A Scalable Distributed Stream Mining System for Highway Traffic Data*

Ying Liu[1], Alok Choudhary[2], Jianhong Zhou[3], and Ashfaq Khokhar[3]

[1] Graduate University of Chinese Academy of Sciences
Data Technology and Knowledge Economy Research Center, Chinese Academy of Sciences
Beijing, China 100080
yingliu@gucas.ac.cn
[2] Electrical and Computer Engineering Department, Northwestern University
Evanston, IL, USA 60208
choudhar@ece.northwestern.edu
[3] Computer Science Department, University of Illinois at Chicago
Chicago, IL, USA 60607
{jzhou13, ashfaq}@uic.edu

Abstract. To achieve the concept of smart roads, intelligent sensors are being placed on the roadways to collect real-time traffic streams. Traditional method is not a real-time response, and incurs high communication and storage costs. Existing distributed stream mining algorithms do not consider the resource limitation on the lightweight devices such as sensors. In this paper, we propose a distributed traffic stream mining system. The central server performs various data mining tasks only in the training and updating stage and sends the interesting patterns to the sensors. The sensors monitor and predict the coming traffic or raise alarms independently by comparing with the patterns observed in the historical streams. The sensors provide real-time response with less wireless communication and small resource requirement, and the computation burden on the central server is reduced. We evaluate our system on the real highway traffic streams in the GCM Transportation Corridor in Chicagoland.

Keywords: data stream, distributed computing, real-time, traffic, sensor.

1 Introduction

Advances in computing and communication over wired and wireless networks have resulted in many pervasive distributed computing environments, such as PDAs, cell phones, sensors, etc. Data in such applications is increasingly getting transformed to continuous data streams, which are dynamically changing, exhibit high volume, have high dimensionality, and are potentially infinite in length. Several issues related to mining stream data have been investigated in [1, 2, 3, 4, 8, 11]. The ever-increasing computational capacity of ubiquitous equipments presents an opportunity for intelligent data analysis to be performed "anytime, anywhere" with constrained resources. Stream mining on pervasive computing devices has a broad range of applications. For example, commercial fleet management companies spend a lot of

* This work was done during the first author's doctoral study in Northwestern University.

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, pp. 309 – 321, 2006.
© Springer-Verlag Berlin Heidelberg 2006

time and labor in collecting vehicle performance data, studying the data offline, and estimating the condition of the vehicle primarily through manual efforts. If on-board PDA was installed in every vehicle and connected with the remote server through wireless networks, real-time analysis of vehicle data streams would be achieved by the PDAs with less communication with the server. Patient health monitoring, forest fire monitoring, and security surveillance by sensor networks are also good examples.

Mining of highway traffic data, such as that of Gary-Chicago-Milwaukee (GCM) Corridor Transportation System, is a typical example of distributed stream mining application, where hundreds of sensors collect and send the traffic status data to a central server all day long over the expensive wireless and wired connections. Currently, a central server collects all the data and performs congestion analysis offline. This traffic analysis results are then published to the travelers through a central interface. However, this centralized process has several shortcomings: 1) over 1.2 GB data is sent to the server every day (5 MB per sampling) in GCM, which must be a communication burden on low bandwidth networks. 2) The huge amount of data per day is also a heavy burden on the storage system. 3) The traffic information provided to the travelers is not a real-time response due to the communication overhead. 4) This traffic data corresponds to only a small fraction of roads in the Chicago area and thus is likely to increase over time when more roads are equipped with sensors. There is an astonishing fact that traffic congestion wastes 2 billion gallons of fuel per year in the United States alone. 135 million US drivers spend 2 billion hours trapped in congestion per year. The total cost to Americans due to traffic congestion exceeds $100 billion per year. It would be highly beneficial to predict the congestion level as early as possible so that the drivers can avoid being trapped by choosing another route in advance. Therefore, there is an urgent demand for a traffic analysis system where traffic data is processed and mined in a distributed fashion, and the sensors are able to perform stream data analysis techniques on the fly (such as abnormal events real-time detection, traffic jam prediction, flow speed prediction, etc.). This demand for such a system is likely to increase with the increase in the use of mobile database devices inside the vehicles. As a matter of fact, this problem is very difficult because of the following issues:

1) Very little research has been done in distributed stream mining. Most of the existing algorithms are designed to work as a centralized application. This type of approaches is not scalable when the number of ubiquitous devices is large and cannot provide real-time response.
2) Although some distributed data mining algorithms have been proposed in [10], they don't consider the unique characteristics of data streams that the patterns change dynamically.
3) Sensors are lightweight devices, short of power supply, computation capability and storage size. Data mining on resource-constrained devices is not well explored yet.

In this paper, we propose a scalable distributed traffic stream mining system. It is designed to monitor the current roadway traffic status and predict the coming traffic in real-time. The proposed system consists of the following four phases: preprocessing, training, monitoring/predicting and updating. In the training phase, the central server performs various offline data mining techniques on the cleaned historical data streams, and ships the discovered patterns back to the sensors. Based

on the similarity of the patterns, it also groups the sensors into clusters. In the monitoring/predicting phase, each sensor predicts the coming traffic using the global patterns. If an "abnormal" event is predicted or observed at a sensor, the sensor raises an alarm immediately. The alarm is sent to the central server which can notify all the members in the group. Real-time analysis is achieved because each sensor works independently and incurs no communication delay. The updating phase is triggered periodically or when the number of mispredictions exceeds a threshold. There are three main characteristics of our real-time distributed traffic stream mining system: 1) The central server takes the major computation tasks in the training phase and the updating phase using its strong computation capability; 2) the sensors or similar lightweight devices perform predicting or monitoring using their constrained resources; 3) the data mining techniques used in the updating phase update the patterns efficiently. The design of this system tries to target the three challenges mentioned above. The proposed framework can be applied to similar applications, like forest fire monitoring, vehicle health monitoring, etc.

Data from a real roadway transportation system (GCM) is used in our experiments to evaluate our proposed system. It shows good scalability in various aspects. Memory usage on each sensor and communication overheads are small.

The rest of this paper is organized as follows. Section 2 briefly introduces the related algorithms and systems. Section 3 describes a highway transportation system. Section 4 presents the overall architecture of the real-time distributed traffic stream mining system and describes the methodology in each phase, respectively. Experimental results are presented in Section 6, and Section 7 summarizes this paper.

2 Related Work

2.1 Stream Mining

Lossy Counting [8] presents an algorithm for computing frequency counts exceeding a user-specified threshold over data streams. Gianella et al. [3] proposed an efficient approach to mine time-sensitive frequent patterns. It incrementally maintains only the historical information of (sub)frequent patterns. In order to adapt quickly to the changes of the underlying data stream, Aggarwal et al. [2] trains and tests streams simultaneously by selecting an appropriate window of past data to build the classifier. In [4] Guha et al. proposed a constant-factor approximation algorithm for K-Median problem in data stream clustering. This algorithm maintains a consistently good quality using a small amount of memory and time.

2.2 Time-Series Data Mining

A time-series database consists of sequences of values or events changing with time. Similarity search [12] finds sequences that differ only slightly from a given sequence. Similarity search analysis is useful in stock data analysis, cardiogram analysis, traffic patterns, power consumption analysis, etc.

2.3 Distributed Stream Mining

As many large transaction databases are available, [15] proposes Fast Distributed Mining (FDM), which generates a small number of candidate sets and reduces the number of messages to be passed. [14] extends the traditional ARM to peer-to-peer computing environments. This algorithm combines ARM, executed locally at each node, with a majority voting protocol to discover all the rules that exist in the combined database. It is asynchronous and communication efficient.

VEDAS, a real-time on-board stream mining system for vehicle-health-monitoring and driver status characterization, is developed in [6]. The PDAs perform most of the data management and mining tasks and send the analysis results to the central server site. This system minimizes bandwidth usage by limiting the centralization of the sensed data. The mobile stock monitoring system in [7], and the health monitoring tool in [5] are similar to VEDAS in terms of conceptual design. Our proposed system is sensor networks based, where the sensors have very limited computation capability and resources. Therefore, the central server has to take the main computation tasks.

2.4 Roadway Traffic Management System

Contemporary roadway traffic management systems have investigated issues related to route planning, automatic accident detection, short-term travel prediction, and better user interfaces for answering these questions. Advanced traffic management systems such as TrafficWise (Indiana's Intelligent Transportation System) rely on traffic flow speed sensors and cameras, as well as information from emergency responders and roadside assistance patrols, to detect traffic problems and determine the underlying causes. The information is centrally collected, analyzed, and then the results are delivered back to drivers, dispatchers, and emergency responders. Grossman et al. [13] have developed a framework that detects real-time changes in highway traffic data and send real-time alerts. However, the data collected at different sensors are centralized to a supercomputer cluster system for analysis. In contrast, our system doesn't need to transmit any data streams unless updating the patterns.

3 Gary-Chicago-Milwaukee (GCM) Corridor Transportation Data

The GCM Corridor (Figure 1) consists of the 16 urbanized counties and 2,500 miles of roadways which connect the three cities. 855 sensors are placed on the roads, and each sensor collects 288 streams every day (one sampling every 5 minutes). Each sensor collects the real-time traffic data at its location and sends it to the central server over wireless connections periodically. Each stream consists of static attributes (longitude, latitude, length, direction, etc.) and dynamic attributes (vehicle speed, congestion level, occupancy, volume, etc.)

GCM travel system is providing a number of services to travelers, for example, in Figure 1, the different colors on the roadways mean different congestion levels. Currently, all of the data analysis is still performed offline on a central server. It is not real-time response because the sensors collect data every 5 minutes. In addition, the computation burden on the server is heavy.

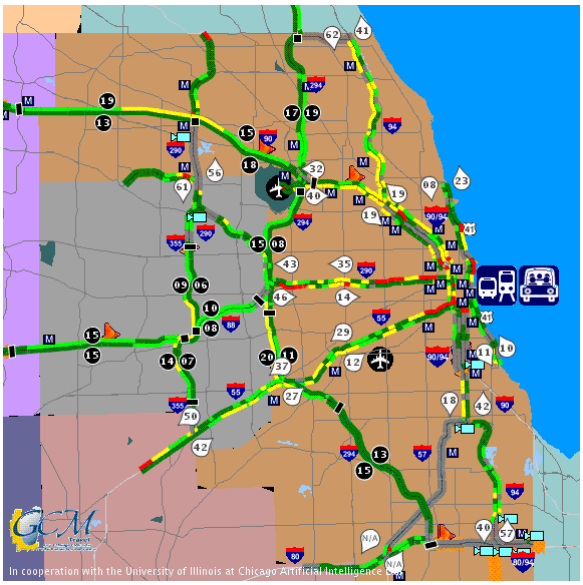

Fig. 1. Gary-Chicago-Milwaukee Corridor Transportation System

4 Framework of Distributed Traffic Stream Mining System

In order to reduce data transmission and computation time of the traditional centralized stream mining model, we propose a distributed traffic stream mining system. In our proposed system, the central server performs data mining techniques to discover patterns in the training phase and update patterns in the updating phase, respectively. The sensors perform monitoring and predicting tasks. This system incurs little data transmission except the transmission for the discovered patterns, alerts and the new streams for rebuilding the model. As the sensors in a roadway traffic system usually do not have as sufficient power supply or computation capability as on-board PDAs in VEDAS, most of the computation has to be assigned to the central server. Figure 2 shows the framework of our proposed system, which consists of four phases: *preprocessing, training, monitoring/predicting, and updating*. Each phase is described in detail in the following subsections.

5.1 Preprocessing Phase

The central server collects streams from the distributed servers, then, extracts time-series attributes that dynamically change with time (vehicle speed, congestion level, etc.) from the raw data, and eliminates the static attributes. Clean the outliers and fill up the missing numbers.

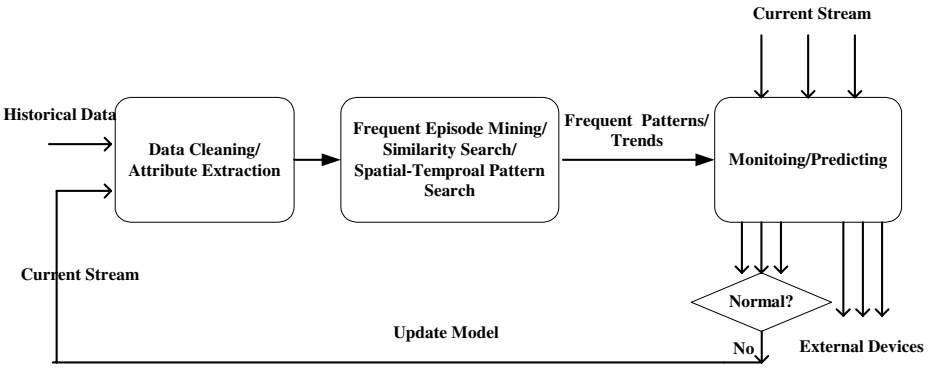

Fig. 2. Framework of the distributed traffic stream mining system

5.2 Training Phase

In this phase, the central server performs various data mining techniques on the preprocessed data. A number of interesting patterns or trends can be mined. Then, the interesting patterns are relayed to the sensors over the wireless network connections. Since the congestion level prediction is very valuable to travelers as mentioned in Section 1, we use "frequent episode mining" technique, which can help predict the congestion level, as an example to illustrate the training phase. We would like to emphasize that there are several algorithms can be applied to this application. However, due to limitations in space, we only present the details of one technique, although we would like to provide more.

Frequent Episode Mining. In roadway traffic data, the "congestion level" observed by each sensor can be viewed as a sequence (A sequence is a series of events where each event has an associated timing stamp.). A congestion level sequence database is created by putting all the sequences from all the distributed sensors together. An example of a congestion level sequence taking place at a sensor is

(non, 06:05), (light, 06:10), (medium, 06:15), (light, 06:20), (medium, 06:25), (heavy, 06:30)

The numbers indicate the timings when the events happened. The episode "light_congestion followed by medium_congestion" occurs twice in a window of 6 events.

Frequent episode mining is to find the collections of events which occur so frequently that exceed a pre-defined threshold [9] in the sequence database. Note that, here, we are only interested in the episodes where the events have no intervals in between. In the example sequence, if the threshold is set at 2, "non_congestion followed by medium_congestion" is not considered as a frequent episode because "medium_congestion" doesn't follow "non_congestion" immediately. Figure 3 presents the pseudo code of the frequent episode mining algorithm used in our system. It adopts an Apriori-like level-wise searching approach that generates new candidates and calculates their supports by scanning the database at each level. Remember that all the subepisodes of any frequent episode must be frequent. Please note line 6 in Figure 3. For each occurrence of candidate episode c in sequence s, the support of c is increased by 1, which is different from the support counting in association rule mining

```
Input: sequence database SD, congestion level set E, min support threshold min_fr
Output: frequent episodes collection FE
1.  C_1 := {{e} | e ∈ E};
2.  i := 1;
3.  while C_i ≠ ∅ do
4.    forall s ∈ SD do          //scan database, calculate support
5.      forall c ∈ C_i do
6.        for each occurrence of c in s, c.support ++;
7.      end forall
8.    end forall
9.    forall c ∈ C_i do      //identify frequent episodes
10.     if c.support > min_fr
11.       add c to FE_i;
12.     end if
13.   end forall
14.   build C_{i+1} from FE_i;//generate candidate episodes
15.   i := i + 1;
16. end while
17. FE := ∪ FE_i;
```

Fig. 3. Pseudo code of the frequent episode mining algorithm

where the support of c is only increased by 1 no matter how many times c occurs in s. In addition, we keep track of which sensor and when each frequent episode happens.

The frequent congestion level episodes can help transportation domain experts answer questions such as how the episodes evolve over time by sorting the interesting patterns by the timing stamps, how a congestion propagates spatially (what is the origin of the congestion and by what routes the congestion propagates) by combining the knowledge of the spatial relations (distance, direction, inter-connection, etc.) between the corresponding sensors, etc. Spatial effects are indirectly taken into account because we do observe unusual effects of traffic pattern in one area to another one in a "seemingly unrelated area", which is not so obvious.

Consistent Pattern. We proceed to find the *consistent patterns*. Assume we have N days' data and a user-specified threshold M. If a frequent episode *fe* happens on M out of the N days at a sensor A at a certain time *t,* we call (A, *t, fe*) a *consistent pattern*. All the sensors that have a common *fe* are clustered into a same group. A counter is maintained in each group to record the number of updating request. Finally, we send the consistent patterns to the corresponding sensors over wireless connections. We are interested in the consistent patterns and the corresponding sensors because the patterns tend to re-occur. In addition, since the sensors in the same group show similar traffic flow trends for known or unknown reason, a sensor's coming traffic could be predicted based on the most recent observations from its group members.

One may have a question that why not to let each sensor predict its coming traffic by using its local statistical values, such as the mean speed of the traffic flow

observed in the past few days at a given time, or the mean congestion level in the past few days at a given time, or the observations of the last day. The answer is in the following three aspects: 1) the storage cost is higher than using our proposed framework and algorithms. A sensor needs a relatively large memory to store the streams in the past few days in order to maintain the statistical numbers. In contrast, in our system, each sensor only stores the frequent patterns delivered by the central server. 2) It requires higher computation capability and power consumption for a sensor to calculate the statistical values, which may exceed the capability of any similar lightweight devices in pervasive environments. In contrast, the computation on each sensor in our system is very simple: Compare the new observations with the patterns; if not matching, send an updating request to increase its group counter by 1, otherwise, do nothing. 3) Using local statistics, a sensor only sees its local view with no chance to know the global view of the neighboring areas, which actually may impact its traffic flow. A group of sensors that share a common *consistent pattern* may have similar traffic behaviors. Therefore, it may be more confident to predict the coming traffic at a certain sensor based on an event happened a moment ago. For example, consider that sensors *A, B,* and *C* always experience the same congestion level sequence (*light, medium, medium, medium*) from *6:30am*, *6:50am*, and *7:20am* for the subsequent 20 minutes, respectively. *A, B,* and *C* may or may not be spatially connected to each other. If today, somehow, the congestion level sequence observed at *A* at *6:30am* is (*medium, heavy, heavy, heavy*), it is highly likely that this early rush hour jam will happen at *B* and *C* soon. So *A* will send a warning to the server, and then the server will notify *B* and *C* so that they can send out "early traffic jam" warning signals to its local external devices in advance.

Predictive models (decision tree, logistic regression, Naive Bayesian classifier, etc.) could be used to predict if a heavy congestion would happen in the next moment. However, either the computation or the storage cost (there must be a number of rules for different time slots) is larger than those of the frequent episode mining algorithm.

5.3 Monitoring/Predicting Phase

Each sensor stores its significant patterns mined from the recent historical data, such as the frequent congestion level episodes at some timing points, the speed fluctuating patterns, etc. The monitoring/predicting phase is designed to predict the local roadway coming traffic flow and detect abnormal events deviating from the observed patterns. For example, sensor *X* predicts the forthcoming congestion level *l*. If *l* is a severe situation, then *X* will send a warning to its external device and an alarm to the server as well. Then, the server will notify all the members of the cluster containing *X*. If *l* is different from the actual level *l'*, sensor *X* will send an updating request to augment the counter in its corresponding group on the server. The server maintains a counter for each group. The aim of the predictions is to provide travelers with up-to-date information so that they can choose routes to avoid being stuck in traffic jams.

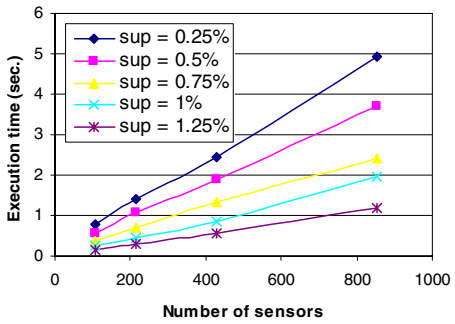

Fig. 4. Execution time on one day's data with varying number of sensors. The number of sensors varied from 107 to 855.

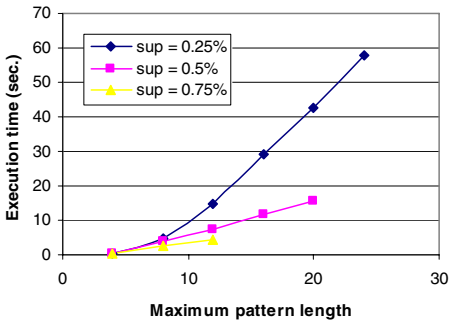

Fig. 5. Execution time on one day's data. The maximum pattern length varied from 4 to 24.

5.4 Updating Phase

While monitoring their local traffic streams, the sensors send updating requests to augment the counter in its corresponding group on the server whenever any misprediction happens. The server maintains a counter for each group. Once the counter exceeds a user-specified threshold, the server starts to download new streams from the sensors which send out the requests. Thus, the data transmission cost is much smaller than that of collecting new streams from all the sensors. It is not a heavy burden for the central server, and will not exhaust the sensors' power.

One important feature of data stream is that its patterns change dynamically, therefore, there is no need to keep the old patterns when updating the patterns. The server replaces the oldest day's data in the database with today's new data. Then, it starts to perform some corresponding data mining algorithms. Finally, the updated patterns are relayed to the sensors over wireless connections. The members in this specific group may have to wait for one day to be informed the latest patterns.

6 Experimental Results

We evaluate our distributed traffic stream mining system on real traffic streams from a roadway transportation system, Gary-Chicago-Milwaukee (GCM) Corridor Transportation System (See details of GCM in Section 3). For the purpose of our experiment, we extract the dynamic attribute "congestion level". Each sensor of the 855 sensors contributes a sequence of 288 congestion levels every day. There are four different congestion levels: *non, light, medium, and heavy*. We download the traffic streams of 5 weekdays.

In this section, we only discuss the performance of frequent episode mining. Its scalability and prediction accuracy are analyzed. Communication overheads and memory usage are also discussed. All the experiments are performed on a 700-MHz

Table 1. Traffic congestion level prediction accuracy

Days for training	Threshold	Days for predicting	Avg. cluster size	Accuracy
day_1,day_2,day_3	3	day_4	10.3	25.2%
day_1,day_2,day_3	3	day_5	10.3	7%
day_2,day_3,day_4	3	day_5	22.4	11.9%
day_1,day_2	2	day_3	106.4	9.7%
day_1,day_2	2	day_4	106.4	7.7%
day_1,day_2	2	day_5	106.4	8.6%
day_2,day_3	2	day_4	105.8	21.5%
day_2,day_3	2	day_5	105.8	8.9%
day_3, day_4	2	day_5	291.8	6.6%

Xeon 8-way shared memory parallel machine with a 4GB memory, running the Red Hat Linux Advanced Server 2.1 operating system. The program is implemented in C.

6.1 Scalability in Training Phase

Since the central server performs the data mining tasks in the training stage, we would like to investigate the scalability of the data mining algorithms on the server, specifically the scalability of frequent episode mining when varying the number of sensors, minimum support or maximum pattern length.

As the number of sensors in different systems may be different, the scalability when varying the number of sensors is evaluated in Figure 4. The number of sensors is varied from 107 to 855. The maximum length of a frequent episode is set to be 8. The execution time on one day's (24 hours) congestion level sequences scales well when the number of sensors is increasing.

In order to provide real-time congestion prediction, short episode is more useful. Thus, we restrict the maximum length of the frequent episodes in each run. The total number of frequent episodes increases as the maximum length increases. Figure 5 presents the execution time on one day's (24 hours) congestion level sequences. The support threshold is varied from 0.25% to 0.75%. When the support is set at 0.25%, the execution time is increasing fast because both the number of frequent episode candidates and the number of database scans are huge due to the low support threshold. For the case of 0.5% and 0.75%, it scales well as the length of episodes increases.

6.2 Traffic Prediction

As described in Section 5.2, all the sensors (associated with the same timing *t*) that have a common "consistent frequent episodes *fe*" are clustered into the same group. Then the consistent frequent episodes are sent to the corresponding sensors over wireless connections for monitoring or predicting purpose. We use a fraction of the 5 weekdays' data for training and the rest for predicting. Table 1 shows the prediction

Table 2. Average number of patterns on each sensor

Days for training	Consistent pattern threshold	Avg. # of patterns / sensor
day_1, day_2, day_3	3	0.46
day_2, day_3, day_4	3	0.46
day_3, day_4, day_5	3	0.97
day_1,day_2	2	4.6
day_2, day_3	2	4.6
day_3,day_4	2	14

accuracy. The maximum pattern length is set to be 8 because users are more interested in short-term predictions. The support threshold is 0.75%. Overall speaking, the prediction accuracy is not high. The best case happens when using day_1, day_2 and day_3's patterns to predict day_4's traffic, where the overall accuracy is 25.2%. In addition, in this case, 27% of the sensors get 100% prediction accuracy (All the predictions made at these sensors match the actual streams.). The results in this table verify the dynamic nature of traffic data streams.

As described in the monitoring/predicting phase in Section 5.3, if an "abnormal" event is observed, the sensor sends an alert to the central server, and then, the server notifies all its group members. In our experiments, we observe that the warnings are pretty helpful in some cases. For example, sensor #337, #415, #731 and #732 are in the same group because they have (*light, light, light*) at *1:00am, 1:20am, 3:10am,* respectively. The congestion levels on sensor #337 at *1:00am* was (*medium, medium, medium*), deviating from the pattern (*light, light, light*). Then, (*medium, light, medium*) happened on sensor #415 at *1:20am*, (*medium, medium, light*) happened on sensor #731 and #732 at *3:10am*. If the server sent an "abnormal congestion" alert to sensor #415, #731 and #732 at *1:15am*, they would have sent warning signals to the drivers in advance, so that they could avoid the jam near sensor #731 and #732 at *3:10am*. This observation verifies our claim that congestions may propagate spatially and temporally between roadways where similar behaviors often happen.

6.3 Storage

As the storage resource on a sensor or a lightweight computing device is very limited, we would like to investigate the cost for storing the patterns. Table 2 presents the average number of patterns stored on each sensor. We use 3 days', and 4 days' data for training and set the "consistent" pattern threshold as 2, 3, respectively. From Table 2 we can see that the average number of patterns on each sensor is quite small for any case. Therefore, our proposed system doesn't require a large storage resource.

6.4 Communication Cost

The number of mispredicitons is small, ranging from 300 to 3000 per day. Thus, the data transmission cost must be small, because there is no data transmission between

the central server and a sensor unless a misprediction happens. In contrast, if the central server were to download all the streams from all the sensors to analyze and then provide up-to-date information, the data transmission cost must be huge.

7 Conclusions and Open Issues

This paper presented a scalable distributed traffic stream mining system. In the training phase, the central server performs various data mining techniques on historical data streams to find useful patterns. In the monitoring/predicting phase, each sensor predicts the forthcoming data or raises alarms if an abnormal situation is predicted by comparing with the patterns in recent historical streams. Our experiment results on the real traffic data from GCM demonstrate that this model is scalable and effective. The "abnormal" congestion warning is helpful. This system can provide travelers with real-time traffic information and help transportation domain experts understand the characteristics of a transportation system. It achieves real-time analysis with low communication cost. In addition, the computation resource requirement on each sensor is small as well as the storage requirement.

The proposed system is at an early stage of development and will be substantially enhanced by incorporating the following aspects:

1) The most challenging part is how to build a model to involve both temporal and spatial features. We plan to use connected graph to represent the spatial relations between different sensors, and incorporate graph mining algorithms to our system. The prediction accuracy must be improved a lot by then.
2) In order to discover more interesting patterns, more stream mining techniques applicable to traffic streams should be explored.
3) Building of a simulation environment where we can see how the "early" warnings affect the traffic flow.

Acknowledgments. This work was supported in part by National Science Foundation grants IIS-0536994, CNS-0551639, CNS-0406341, Intel, IBM, CNS-0550210, and in part by National Natural Science Foundation of China Project #70531040, #70472074, Ministry of Science and Technology of China 973 Project #2004CB720103.

References

1. Aggarwal, C., Han, J., Wang, J., and Yu, P. S.: A Framework for Clustering Evolving Data Streams. In Proc. Intl. Conf. on Very Large Data Bases (VLDB), 2003
2. Aggarwal, C., Han, J., Wang, J., and Yu, P. S.: On Demand Classification of Data Streams. In Proc. 2004 Intl. Conf. on Knowledge Discovery and Data Mining (KDD), 2004
3. Giannella, C., Han, J., Pei, J., Yan, X., and Yu, P. S.: Mining Frequent Patterns in Data Streams at Multiple Time Granularities. H. Kargupta, A. Joshi, K. Sivakumar, and Y. Yesha (eds.), Next Generation Data Mining, 2003
4. Guha, S., Mishra, N., Motwani, R., and O'Callaghan, L.: Clustering Data Streams. In Proc. IEEE FOCS Conf., 2000

5. Jorge, A.: Adaptive Tools for the Elderly: New Devices to Cope with Age-induced Cognitive Disabilities. In Proc. EC/NSF workshop on the Universal accessibility of ubiquitous computing: providing for the elderly, 2001, 66-70
6. Kargupta, H., Bhargava, R., Liu, K., Powers, M., Blair, P., Bushra, S., Dull, J., Sarkar, K., Klein, M., Vasa, M., and Handy, D.: VEDAS: A Mobile and Distributed Data Stream Mining System for Real-Time Vehicle Monitoring. In Proc. SIAM International Conf. on Data Mining, 2004
7. Kargupta, H., Park, B. H., Pittie, S., Liu, L., Kushraj, D., and Sarkar, K.: MobiMine: Monitoring the Stock Market from a PDA. SIGKDD Explorations, Jan. 2002, 3(2): 37-46
8. Manku, G. S., and Motawani, R.: Approximate Frequency Counts over Data Streams. In Proc. 28th Intl. Conf. on Very Large Databases (VLDB), 2002
9. Mannila, H., Toivonen, H., and Verkamo, A. I.: Discovering frequent episodes in sequences, In Proc. Intl. Conf. on Knowledge Discovery and Data Mining, 1995
10. Park, B., and Kargupta, H.: Distributed Data Mining: Algorithms, Systems, and Applications. Data Mining Handbook, 2002
11. Toivonnen, H.: Sampling large database for association rules. In Proc. 22nd Intl. Conf. on Very Large Databases, 1996, 134–145
12. Agrawal, R., Faloutsos, C., and Swami, A.: Efficient Similarity Search in sequence databases. In Proc. 4th Intl. Conf. Foundations of Data organization and Algorithms, 1993
13. Grossman R., Sabala, M., Alimohideen, J., Aanand, A., Chaves, J., Dillenburg, J., Eick, S., Leigh J., Nelson, P., Papka, M., Rorem, D., Stevens, R., Vejcik, S., Wilkinson, L., and Zhang, P.: Real Time Change Detection and Alerts from Highway Traffic Data, In Proc. Intl. Conf. Super Computing, 2005
14. Wolff, R., and Schuster, A.: Association Rule Mining in Peer-to-Peer systems. ICDM, 2003
15. Cheung, D., Han, J., Ng V., Fu, A. W., Fu, Y.: A Fast Distributed Algorithm for Mining Association Rules. In Proc. of PDIS, 1996