received values for each segment and update the speed of each segment based on this average. Each vehicle can in turn update their databases with the new values at the server's database.

Let $k$ be the number of speed data points that the server expects to receive during a collection period for a given road segment. Then $k$ can be constructed such that the computed speed average falls between some error of the actual speed average with some confidence percentage. This sample size determination can be performed when the variance of the data is known by using a method for sample size construction that one would encounter in any standard statistics textbook.

Let $v_b$ be the computed average speed for a road segment on the server's database, then in order to update the vehicles' velocity information dynamically, the server continuously broadcasts the velocity $v_b$ of each segment.

## III. Server Update Policies

A server update policy is a method used by each vehicle to decide whether or not to send a transmission to the server.

In this section we present the deterministic, information cost, and the flow-based server update policies.

### A. Deterministic Policy

Kerner et. al. [4] have developed a traffic information sharing system using a client/server architecture. In order to prevent small differences in velocity from being transmitted to the server, a velocity threshold is used. Let $T$ be this velocity threshold, then the *transmission rule* that permits the vehicles to send $v_m$ to the server is expressed as follows:

$$|v_m - v_b| \geq T \tag{1}$$

In other words, the transmission rule indicates that the vehicle transmits $v_m$ to the server if and only if the difference between the broadcasted velocity (received from the server and stored in the vehicle database for that road segment) and the measured velocity exceeds $T$. We call the policy using a particular threshold $T$ as the deterministic policy.

### B. Information Cost Policy

Tanizaki and Wolfson [5] introduced the randomized data update policy in FCD traffic information sharing applications. The randomized policy uses a randomization function to reduce redundant transmissions. Basically the randomized policy is the same as the deterministic policy (uses threshold transmission rule), except that when the transmission rule is satisfied, instead of transmitting $v_m$ with probability 1, the randomized policy does so with some given *transmission probability* $p$. The main research question in this randomized update policy is how to determine the transmission probability $p$. Ideally this $p$ should be chosen so that the server can achieve maximum accuracy on the broadcasted data with a minimum number of transmissions.

An Information Cost Model was developed to study the trade-off relationship that exists between the Communication Cost and the Uncertainty of the data. This trade-off relationship comes from the fact that the more communication from the vehicles to the server the less the uncertainty of the data. But if reduction in communication costs is desired then a penalty in data uncertainty is accumulated. They define the total Information Cost to be the sum of the Communication Cost and the Uncertainty Cost.

As a function of system parameters like the server delay, the flow of vehicles on the road, uncertainty unit value, the transmission probability and others; they formulated the Communication Cost and Uncertainty Cost and solved for the transmission probability by minimizing the total Information Cost.

### C. Issues with Previous Policies

There are various issues with the previous server update policies. Both of these policies make use of the transmission rule with the threshold (Eq. 1). This scheme is problematic because of the inherent variance of the speed data from vehicle to vehicle. If every vehicle were measuring the same data point then this rule would work because the server would average the correct value. But with this threshold, the information that is sent to the server is incomplete. The server will not receive any data point in $[v_b - T, v_b + T]$. But in general, averaging every data point except the points in that interval will yield the wrong average speed. The computed average at the server will generally be inaccurate with these policies.

Besides sending an incomplete and potentially skewed version of data to the server, these policies are sending more data than is actually needed. The server is going to compute the average speed at the segment to update the database. Statistics can be used to construct a scheme where the average speed is computed within an error range with a specified confidence percentage. In this scheme the size of the sample that guarantees the given confidence can be computed. Let $k$ be this computed sample size, then there is no need for the server to receive more than $k$ updates if communication cost reduction is desired.

### D. Flow-based Policy

To deal with these issues the flow-based policy is proposed. This server update policy does not make use of the transmission rule with the threshold. That means that every vehicle has an equal chance of transmitting to the server, regardless of its speed. Thus the server will not run the risk of receiving incomplete information.

Also, the cars will use a transmission probability that will be constructed such that the expected number of messages sent to the server is $k$. The *transmission probability* will be $p = k/N$ where $k$ is the number of messages that the server expects to receive from the vehicles in order to guarantee a given confidence in the average speed computation and $N$ is an estimate of the flow of vehicles through the road segment during the collection period. The *flow* on a road segment is a measure of how many vehicles pass by a certain point in a certain amount of time (veh/s).

*1) Determining $k$:* $k$ is the number of messages that the server will expect to receive during each collection period. Let $X$ be a random variable representing the speed of a car in the segment during a collection period. Let $X$ have mean $\mu$ and variance $\sigma^2$. $\mu$ is what the server wishes to compute in order to update the database.

By the central limit theorem, the average speed $\overline{X}$ is normally distributed with mean $\mu$ and variance $\sigma^2/N$. Then it follows that if a $100(1-\alpha)\%$ confidence interval for the average speed $\mu$ is desired with maximum error of the estimate of $\varepsilon$, i.e. the computed mean will be in $\mu \pm \varepsilon$ with the specified confidence; then the sample size $k$ is [10]:

$$k = \frac{z^2_{\alpha/2}\sigma^2}{\varepsilon^2}, \tag{2}$$

where $\sigma^2$ is the variance of the data and $z_{\alpha/2} = \Phi^{-1}(1 - \frac{\alpha}{2})$. $\Phi^{-1}$ is the inverse of the cumulative function of the normal distribution. The real-time variance of $X$, $\sigma^2$, is unknown to each vehicle. In a practical setting $\sigma^2$ would have to be determined a priori by using historical data for a specific time of day. In our forthcoming simulations (section 4) the variance was assumed to be known and a constant for the tested service period.

*2) Determining N:* $N$ is an estimate of the flow of vehicles through the road segment during the collection period. The flow on the road is given by the formula:

$$flow = density * speed, \tag{3}$$

where the density is a measure of vehicles per meter. So to calculate the flow, a car needs to know the density in the segment and the speed. The density is not known to the vehicles traveling on the segment. The only speeds the vehicles are aware of are their own speed and the speed that the server is broadcasting, $v_b$ (average speed on the segment during the previous collection period).

Speed-flow models exist which estimate the flow of vehicles based on an average speed [11], [12]. These models are used on highway road segments. It would be incorrect for a car to use a speed-flow model with its own speed as a parameter, these models estimate flow based on an average speed. The vehicles would have to use a speed-flow model based on $v_b$, the average speed on the previous collection period, since this is the only average speed they possess. This would yield an estimate of the flow of vehicles during the previous collection period. This computed flow will be used as $N$, the estimate for the flow of vehicles during the current collection period based on the notion that drastic changes in flow are not expected from one collection period to the next.

The speed-flow model chosen for the flow-based policy is Greenshields model [12]. This model is old but widely cited [13]. The relationship is given by equation (4). In this equation $d$ is the traffic-jam density, i.e., how many vehicles fit per unit-length when traffic is at standstill. $v_b$ is the average velocity of the previous collection period. $V$ is the free-flow speed of vehicles on the given segment. $d$ and $V$ are determined a-priori and are specific to each road segment. The equation comes from the speed-flow relationship where flow is 0 at zero speed ($v_b = 0$) and at free-flow speed ($v_b = V$). Indeed the flow is 0 when traffic is at standstill, and at free flow (assuming that free flow occurs when there is a single vehicle on the road, and the speed is reduced with each additional vehicle).

$$flow \approx d * v_b(1 - v_b/V) \tag{4}$$

Using equation (4) then we can compute the flow throughout the collection period as:

$$N = flow * \tau, \tag{5}$$

where $\tau$ is the length of the collection period (in seconds).

Now, vehicles will use a transmission probability of $p = k/N$. When a vehicle reaches the end of a road segment it computes $k$ and $N$, and it transmits its speed to the server with probability $k/N$. Using this transmission probability the expected number of messages sent to the server will be close to $k$ when the estimated $N$ is close to the actual flow of vehicles during the collection period. In the following section we show how this scheme outperforms the deterministic and information cost server update policies.

## IV. Simulation and Results

In order to compare the efficiency of the Flow-based policy with the other update policies we developed a simulation system using real traffic data. In this section we describe the simulation environment and the results of the comparison.

### A. The Traffic Data for Simulation Input

We have utilized highway traffic speed data provided by the GCM travel web site [14] for evaluation of the update policies. This website is operated by the Illinois, Indiana, and Wisconsin Department of Transportations and provides a real-time map of road congestion and construction data, construction and closure reports, congestion and travel time reports, video images of the highways they gather data from, and other services.

We used a detector record that is gathered from loop sensors installed in highways. The detector records are provided by the GCM site as XML data and they contain a data set of records, each of which consists of update time of the record, sensor device unique id, velocity, number of vehicles per lane per hour, and others.

Records generated by a sensor at the end of a road segment on highway I90 in Chicago were downloaded, for the period of 6AM to 8AM on Tuesday May 22nd, 2007. The data in this period was ideal since it involved congestion and non-congestion situations and the number of vehicles that ran over the sensor was 6235. The sequence data that was collected contained records of time, velocity, and density over the road segment in five minute intervals. Using this sequence of records the following information was computed: how many vehicles went through the end of the road segment, at what time each one went through, and what was the speed of the vehicle at that time. In other words, a vehicles-sequence was generated where each vehicle in the sequence is a pair consisting of the vehicle's sensor (end-of-segment) crossing time, and its velocity at that time. The vehicle speeds were generated as normally distributed with mean of the average speed collected by the loop sensor and a constant variance of $\sigma^2 = 16$mph.

## B. *Procedure of the Simulation*

In this section we describe how one simulation run was executed. The input to each simulation run is as follows:

- the generated vehicles-sequence (see last subsection),
- a velocity threshold $T$ used in the transmission rule (for deterministic and information cost policies),
- a collection period length $\tau$ which is fixed for the duration of the simulation,
- a server update policy (Deterministic, Information Cost based or Flow-based).

The simulation is executed by two threads, the server thread, and the client thread. The server thread keeps receiving the measured velocities $v_m$ sent by the client thread and updates $v_b$ (broadcasted velocity by the server) at the end of the collection period $\tau$ by computing an average of the received speed values. The client thread then uses this updated $v_b$ to test the transmission rule for each car that passes the end of the segment (deterministic and Information cost only). $v_b$ is also used by the client thread to estimate the flow $N$.

If the server receives $m$ messages, $m < k$ then it will compute a weighted average with the average of the $m$ values having weight $m/k$. It will use current value of $v_b$ with weight $(k-m)/k$. This is done to validate cases where an extremely low number of messages is received.

Every time a vehicle passes the end of the segment, it decides whether to send a transmission to the server by using the transmission rule and by tossing a coin with probability $p$ (transmission probability). The computation of this probability differs for both randomized update policies.

Table 1 shows the value ranges for each system parameter that was tested in the simulations.

| **Parameter** | **Symbol** | **Unit** | **Range** |
|---|---|---|---|
| Threshold | $T$ | mph | [1,2,3,4,5,6,7,8] |
| Collection Period | $\tau$ | min | [2,3,4,5,6,7] |

Table I
PARAMETERS TESTED ON SIMULATION

The free-flow speed $V$ of the segment was estimated as 77.845 mph, and the traffic-jam density $d$ of the segment as 0.1085 vehicles/m. $d$ and $V$ were determined a-priori by the least square method using equation (4) and data downloaded from the GCM website [14]. These values are used in the Greenshields model for estimating the flow.

The server was setup to expect a 95% confidence ($\alpha = 0.05$) that the maximum estimation error would be $\varepsilon = 2mph$. These values are used for computing $k$. This setup yields $k = 15.366$ using equation (2) with $\sigma = 4$. The value for $\varepsilon$ was determined through simulation. In the next section we describe this procedure, the performance metrics and results of the simulations.

## C. *Comparison of the Server Update Policies*

*1) Performance metrics:* We compare the two randomized policies and the deterministic policy by using two values that are measured during the two-hour execution of the simulations. One is the *communication cost*, namely, the number of data transmissions from the cars to the server that occurred during the simulation run. The other is the *average error*.

The *error* is calculated as the absolute value of the difference between the computed average speed from the received sample at the server and the actual average speed during the collection period. To compute the *average error* for a simulation run, for each collection period of the simulation run, the error of the sampled average speed is computed. Then we take the sum of all these error values and divide it by the total number of collection periods in the simulation run. Then we have the *average error* measured in [mph].

This *average error* computed in the simulations is not the same as the $\varepsilon$ used for determining $k$. $\varepsilon$ is used to establish an error range for the average speed on one collection period to have with a specified confidence. The average error is the actual computed average error for each collection period during a simulation run. Still, it is true that as $\varepsilon$ increases, so will the average error as well. Figure 1 shows this relationship since the average error increased as $\varepsilon$ increased. However, simulations showed that they don't increase at the same rate. Figure 1 will be further explained and formalized in section 4.3.2.

We compare the policies by combining the *average error* and *communication cost* into an efficiency metric. The number of messages spent (communication cost) by the cars in a simulation run by the cars can be considered the input to the system. The output of the system would be the average error that was attained by using the given communication cost. But the goal of this system is to minimize this average error. In other words we could say that we want to maximize $1/averageError$ and for the efficiency metric this will be the output of the system. Then,

$$\begin{aligned} efficiency &= output/input \\ &= (1/averageError)/communicationCost \\ &= 1/(averageError * communicationCost) \end{aligned} \tag{6}$$

Notice that for instances in which the $averageError$ is approximately equal, this metric will favor a policy which has less $communicationCost$. Likewise, if the communication costs are equal, the metric will favor a policy which has less average error.

*2) Maximum Estimation Error ($\varepsilon$):* Simulations were ran to determine the value for $\varepsilon$ that should be used. This value will affect the computation of $k$ and will in turn affect the

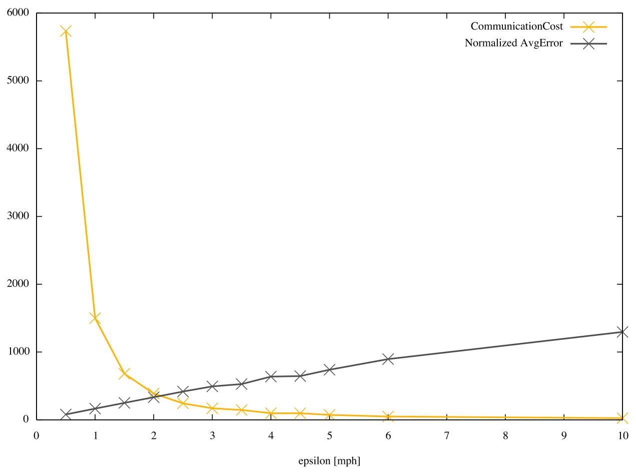

Figure 1. $\varepsilon$ equilibrium ($\tau = 300s$)

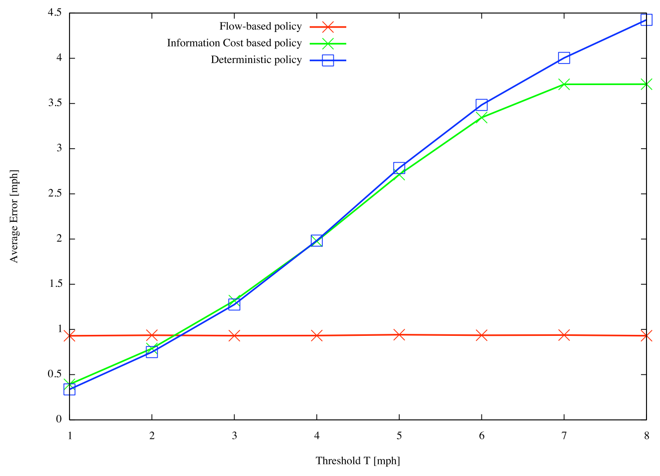

Figure 2. $T$ and Average Error ($\tau = 300s$)

performance of each method. As $k$ increases so will then increase the number of messages sent to the server. The aim of our method is to reduce the communication cost so then $k$ should not be too big that the communication is too large, but also it should not be so small that the communication is reduced but the accuracy of the average speed computed at the server is compromised.

Simulation runs as described in section 4.2 were executed for different values of $\varepsilon$. The *average error* and *communication cost* were computed for each run (using flow-based policy). These two metrics are measured in different scales. The communication cost, the number of transmissions, can take values between 0 and 6235 (total number of cars). The average error can take values between 0 and 17mph. The maximum average error (17mph) was determined by running a simulation in which no messages are sent to the server and thus there is no update of the database. Then the average error values were normalized to the [0,6235] scale so that both metrics could be visualized on the same scale.

Figure 1 shows a plot of the communication cost and normalized average error with varying $\varepsilon$ values. There exists a trade-off relationship between the communication cost and the average error as it pertains to $\varepsilon$. If $\varepsilon$ is too big then the communication cost is low but the average error will be much higher. If $\varepsilon$ is too small then the average error is low which is good as well but the communication cost is too high. Then an equilibrium $\varepsilon$ can be chosen that balances the average error and communication cost. Similar to the supply-demand model in economic theory, the equilibrium occurs at the point where the two curves intersect. In this case the equilibrium $\varepsilon \approx 2mph$.

*3) Results evaluation:* Simulations were also run to compare the server update policies using the parameters specified on Table I.

In Figure 2 we can see the comparison of average errors obtained by the different policies when the collection period was 5 minutes and for different threshold values. The flow-based policy has a constant average error because it makes no use of the threshold. In this case the flow-based policy obtained a better average error than the other policies when the threshold value used was greater than 2 mph.

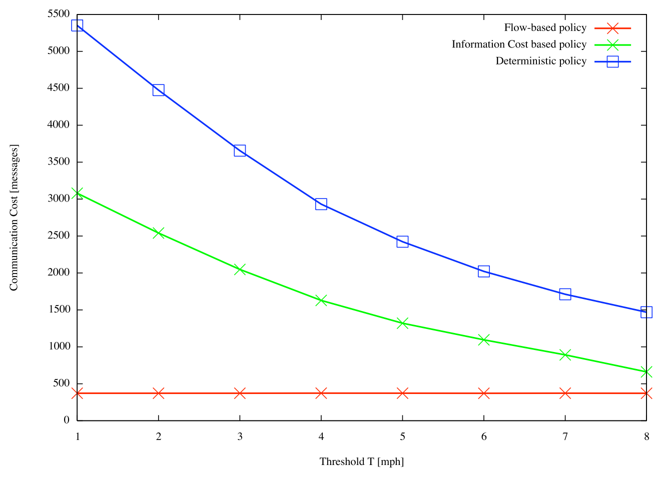

Figure 3. $T$ and Comm. Cost ($\tau = 300s$)

Nevertheless, the flow-based policy obtained these average errors at the servers even though the number of messages sent were much less. In Figure 3 we can see that for this same case of the 5 minute collection period there was less communication cost for any threshold used by the other two policies.

Figure 4 shows how the flow-based update policy outperforms the other policies in terms of the efficiency metric for any of the instances shown. This pattern was observed for all of the simulations that were run. In some cases were the threshold is very low the other policies obtained

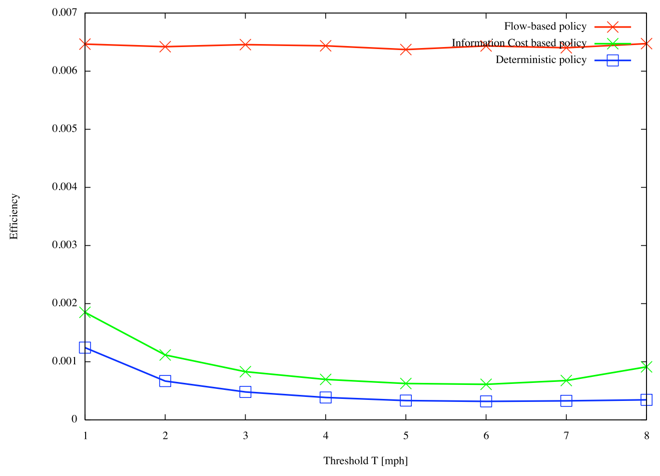

Figure 4. $T$ and Efficiency ($\tau = 300s$)

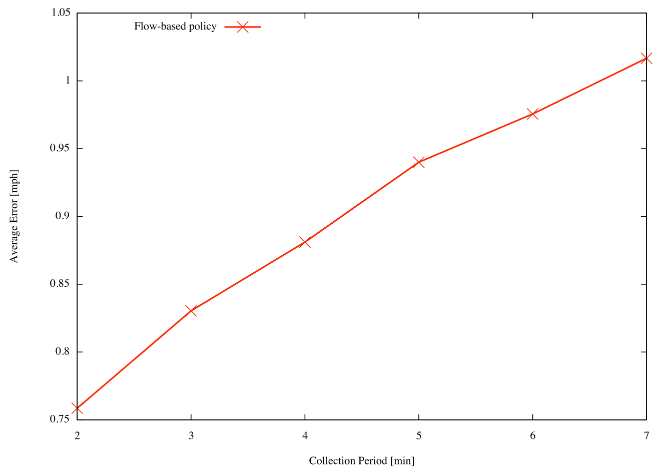

Figure 5. Collection Period Length and Average Error

a better average error but at the expense of having a high communication cost.

Even though many more messages are sent by the other policies the flow-based policy obtains better results because with it the vehicles that send their speeds are a complete representation of the whole population. In the other policies because of their use of the threshold there is an interval of data that is not represented at the server when computing the average speed. It's clear that the flow-based policy is more efficient with the smaller amount of messages that it's sending to the server.

Figures 5 shows how the average error changes with different lengths for the collection period. The trend is that as the collection period length increases so does the average error. The main reason for this phenomenon is that the longer the collection period is then the bigger the change in flow will be from one collection period to the next. This makes it so that the smaller the collection period is then the better the estimation of the flow for the current collection period will be based on the flow estimation of the previous collection period. Also, when the collection periods are shorter, more data is sent to the server since the same expected number of messages should be sent to the server during each period ($k$) but there are more collection periods during the two-hour simulation run.

Using the flow from the previous collection period as an estimation for the what the flow will be in the current collection period turned out to give excellent results. In our simulation we set the server to desire a confidence interval of 95% ($\alpha = 0.05$) and with a maximum estimate error of $\varepsilon = 2mph$ which yielded $k = 15.366$ with $\sigma = 4$. For all simulation cases, the average number of messages sent during each collection period was between 15 and 16 messages. So then the number of messages the server was expecting ($k$) on average was achieved with the flow-based server update policy.

## V. Conclusions

In this paper, we addressed the problem of reducing the communication cost in traffic information sharing systems that are based on Floating Car Data. In these systems cars travelling on a road segment measure their speeds on each road segment and send updates of current traffic conditions to a server. The server then broadcasts the updated speed data to all cars on the road. The vehicles in turn use this data to compute optimal travel routes.

We described a server model that computes average speeds on road segments during pre-defined collection periods. To address the problem of reducing the communication cost to the server while maintaining data integrity at the databases we proposed the flow-based server update policy. This update policy uses a transmission probability that enables the server to receive the amount of messages that it desires from each segment depending on a confidence percentage and maximum allowable error for the average speeds that will be computed.

Simulations and results were presented that showed how the flow-based update policy outperformed the deterministic and information cost based policies in various respects. The simulations were performed with data from Chicago highways. An efficiency metric was presented to evaluate both policies. In terms of system efficiency, the flow-based policy exhibited superior results in all the simulations that were run. The average error was less for the flow-based policies in most cases. Also, the flow-based policy accrued less communication cost for all of the simulations that were run.

Future work includes extensions in several directions. There exist other more modern speed-flow models that could work well and were not tested in this work (e.g., [11]).

The proposed method is designed to work for highway road segments because of the use of the highway speed-flow model. Further work is required to propose a different approach for arterial or minor roads. We're also interested in server side reductions of transmissions in this system. For example, the server could possibly use server-side predictions of the average speed to further reduce $k$ and thus reduce the communication cost further. It would also be interesting to adapt our update policy to a P2P architecture like the one proposed in [6].

## References

[1] S. Turksma, "The various uses of floating car data," in *Tenth Intl. Conf. on Road Transport and Information Control (Conf. Publ. No. 472)*, April 2000, pp. 51–55.

[2] "http://www.premium-club.jp/technology/index.html," Last visited Dec. 2009.

[3] "http://www.vmzberlin.de/vmz/," Last visited Dec. 2009.

[4] B. Kerner, C. Demir, R. Herrtwich, S. Klenov, H. Rehborn, M. Aleksic, and A. Haug, "Traffic state detection with floating car data in road networks," in *IEEE Proc. on Itelligent Transportation Systems*, 2005, pp. 44–49.

[5] M. Tanizaki and O. Wolfson, "Randomization in traffic information sharing systems," in *GIS '07: Proc. of the 15th annual ACM Intl. Symp. on Advances in Geographic Information Systems*, 2007.

[6] S. Goel, T. Imielinski, K. Ozbay, and B. Nath, "Grassroots: A scalable and robust information architecture," Dept. of Computer Science, Rutgers University, Tech. Rep. DCS-TR-523, June 2003.

[7] T. Shinkawa, T. Terauchi, T. Kitani, N. Shibata, K. Yasumoto, M. Ito, and T. Higashino, "A technique for information sharing using inter-vehicle communication with message ferrying," in *MDM '06: Proceedings of the 7th International Conference on Mobile Data Management*, 2006.

[8] A. Shinya, N. Satoshi, and T. Teruyuki, "Research of compression method for probe data-a lossy compression algorithm for probe data," *IEIC Technical Report (Institute of Electronics, Information and Communication Engineers)*, vol. 104, no. 762, pp. 13–18, 2005.

[9] A. Civilis, C. S. Jensen, and S. Pakalnis, "Efficient tracking of moving objects with precision guarantees," in *In Proc. MobiQuitous*, 2004, pp. 164–173.

[10] R. V. Hogg and E. A. Tanis, *Probability and Statistical Inference*. Prentice Hall, 2001.

[11] *Highway Capacity Manual 2000*. Transportation Research Board, 2000.

[12] B. D. Greenshields, "A study of traffic capacity," *Highway Research Board Proc.*, vol. 14, pp. 448–477, 1935.

[13] H. Lieu, "Traffic flow theory - a state-of-the-art report: Revised monograph on traffic flow theory," Turner Fairbank Highway Research Center (TFHRC) at http://www.tfhrc.gov/its/tft/tft.htm, 2002.

[14] "http://www.gcmtravel.com/," Last visited Dec. 2009.