Finding Traffic Information With Mobile P2P Networks

Distributed Data Mining Project
Sandor Dornbush
January 4, 2006

1. Introduction

This paper details a simulation of automobile traffic monitoring using mobile peer-to-peer networks. There is a great market need to be able to monitor traffic in real time. There have been several centralized approaches to this, but it is cost prohibitive to implement these systems in a large scale.

The goal of the project is to create a system that could be implemented with currently available technology. Specifically the system would use a Global Positioning System (GPS) to determine current location and speed. Speed can be calculated from two consecutive GPS readings. The system would use a wireless networking communication medium such as 801.11 a, b or g. Many cars already come equipped with GPS systems and 802.11 hardware is quite cheap at this point. I believe that a commercial version of this system could be sold for around $200. A price many people would pay to avoid traffic.

As cars travel through the system the traffic device would track the speed of the car at every location. The speed of each car can be calculated by looking at successive GPS way points. GPS systems for monitoring automobile speed have already been deployed in a variety of situations[1]. This information could be used to build a speed map of the area. As cars come close to each other they would exchange their speed map with each other. Through these interactions each peer in the system will be able to build a map of expected speed on every route, even those they have not visited.

I propose to study this problem in a simulation. In the simulation cars will drive random paths through a grid. I will use a simple square grid. The grid will have a predetermined speed map. The problem will be how accurate will the estimate of the speed map be to the actual speed map.

This paper introduces the MetaKMeans algorithm. This algorithm computes K-Means clusters using only local messages. The system uses epidemic communication. Every time two cars interact with each other they try to share traffic information. A structure similar to Lamport Vector Clocks is used to ensure that cars have the most recent data possible.

The system is tested in a Java simulation. A set of K random Gaussian traffic congestion areas are created. As the cars traverse map they learn the traffic of the road they are traveling over.

Section two of this paper details related work and background information. Section three contains the a description of the MetaKMeans algorithm. Section four describes the simulation used to test the algorithm.

Section five contains the results of experiments conducted. Section six contains an analytical review of the algorithm. Section seven contains the conclusion and discussion of future work.

2. Background

2.1. Clustering

Clustering is an important data mining tool. Clustering is the act of finding data points that are similar to each other by some measure. Clustering is used widely for a variety of purposes from grouping gene sequences with similar function to grouping related news stories. [2][3]

This data monitoring discovers clusters of slow speed. This would provide the end user with an easy way to see traffic issues in the system. In this experiment we wish to find how accurately the speed map matches the actual map; however the typical use case for a traffic monitoring system is not a report of the entire map. The typical user only wants to know about clusters of slow speed. Therefore in this experiment the cars will only express information about areas of slow speed. If the speed is not transmitted it is assumed to be close to normal. It is assumed that in the deployed system the GPS device would have a map of the roads along with the posted speed limits. Most GPS driving aids have this information today.

K-Means is a well established method for finding a user specified set of K clusters identified by K centroids. There are a few deficiencies in the K-Means algorithm. It is a greedy search so it can get stuck in local optimums. It is very sensitive to initial conditions. It does not deal well with empty centroids. For each one of these problems there are simple fixes. For an efficient traffic monitoring scheme each one of these must be addressed. The solutions and motivations will be presented in section 3.4.

2.2. Related Work

There are several algorithms that have similar components as this algorithm. It was not the goal of this project to create a new algorithm. However as I will discuss none of the algorithms could work in the manner required for automobile traffic monitoring on a mobile ad hoc network. This paper synthesizes two different types of algorithms to produce a novel and empirically efficient clustering algorithm.

2.2.1. Distributed Clustering

Several algorithms have been presented to find clusters in a distributed fashion. For our purposes we only wish to consider the homogenous situation. We will assume that every car collects similar data types. There several papers presented ways to perform a bit of data analysis at each node then send the results of that analysis to a central node. A few examples of this esemble based approach can be found at [4][5] The MetaClusters algorithm applies this sort of technique however each node tries to do the analysis of the collected statistics.

2.2.2. Endemic Communication

This paper deploys a form of communication that is alternately referred to as rumor [6], gossip [7] or endemic based. This class of communication protocols is based on the practice of saving messages from previous nodes and relaying those messages to other nodes. While these communication protocols do not provide

any guarantee on the timeliness of the data, they are often the best possible in disconnected settings.

It is possible to prove the accuracy of data mining tasks performed on these loosely connected networks. [3][8] However all of the proofs so far have relied upon conservation of mass. For example in a means calculating scheme so long as no value is lost any two nodes can average their values and the network will converge towards the correct value. However in our experiments there is no conservation of mass. When the nodes start out they have no knowledge of traffic, they discover the traffic as they travel through the map. So it is not feasible to distribute the mass throughout the network.

2.2.3. Lamport Clocks

In distributed systems it is impossible to maintain a global clock. For this reason several schemes have been established for ordering events in distributed systems devoid of a global clock. The best such scheme is Lamport's Vector Clock [9]. In this system nodes keep a record of what time they was last reported for every other node. Let $t_{i,j}$ represent the time recorded at node i for node j. When ever two nodes i and j communicate they will both update their records by choosing the new time to be $\max(t_{i,k}, t_{j,k})$ for all k. The original purpose of Lamport's Vector Clock is to be able to order events in a distributed system. This paper demonstrates a technique that is similar to this but associates information with each of the timestamps.

2.2.4. Automotive Peer-To-Peer

Several groups have investigated creating peer to peer networks with automobiles. [10] The connection is very natural. Already cars have significant processing power, communication and storage. Most of the resources are dedicated devices, such as mobile phones or engine controls. However it is quite feasible in the near future to create WiFi networks over cars. Most of the work has focused on providing WiFi services such as music over these networks. [11]

3. Algorithm

This paper presents a distributed K-Means clustering algorithm that does not require constant connectivity. The goal of this algorithm is to find k-means on a mobile ad hoc network were disconnected components are more common than connected components. There are many obstacles to creating a fully connected traffic monitoring system. Cars can move very fast, which limits connection time. Cars moving in opposite directions have a very short window to exchange information. Cars can operate in sparse settings, where networks would be disconnected. One of the motivations for this project is to build a system that would work outside of cellar coverage. Initially any commercial system would take a bit of time to deploy. So in many situations there may be a large number of cars but only a few who are participating in the P2P traffic monitoring. For all of these reasons we cannot rely on a fully connected network.

There are several techniques for calculating K-Means in a P2P fashion. However all of them have relied on a fully connected network. In the algorithm P2P-K-Means [12] every node must interact with other nodes every iteration. L2 Norm based K-Means [13] improves on this some by only asking each node to talk to its immediate

neighbors on most iterations. However if there is an alert; the algorithm requires a global converge cast.

There are several algorithms for distributed clustering that rely on centralized ensembles. These algorithms calculate statistics at each node then send those statistics to a central site. Then the central site calculates the global clusters from the statistics from the other nodes. The major contribution of this paper is to take this ensemble approach to a peer-to-peer environment. Unfortunately I read these papers very late in the project and could not incorporate their ideas into this project. There are some items from these papers that I would like to incorporate into this project.

The motivation of the algorithm is for each node to keep a compressed version of every other nodes traffic information. This information is expressed as the k cluster locations and member ship counts. Then each node will calculate a set of k MetaClusters based on the clusters locations and counts. Since each node cannot know each other nodes up to date traffic information each node keeps a record of the most recent version of that nodes information.

3.1. Notation

3.1.1. General Notation

For notation n is the total number of nodes in the system. k is the number of clusters calculated. id is the identification number for the node in question. φ_i is the i^{th} cluster of the current node. In the traffic monitoring application of Meta K-Means each φ_i is a point in 2D space. ω_i is number of data points that contributed to φ_i, for the i^{th} cluster of the current node.

3.1.2. Exchange Notation

Each node will maintain an array of centroid information about other nodes. The array $\alpha_{i,j}$ is the j^{th} cluster of the i^{th} node as recorded by the current node. The array $\beta_{i,j}$ is the number of data points in the j^{th} cluster of the i^{th} node as recorded by the current node. μ_i is the time that node i calculated the clusters as recorded in $\alpha_{i,j}$ at the current node. μ_i is the timestamp of the data recorded at the current node, that originated from i. In this implementation a global clock is used. However there is no need for a global clock as μ values are only compared against values from the same node. So long as each node uses a monotonically increasing value that will suffice.

3.1.3. MetaCluster Notation

Each node will calculate MetaClusters from the centroids gathered from other nodes. Φ_i is the i^{th} MetaCluster at the current node. Ω_i is number of data points that contributed to Φ_i, for the i^{th} cluster.

3.2. Algorithm

Every iteration each node will sample the traffic map and calculate the local K-Means centroids. The local K-Means is only executed if the sample of the traffic map has changed. If the sample is a repeat of a previous sample then the K-Means will not change. If the K-Means changed then the exchange data for this particular node must be updated. Each node executes the following code in every iteration:

```
calculateLocalKMeans() {
1. Sample data.
2. If the sample contributed new information calculate new ω and φ otherwise return.
3. Update α_{i,j} and β_{i,j} for this node.
4. Set μ_i = now().
}
```

During each iteration the nodes attempt to send their exchange information to other nodes. Each node sends their view of the world to other nodes.

Upon receiving a message each node checks to see if the data that is sent to it is newer than the information already stored at that node. If the message contains new information that information is stored in the local α, β and μ variables. This is an endemic form of communication.

```
recieveMessage()
{
for i=0 to n
do
    if μi from the message > μi
    then
      Copy αi,j, βi,j for all j.
      Update μi.
    end
end
}
```

After sampling the data, and exchanging messages each node calculates new MetaClusters. Each node merges the snap shot of other nodes centroid data with its local centroids. The collection of α are assigned to the closest Φ. The Φ_i are calculated based on a average of the $\alpha_{i,j}$ weighted by the $\beta_{i,j}$ values of all the data points that are assigned to Φ_i.

```
calculateMetaClusters() {
repeat
  for each item in α
  do
    assign that centroid to the closest Φ
  end
  for each Φi
  do
    let ζ be the set of items from α
    assigned to Φi.
    Ωi= Σβi for all i such that αi is in ζ
    Φi=(Σαi for all items in ζ)/ Ωi
  end
until Φ does not change.
}
```

3.3. Data Modeling

This project modeled traffic information as a 3rd dimension on top of a 2 dimensional map. The 3rd dimension expressed the speed at that particular point. The system can either load the traffic information from an image file, or generate random Gaussian traffic disturbances. There are some problems with this representation. In many situations the traffic on one side of a highway has little effect on the traffic on the other side of the divider. It would be interesting to model all with different maps for different directions, one for each direction of traffic. However since road run in an infinite number of different directions this is impractical. The simple 3 dimensional model allows for simple data modeling and error monitoring.

3.4. Optimization Techniques

There are several problems with the standard K-Means algorithm. To achieve acceptable results it was necessary to implement optimizations that avoid those problems. In this traffic monitoring simulation each node starts with no knowledge of the traffic, then learns as it goes along. As such there are two main problems. First all centroids start empty so those must be handled. Also the K-Means will get stuck in local minima that are discovered early instead of global minima.

To address the first problem after executing the local K-Means each node

checks for empty centroids. If there are empty centroids the algorithm will move one empty centroid to the data element farthest from the established centroids.

To address the second problem periodically the nodes will generate several new sets of random centroids, then choose the best set of centroids. The best set of centroids is the set with the lowest mean squared error. In the charts of the error is possible to radical drops in error when all of the nodes recalculate the centroids.

4. Test Scenario

Two major sets of tests were conducted. The first set of tests is built to verify the MetaCluster algorithm. These tests are devoid of cars, instead use abstract sampling nodes to eliminate sampling bias. The second set of test is built to simulate cars traveling along roads. The two test have fairly different parameters and different results. The results section will discuss this further.

To minimize sampling error and prove the effectiveness of the MetaCluster algorithm an abstract implementation was built. In this test scenario in every iteration each node randomly samples a point on a road. In that way the samples are independent and random. Each node randomly selects one node to communicate with. In this way the communication is independent and random. This test has much better performance than the car simulation system.

This project seeks to study static traffic information. To provide a good solution for stationary traffic in a P2P manner is a novel work. It would be much more difficult to study non-stationary traffic and would be difficult to prove any results about such a system. There are many good free traffic simulators[2]. These traffic simulators typically model how traffic patterns change with congestion. This experiment instead imposes an arbitrary speed map and forces cars follow that map.

I created a simulation that creates a map of roads then imposes an arbitrary speed map on that map. The simulation is written as a Java application. The speed map can be any image file or randomly generated Gaussian. For an image the speed is derived from the hue of each pixel. The simulation has many faults. One fault is that each side of the road is considered to be 1 km wide. With a few tweaks I should be able to fix this. Another problem is that cars can run over each other. In the simulation cars will turn on a dime at full speed. I don't think it is very important to this project to model turning radius, etc. Another oddity of the simulation is that cars slow down when there is no other car around, but the speed map dictates a slower speed. I think that is actually accurate to the problem. In a real environment not all cars would run this system. Therefore cars could slow down without any cars ahead of them reporting a slow down. If none of the cars in the congestion use the system, there would be no message of the congestion. In this way cars that don't use the system are invisible, but there presence is felt.

The image below is from a run of the simulator with 40 cars. The cars are represented by the green dots. The roads are colored according to the direction of travel and the speed at that point. North bound and east bound roads are colored reddish. South bound and west bound roads are bluish. Intersections are a orangish color. The saturation and brightness are controlled by the speed at that point on the map. The following

map was generated from 5 random Gaussian.

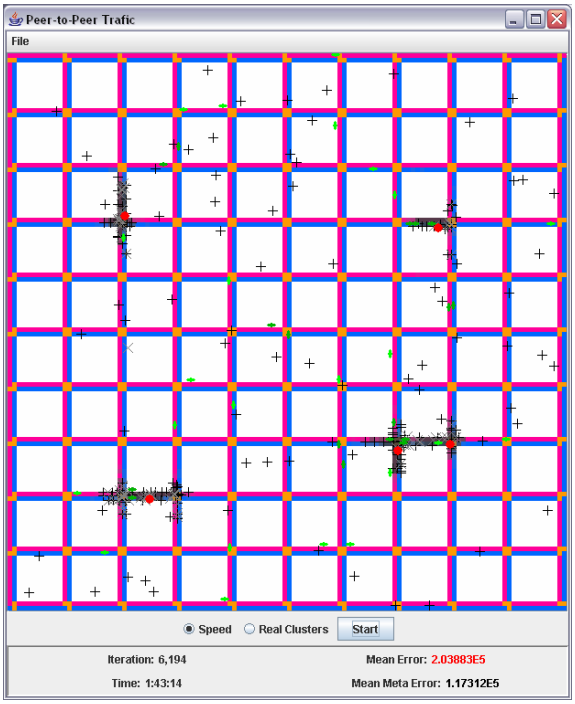

Figure 1 A screen capture of the simulator.

Another way to view the same traffic map is to view the clusters of traffic. These clusters are calculated by the simulator given full knowledge about the traffic map. Each cluster is shaded differently.

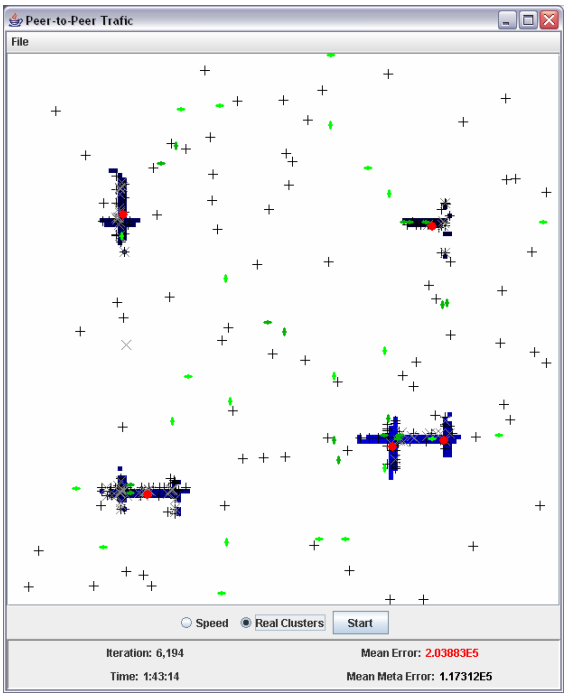

Figure 2 The cluster view of the speed map.

	A single East/West bound car.
	A single North/South bound car that is communicating.
+	A local cluster of one node.
×	A meta cluster of one node.
	The center of a globally calculated cluster.

Figure 3 Simulation icon legend.

This simulation used a simple model of wireless communication. Due to time and cost constraints I was not able to incorporate a more sophisticated model of wireless communication such as OptNet or NS-2. In the model a node can send a message to another node if no other node has already sent a message that would have collided and the nodes are within a distribution radius of each other. It is assumed that there is no hand shaking and that the complete state information can be sent in one second. A justification for the later assumption is detailed in 6.1. The simulation does not simulation collisions or exponential random back off. If more than one node

could communicate during one time slice one of the nodes is chosen at random to succeed. The successful node will not try to send a message in the following iteration. This provides a close approximation to CSMA/CA.[14]

5. Experimental Results

This paper evaluates the performance of the system under a variety of circumstances. The experiments determined the effect the following parameters have on the accuracy of the estimated speed map. Each experiment was repeated either 3 or 4 times to minimize the effects of random decisions in the process.

In the diagrams that follow the error is denoted in meters. The error the difference between the average squared error between the centroids found by the cars and the centroids calculated with perfect knowledge of the traffic map. Each iteration is equivalent to one second. Most of the experiments ran for 10,000 seconds or 2 hours 46 minutes, less than the span of a common rush hour.

5.1. Random

A set of experiments were performed to establish that the MetaCluster algorithm works well with ideal conditions. In this experiment every iteration the nodes sampled the traffic with independent and random samples with replacement. In the car simulation the sampling is not random. The cars randomly choose destinations and take a logical path to the destination. As such the points in the middle of the map will be sampled much more often than those at the edges. The nodes in this set of experiments randomly choose a node to communicate with in each iteration. There are several reasons that this situation could not be created in a real peer-to-peer network. However it does show the fast convergence of the algorithm in an ideal setting.

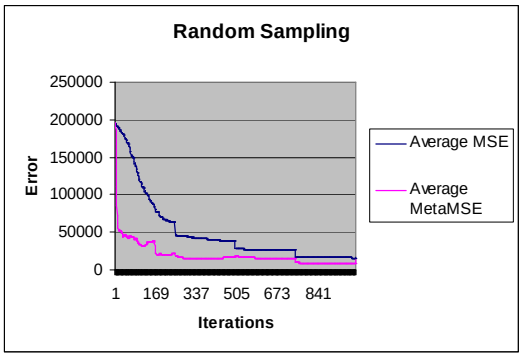

Figure 4 Random Sampling

5.2. Density of Cars

The simulations all use a fixed size traffic map. Each simulation generates a new set of k Gaussian clusters. A series of experiments have been conducted to determine the effect of the density of cars. Three different sets of experiments were conducted with numbers of cars, 20, 40 and 80. The obvious conclusion that as more cars participate in the network the error at each node decreases.

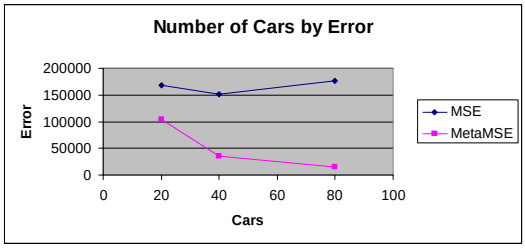

Figure 5 Number of Cars Vs Error

5.3. Transmission Radius

Several tests were performed with different transmission radii. The goal is to determine what size is needed to ensure that a high quality map can be built. The transmission radius will affect the number of interactions that a car can have. For two vehicles to communicate

they must remain in within the transmission distance of each other long enough for both sides to send the relevant messages.

For two cars traveling in the same direction the distance need not be very large. They will have a small relative speed, and hence a large window of time to transmit information. For these types of interactions the distance may only need to be a couple of times the size of a car. Let s_1 and s_2 be the speed for the two cars. Let be t the time needed for two cars to exchange a set of messages. Let r be the transmission radius of the two cars. Then we know that:

$$t \leq \frac{r}{|s_1 - s_2|}$$

If two cars traveling in opposite directions wish to communicate they only have a small window of transmission. The two cars will have a large relative speed and small window of time to transit their information. However that information is often the most important information. Cars traveling in the same direction will have information about the same portion of the map. Cars traveling from either direction can contribute information about opposite sides of the map. More importantly drivers typically don't care what the traffic is like behind them; they want to know what is ahead of them. For this scenario know that:

$$t \leq \frac{r}{(s_1 + s_2)}$$

The typical Bluetooth transmission distance is 10 to 42 meters. For this to be effective the cars would need to perform all communication, including discovery in just 0.18 seconds. This is not feasible, handshaking can not be performed on that time scale. In an outdoor setting commercial class 802.11b can have transmission distances as great as 500 meters.[15] At such distance the transmission rate falls. Cars would have 9 seconds to communicate at that distance. This application may need a better transmission medium because it performs rather poorly at these small distances.

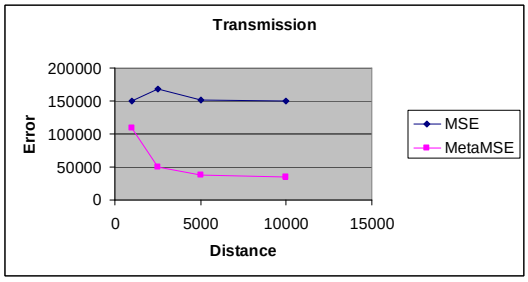

Figure 6 Transmission Radius Vs. Error

5.4. Memory

The cars in the simulation can run in two modes. In most experiments the cars behaved as a taxi or bus would; they never stop driving. In these experiments the cars don't forget the traffic information that they have learned so far. Obviously this helps the cars learn the traffic map more quickly. A more realistic model would be that once a driver reaches their destination they turn off their car. Tests were performed to simulate this behavior. In these tests once a car reaches its destination it erases its memory, generates new centroids, and chooses a new id. I new id is needed so that the memory of the old car can still echo around the network. While these tests did not perform as well as the taxi model tests they still performed decently.

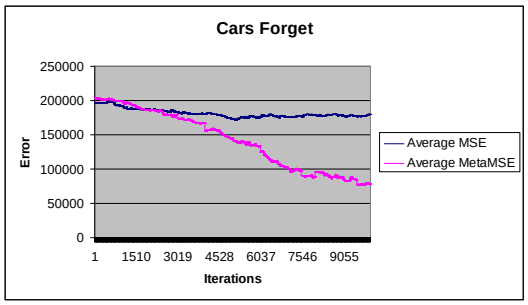

Figure 7 Cars that Forget

5.5. History Accuracy

When the cars travel through the map they must decide at what accuracy to record the traffic information at. The true traffic map has a resolution of 1km. The cars sample this at a greater resolution; once every second. The system must decide what data items are identical and which constitute new information. To determine that the cars find the traffic record that has the smallest Euclidian distance to the new data point in question. If that distance is less than some threshold then the data point is considered a repeat and discarded. In the following experiments we determined that the optimal value for that distance is actually 1k, the same as the map resolution.

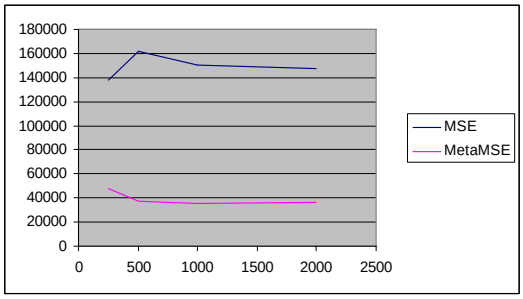

Figure 8 Accuracy as a function of history accuracy.

6. Algorithm Analysis

There are several key aspects to this algorithm that I was not able to quantify. One item that would be very useful to understand would be, what is the probability that a given node i has the cluster information from a node j. If so can you state how old that information would be on average. There are two parts to this. The first part is simple. Has node i talked to j? If this is represented as t(i,j). That value is simply $1-(1/n)^c$. Where c is the number of communication rounds and n is the total number of nodes. The more difficult question is what did those nodes tell each other. So the more difficult question is has node j talked to node k before it talked to i. That gets to the difficult question. In order to analytically bound this algorithm one needs an expression for the probability that a node i has information about node j either from direct communication or from gossiping. The other expression needed is the expected age of that information. I was not able to quantify those values, and hence could not analyze the system.

In [??]

6.1. Communication Cost

Ideally one would like a local algorithm to scale infinitely with the number of nodes. This algorithm does not currently support this. Instead the size of each message is O(nk) where n is the number of nodes. Message size can be broken down by the constitute parts. In this example I assume that the centroids are represented by double precision floating point number. The centroid counts and updates are represented by single precision integers and a double precision integer respectively. For this example the number of nodes is 100 and the k is set to 10.

Size of Memberships	n*k*4	4000
Size of Centroids	n*k*2*8	16000
Size of Updates	n*8	800
Total Size Bytes		20800
KB		20.3125

In this simple example the messages is only 20k. These messages are not tiny, however they are very reasonable size for transmission for 802.11.

On way to limit the size of the messages is to limit the number of messages passed to some constant c. The key to an effective implementation of this is choosing the most important c centroids and membership counts to send. Ideally one would like to send only the centroids that would contribute the most to the solution. Some of the items to be looked at would be the age of the information, the quality of the information, and the quantity of that information. One would want to send the most recent centroids. The centroids with the lowest mean squared error would be sent. The centroids with the highest membership counts should be sent.

Another way to approach this problem is to combine the clusters then send a summary. However if no formal tree is imposed there is a danger of over reporting some items. If two nodes, i and j combine their clusters which both include a cluster k. If another node l receives the summaries from i and j, it will receive a report of k's clusters twice.

7. Additional Research

Another interesting problem would be to study the effect of changing traffic patterns. Traffic patterns are very fluid and a good system must take this into account as well. I may pursue this if I succeed with all of the previous studies. To model changing patterns the algorithm would have slowly forget old information. It is not clear to me how to cause old data to fade out of the system.

8. Conclusion

This paper introduced a new approach to finding clusters in a disconnected peer-to-peer network. The algorithm has been empirically shown to converge to the correct answer in ideal settings. The algorithm was also shown to be capable of converging on a descent answer even in the constrained setting of an automobile network. The algorithm performs well even if the sampling and communication are not random. The algorithm can even be extended to work on non-stationary data.

This paper focused primarily on how to perform well with disconnected graphs. Further work should try to create a system that works well in a disconnected setting but also takes advantages of connected graphs. If a traffic system such as this was widely deployed it is conceivable that in a traffic jam a strongly connected graph would form. An ideal algorithm would take advantage of this situation. In these situations a hieratical approach could provide significant advantages over the flat topology used in this paper.

The goal of this paper was to show that it is possible to build a peer-to-peer traffic monitoring system with standard components. These experiments showed that there is an algorithm that would work for this problem, but that standard 802.11 networking may not be sufficient. There are many other networking standards available and more are being developed. In the very near

future we will be able to build systems similar to this very cheaply.

9. Bibliography

[1] M. Dello, Rent-a-Car Motto: Speed Bills *Wired Magazine*, July 2001

[2] P. Tan, M. Steinbach, and V. Kumar *Introduction to DataMining* Pearson Addision Wesley 2005

[3] news.google.com

[4] E. Januzaj, H.-P. Kriegel, and M. Pfeifle, "DBDC: Density Based Distributed Clustering" in *Proceedings of EDBT in Lecture Notes on Computer Science 2992,* pp.88-105, 2004

[5] A. Lazarevic, D. Pokrajac, and Z. Obradovic, "Distributed Clustering and Local Regression for Knowledge Discovery in Multiple Spatial Databases," in *Proceedings of the 8th European Symposium on Artificial Neural Networks,* pp 129-134, 200

[6] "Peer-to-peer reconciliation based replication for mobile computers" (1996) *Proceedings of the ECOOP Workshop on Mobility and Replication*

[7] D. Kempe, A. Dobra, and J. Gehrke (2003) Gossip-Based Computation of Aggregate Information *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*

[8] W. Kowalczyk, and N. Vlassis (2005) "Newscast EM" *NIPS-17*

[9] L. Lamport, "Time, Clocks, and the Ordering of Events in a Distributed System" (1978) *Communications of the ACM Vol. 21 Number 7*

[10] M. Delio (2003) "Wireless Caravan: Geeks on Parade" *Wired Magazine*

[11] J. Garretson, W. Hess, J. Kanarek, M. Pignol, M. Shai http://roadcasting.org

[12] S. Bandyopadhyay, C. Gianella, U. Maulik, H. Kargupta, K. Liu, and S. Datta. (2004). "Clustering Distributed Data Streams in Peer-to-Peer Environments" *Accepted for publication in the Information Science Journal, in press*

[13] R. Wolff, K. Bhaduri, H. Kargupta. (2005). "Local L2 Thresholding Based Data Mining in Peer-to-Peer Systems" *UMBC Technical Report TR-CS-05-11*

[14] BreezeCom Wireless Communication, (1996) "A Technical Tutorial on the IEEE 802.11 Standard"

[15] P, Ramano, "The Range vs. Rate Dilemma of WLANs", *Wireless Net DesignLine*, 1/2004