Towards detection of faulty traffic sensors in real-time

Nikolas Zygouras, Nikolaos Panagiotou,
Ioannis Katakis, Dimitrios Gunopulos {NZYGOURAS,N.PANAGIOTOU,KATAK,DG}@DI.UOA.GR
University of Athens, Athens, Greece

Nikos Zacheilas, Ioannis Boutsis, Vana Kalogeraki {ZACHEILAS,MPOUTSIS,VANA}@AUEB.GR
Athens University of Economics and Business, Athens, Greece

Abstract

Detecting traffic events using the sensor network infrastructure is an important service in urban environments that enables the authorities to handle traffic incidents. However, irregular measurements in such settings can derive either from faulty sensors or from unpredictable events. In this paper, we propose an efficient solution to resolve in real-time the source of such irregular readings by examining the correlation and the consistency among neighbor sensors and exploiting the wisdom of the crowd. Our experimental evaluation illustrates the efficiency and practicality of our approach.

1. Introduction

Sensor network infrastructures have been widely used for traffic management in smart cities to provide important services for the benefit of pedestrians, cyclists, motorists and public transport. Such services are typically provided by analyzing data provided by heterogeneous static and mobile sensors. This enables the implementation of numerous applications like proposing alternative routes, altering traffic lights, etc.

The most common type of sensor which is utilized in such environments is the SCATS sensor. They are static sensors embedded at the city roads providing rich, real-time information such as traffic flow measurements based on vehicles that cross a specific segment. Despite their utility in many traffic applications, SCATS sensors can be faulty. Thus, one fundamental challenge in these settings is how to efficiently distinguish between irregular and faulty measurements before taking any unnecessary actions.

Proceedings of the 2^{nd} International Workshop on Mining Urban Data, Lille, France, 2015.

Automatic identification of anomalies in streaming data is an emerging field of research due to the large number of applications (intrusion detection, event identification, etc). Many algorithms that utilize machine learning and time series analysis techniques have been successfully applied for the detection of unexpected events during the last years (Yi et al., 2000). These methods offer high quality results and are able to perform on massive data streams in real-time. An interesting use-case is the automatic analysis of traffic data generated by Smart Cities infrastructures. Human personnel are unable to monitor and efficiently identify problems on these data. The utilization of anomaly detection techniques would provide great assistance to traffic operators as it would enable the automatic real-time identification of traffic issues.

Recently, Crowdsourcing has emerged as an attractive paradigm to exploit the intelligence of ubiquitous human crowd (citizens) to extract useful information. Traditional Crowdsourcing systems such as AMT[1], CrowdFlower[2], etc., constitute marketplaces for human intelligence tasks (HITs), that allow a requester to define a task, which is performed by other human workers in exchange for a reward. For example, mobile human workers with different characteristics can be queried for geo-located tasks to extract real-time information without needing an expensive infrastructure(Boutsis & Kalogeraki, 2014).

In this paper we develop an efficient approach that identifies faulty readings from traffic sensors by examining the correlations among them and by taking advantage of the ubiquitous citizens through Crowdsourcing. We summarize our contributions below:

- We present an efficient approach that identifies anomalous sensors and uses Crowdsourcing to resolve whether irregular measurements are due to faulty sensors or irregular traffic.

[1] http://www.mturk.com/
[2] http://www.crowdflower.com/

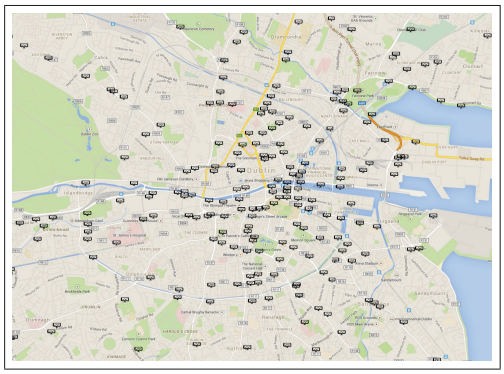

Figure 1. The SCATS sensors locations at Dublin's city centre

- We tackle the problem of automatically detecting anomalous SCATS sensors with three methods: (i) Pearson's correlation, (ii) cross-correlation and (iii) multivariate ARIMA model. The proposed methods have to tackle the task efficiently in real-time.
- We develop our approach using the Lambda-Architecture which combines a batch processing framework (i.e. Hadoop[3]) and a distributed stream processing system (i.e. Storm[4]) for efficiently processing both historical and real-time data.
- We develop a Crowdsourcing system used to extract answers from the human crowd based on the MapReduce paradigm.
- We provide an experimental evaluation, which illustrates that our approach is practical and can effectively identify irregular measurements in real-time.

2. Problem Description and System Model

2.1. Smart City

Smart cities exploit digital sensor devices that can be either embedded at the city infrastructure or they can be mobile (*e.g.*, smartphones) in order to provide services for their citizens that enhance their well-being. Such services may relate to traffic management, housekeeping information, etc.

In this paper we focus on Dublin, a smart city that utilizes sensors for supervising and managing road traffic (Kinane et al., 2014). In Dublin the traffic is controlled by the Dublin City Council (DCC), which is responsible to develop, maintain and manage the city road network. To achieve that they exploit several heterogeneous data sources that include: (i) SCATS sensors which are embedded on the road and monitor real-time traffic density, (ii) GPS traces from sensors embedded on buses, (iii) the LiveDrive radio where users can report traffic, and (iv) pedestrian counters.

2.2. System model

In this section we provide our system model for the data sensors that we examine, namely the SCATS sensors and Crowdsourcing.

SCATS Sensors. SCATS (Sydney Coordinated Adaptive Traffic System) is an innovative computerized traffic management system developed by Roads and Maritime Services (RMS) Australia. SCATS sensors are fixed magnetic sensors deployed on intersections to measure the traffic flow and the degree of saturation of roads' lanes. In Dublin city, each SCATS sensor produces and transmits a new record every minute. Each record contains information related to the timestamp t of the measurement, the sensor's ID i and finally the degree of saturation and traffic flow measurements. In the provided dataset there are approximately 300 SCATS controlled intersections and 1000 different SCATS sensors throughout the road network. The GPS locations of the SCATS sensors are presented in Figure 1. Degree of saturation measures how much a road's lane is utilized, while traffic flow measures the vehicles' volume divided by the highest volume that has been measured in a sliding window of a week[5]. In this work we decided to monitor the degree of saturation value, noted as s, as it more reliable and informative than the traffic flow. The degree of saturation of a particular SCATS sensor with $ID\ i$ at the timestamp t is noted $s_{i,t}$.

Crowdsourcing. Our crowdsourcing system comprises a set of human workers denoted as w_j which are able to receive task assignments. Tasks are being inserted to the system by an authority, such as the DCC. Each task t_k is associated with a number of attributes as $< id_k, latitude_k, longitude_k, reward_k, description_k >$. Hence, every task posses a unique identifier (id_k), the geographical coordinates of the location that the task involves ($latitude_k$, $longitude_k$), the corresponding reward ($reward_k$) for executing the task and a task description that describes the information that needs to be provided by the human worker. An example of such a task description is: *"Is there traffic in O'Connell Street? Yes/No"*. Finally each response provided by a worker is captured with a record by our system using the worker and the task identifiers, coupled with the response as follows: $< w_j, id_k, response_{jk} >$.

3. Architecture

In Figure 2 we display our system architecture which consists of the following components: (i) a Distributed

[3] https://hadoop.apache.org/

[4] https://github.com/nathanmarz/storm

[5] http://dublinked.com/datastore/datasets/dataset-274.php

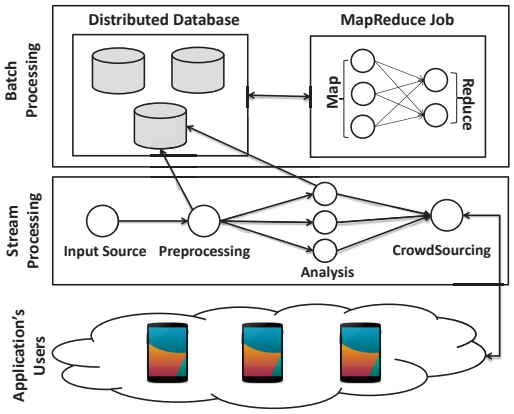

Figure 2. System Architecture

Stream Processing System (DSPS), (ii) a batch processing framework, (iii) a distributed database system, and (iv) the Crowdsourcing component that consists of the users' mobile devices. Our architecture is an instance of the Lambda-Architecture[6] as we exploit the fast processing offered by DSPS and the fault-tolerance and parallelism provided by current batch processing frameworks.

Incoming SCATS-sensor data are forwarded to a stream processing graph. These data are pre-processed and stored in the Distributed Database (i.e. Preprocessing component in Figure 2) for further processing by the batch processing component. We analyze the reported metrics via the Analysis component which examines if one of the sensors deviates significantly from its neighbors so it could possibly be a faulty sensor. This component uses both the current conditions and historical data for identifying such conditions. In case that one such sensor is detected, the Analysis component informs the Crowdsourcing component about this situation. The latter is responsible to send the appropriate Crowdsourcing tasks that will enable us to detect if the sensor is a faulty-one. Finally, the batch processing component periodically computes new statistics about the historical sensor data.

There are multiple DSPSs which support low latency processing in real-time. Some of these systems are Apache Storm, Spark Streaming[7] and TUD-Streams (Bockermann & Blom, 2012). We used Storm as the DSPS that will perform the real-time processing of incoming sensor data. Storm is one of the most commonly used DSPS, and is supported by major companies such as Twitter[8]. It has been successfully applied for processing high volume of data in different application domains, achieving high throughput and low response latencies (McCreadie et al., 2013). Furthermore, we decided to use Storm due to its scalability features that we also exploit in our previous work (Zygouras et al., 2015). Storm users can change the parallelism of the processing components to adapt to possibly workload bursts.

Finally, for the analysis of the historical sensor data we used the most commonly used open-source implementation of the MapReduce programming model, Hadoop. We execute periodical (i.e. at the end of each day) Hadoop jobs for computing the basic metrics required by our proposed techniques, described in more detail in Section 4. Our jobs retrieve historical data from a distributed database, more specifically MongoDB[9]. We decided to use MongoDB instead of the Hadoop Distributed Filesystem (HDFS), as we want to have fast access to the data from the DSPS component of our architecture, for computing and storing short-term statistics in real-time.

4. Methodology

The goal of this work is to monitor the streaming traffic data and automatically pose Crowdsourcing tasks when anomalous sensors are identified. In order to identify anomalous sensors we propose three different outlier tests that examine whether the SCATS sensors behave differently from their normal behavior. These outlier tests are based on the following statistical measurements: (i) Pearson's Correlation (ii) Cross-Correlation and (iii) the ARIMA Model. The normal behavior for each sensor is calculated offline using the historical data. These methods are implemented using the Lambda architecture and Crowdsourcing tasks are assigned to users when anomalous SCATS sensors are identified.

4.1. Identifying Anomalous Sensors

In this section we describe the three statistical measurements that are used and we explain how these are utilized to detect anomalous SCATS sensors. Initially we applied a simple statistic measurement named Pearson's correlation that identifies the correlation between pairs of SCATS sensors. Then we used an extension of the first method, named cross-correlation, to identify how many lags we should shift backward a sensor's values to maximize its pairwise correlation with another adjacent sensor. The first two approaches use two well known measures in time series analysis. The disadvantage is that they check pairs of sensors and not the group of sensors as a whole. For this reason we applied a third approach that can be thought as a multivariate ARIMA model which deals with the aforementioned problem and is faster than the other approaches.

[6] lambda-architecture.net
[7] https://spark.apache.org
[8] http://twitter.com
[9] http://www.mongodb.org/

4.1.1. PEARSON'S CORRELATION

The Pearson's correlation coefficient is a well known statistic that measures the linear relationship of two variables X and Y. It takes values in $[-1, 1]$, where 1 means that the variables are positively correlated, -1 stands for negative correlation and 0 for no correlation between X and Y. The Pearson's correlation, noted $\rho_{X,Y}$, is calculated by dividing the covariance of X and Y with the product of the standard deviations of X and Y (see Equations 1 and 2).

$$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y} \tag{1}$$

$$cov(X,Y) = E[(X - \mu_X)(Y - \mu_Y)] \tag{2}$$

In our scenario we calculated the pair-wise correlation between all SCATS sensors X and Y whose spatial distance does not exceed a predefined threshold. This restriction creates a sparse correlation matrix that contains non-zero elements when SCATS sensors are spatially adjacent. We calculate the sparse correlation matrix from the historical data. Then, utilizing the streaming data that arrive continuously in our system we periodically calculate the streaming correlation of the adjacent SCATS sensors. We note as noisy sensors' pairs those that their streaming correlations disagree significantly with the correlations calculated from the historical data. If a particular sensor disagrees significantly with the majority of his neighbors then a crowdsourcing task is posed.

4.1.2. CROSS-CORRELATION

Cross-correlation is a statistical measure of similarity between two variables X and Y as a function of the lag of one relative to the other. More specifically cross-correlation between X and Y is calculated by shifting forward or backward Y and calculating its correlation coefficient with X. Cross-correlation with lag d, noted $\rho_{X,Y}(d)$, is calculated as seen in Equation 3. The numerator of the equation calculates the covariance of X and Y shifted d time bins backward. Finally the denominator is the product of the standard deviations of X and the lagged Y.

$$\rho_{X,Y}(d) = \frac{\sum_i [x(i) - \mu_X)(y(i+d) - \mu_Y)]}{\sqrt{\sum_i (x(i) - \mu_X)^2} \sqrt{\sum_i (y(i+d) - \mu_Y)^2}} \tag{3}$$

A traffic anomaly at a particular location, in a road network, may require some time in order to be propagated to the adjacent sensors. This observation motivates us to consider the cross-correlation between adjacent SCATS sensors. More specifically we calculated the d_{max} that maximized the correlation between two adjacent sensors X and Y (see Equation 4).

$$d_{max} = \arg\max_d(\rho_{X,Y}(d)) \tag{4}$$

In order to identify anomalies with cross-correlation we followed a similar approach to the one utilizing the Pearson's correlation measure, described before. The main difference is that we identified, using historical data, the lag d_{max} that maximized the correlation between two SCATS sensors X and Y. In the streaming analysis in order to calculate the cross-correlation between the sensors we shifted d_{max} lags backward the Y and we calculated its correlation with X. Finally, we measured how much the streaming cross-correlation deviates from the offline calculated cross-correlation between X and Y using the optimal lag value d_{max}.

4.1.3. MULTIVARIATE ARIMA MODEL

A common strategy to detect outliers in multivariate time series (Yi et al., 2000) is to build a regression model for each time series and evaluate whether the actual values vary significantly from the predictions. The model receives as input the previous L degree of saturation measurements for a particular sensor with $ID = 0$ and the sensor's N nearest SCATS sensors $\{s_{i,j} : i \in [0, N], j \in [0, L], i, j \in \mathbb{Z}\}$. The goal of the model is to make the best prediction for $s_{0,t}$, denoted as $\hat{s_{0,t}}$. The model is presented in detail in Equation 5. This model can be thought as a multivariate ARIMA model, as multiple sensors are used in order to make the predictions.

$$\begin{aligned} \hat{s_{0,t}} =& \phi_{0,1} s_{0,t-1} + \cdots + \phi_{0,L} s_{0,t-L} + \\ & \phi_{1,0} s_{1,t} + \phi_{1,1} s_{1,t-1} + \cdots + \phi_{1,L} s_{1,t-L} + \\ & \cdots \\ & \phi_{N,0} s_{N,t} + \phi_{N,1} s_{N,t-1} + \cdots + \phi_{N,L} s_{N,t-L} \end{aligned} \tag{5}$$

In the training phase we use the historical degree of saturation values in order to calculate the coefficients Φ of Equation 5. In order to solve this problem we created the matrix A and vector b containing the input data (degree of saturation values) and the target values respectively. The Φ parameters are the values that optimally solve Equation 6. The solution of this system is given with the pseudo-inverse transformation of the input presented in Equation 7. The key property of this approach, in contrast to the two previously described techniques, is that it monitors the different sensors together as a whole. The Pearson's correlation and the cross-correlation approaches investigated only pair-wise correlation between SCATS sensors, ignoring potentially useful information. On the other hand, the ARIMA-based method aims at exploiting this information.

$$\Phi = \begin{bmatrix} \phi_{0,1} & \dots & \phi_{0,L} & \dots & \phi_{2,0} & \dots & \phi_{2,L} \end{bmatrix}$$

$$A = \begin{bmatrix} s_{0,t-1} & \dots & s_{0,t-L} & \dots & s_{N,t} & \dots & s_{N,t-L} \\ s_{0,t-2} & \dots & s_{0,t-L-1} & \dots & s_{N,t-1} & \dots & s_{N,t-L-1} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ s_{0,L} & \dots & s_{0,0} & \dots & s_{N,L} & \dots & s_{N,0} \end{bmatrix}$$

$$b = \begin{bmatrix} s_{0,t} & \dots & s_{0,L+1} \end{bmatrix}^\top$$

$$b = A\Phi \tag{6}$$

$$\hat{\Phi} = (A^T A)^{-1} A^T b \tag{7}$$

In order to integrate this approach we split the historical data in training and test set. Initially we calculated offline, using the training set, the $\hat{\Phi}$ parameters. These parameters are the coefficients regarding the sensor's previous measurements and its adjacent sensors' past measurements. Then we calculated how well the data fitted to these models computing for each sensor its Mean Absolute Error (MAE). Finally, in order to identify anomalous SCATS sensors while monitoring the streaming data we compute for each sensor its MAE at a particular time window. We label a sensor as 'anomalous' if its streaming MAE noticeably differs from its MAE measured using the testing set.

4.2. Implementation

Our system calculates the correlation among adjacent SCATS sensors. This is achieved by adding the SCATS sensors' GPS locations in a k-d tree data structure during system initialization and calculating the k nearest SCATS sensors for each sensor. Furthermore, we developed our system using the Lambda architecture. So we should ensure that the required data are transmitted to the appropriate cluster nodes. Thus, we created a mapping of each SCATS sensor ID to one or more cluster nodes. This guarantees that each computing node contains all the required data for a sensor's adjacent sensors.

We define three parameters that help us configure the components of our system. The first one is $job\_periodicity$ and defines when the batch jobs should re-execute (e.g. each day, every week). The other two control the stream processing computations. More specifically, the $stream\_threshold$ parameter defines how often we should re-compute the examined metrics (e.g. every ten minutes), while $time\_window$ defines the sliding time window (e.g. the previous hour) that will be used for keeping the past sensor data necessary for the computations.

As we described in Section 3, we periodically invoke Hadoop jobs that compute the different metrics we explained in Section 4.1. Map tasks read the pre-processed sensor data from the MongoDB, and send them to the reduce tasks. We partition the data based on the SCATS sensor ID to cluster node mapping. The idea is that neighboring sensors should always end up on the same reduce task in order to appropriately compute the examined metrics. Each sensor may belong to more than one nodes in such cases we send the tuple multiple times (i.e. equal to the number of nodes it is part of) to avoid information loss.

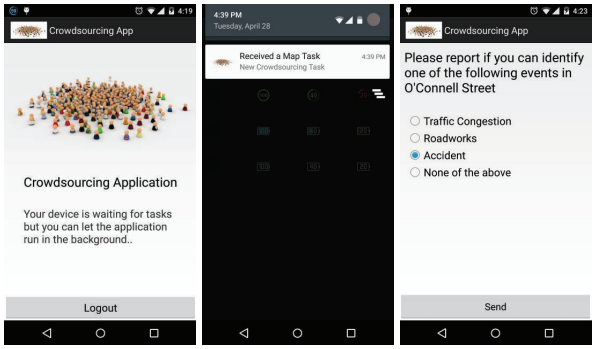

Figure 3. Crowdsourcing Application (a) Main Application, (b) Push Notification, (c) Map Task

Reduce tasks are responsible for computing the metrics described in Section 4.1 and store the results in MongoDB.

In the stream processing component, we implemented a Storm topology (see Figure 2) that processes the real-time sensor data. We exploit the parallelism offered by Storm by having multiple instances of our Analysis component, running in parallel, in order to decrease latency. The topology pre-processes the incoming data and stores them in MongoDB. Also the pre-processed data are sent to the component that invokes the three different techniques. Again we partition the data based on the offline mapping, to guarantee that all neighboring data will be processed by the same component's instance. Detected events are forwarded to the Crowdsourcing component that is responsible to inform the users that will help us detect if the sensor is faulty.

4.3. Crowdsourcing System

Misco. Our Crowdsourcing system has been developed using the Misco framework (Dou et al., 2011; 2010; Kakantousis et al., 2012), which is based on the MapReduce paradigm and tailored for mobile devices to provide an extensible and efficient way to develop distributed applications.

Our Crowdsourcing system is structured using (i) a *Master Server* that keeps track of the tasks t_k submitted when anomalies are detected from SCATS sensors, assigns them to human workers w_j and returns the responses to the system, and (ii) the *Workers* who are the human contributors that process the crowdsourcing tasks. Each Worker is responsible to process queries and return the results to the server. These tasks are executed by workers through their personal smartphone devices or tablets.

Task assignment. Suppose that we need to exploit Crowdsourcing to determine the source of an event using a task t_j. We describe the step-by-step sequence followed so as to process the task and return the results. In the implementation described below we considered Android-based devices

and thus we have utilized the Android SDK[10].

For every task t_j, the Master Server spans the task to a set of *map* tasks that need to be forwarded to the human workers w_k. Since these tasks are geo-located only the workers that reside close to the specific selection need to be selected by the Master Server to provide information. However, in order to avoid tracking the users we follow a different policy. We forward the task to all the workers and the tasks are locally filtered at the mobile devices if their location is far from the location of the task t_j.

We use Push Notifications services to initiate the communication with the human workers, to be able to send the Map task to the users without being restricted by their connection (WiFi, 3G, etc). Such services exist in all major mobile operator systems and allow users to register for message delivery when they are online through a connection server.

In order to be able to receive map tasks, each user first needs to login to our system so that the Master server will be aware of the user. At the same time the user also registers in the push notification service to retrieve its unique id. During normal operation the Crowdsourcing applications runs in the background (Figure 3a).

When the Master Server retrieves a new task t_j from the requester, it delivers a push notification to the user devices with the task, through the Push Notification service. Once the device receives the notification it examines whether the user current location is close to the location of the task so as to alert the user (Figure 3b). Next, if the user selects the notification on his mobile device the Crowdsourcing application is triggered and the task will be displayed in the user screen to process the task (Figure 3c).

Finally, the responses for each *map* task are forwarded to the Master Server that initiates the *reduce* phase to aggregate the answers. The reduce phase is performed through Majority Voting. Hence the Master Server identifies the response $response_{jk}$ for task t_k with the maximum amount of answers from all users w_j and forwards the response that represents the cause of the event to the system.

Crowd Feedback. The response retrieved by the crowdsourcing component enables the system to determine whether the irregular readings derive from an unexpected event (*e.g., roadworks*) or if the sensor is indeed faulty when most of the workers answer "None of the above".

5. Evaluation

We have evaluated our proposals on our local cluster consisting of 4 VMs. Each VM had two CPU processors attached and $3,096$ MB of RAM. All VMs were connected to the same LAN and their clocks were synchronized using the NTP protocol. The frameworks we used were the following: Storm 0.8.2, Esper 5.1 and MongoDB 2.6.5.

[10]Android platform: http://www.android.com/

Parameter	Value
$job\_periodicity$	24 hours
$stream\_threshold$	10 minutes
$time\_window$	1 hour

Table 1. Basic Configuration Parameters

In Table 1, you can see the values of the basic configuration parameters described in Section 4.2. For the experiments, we used SCATS data from the period of April and May of 2014. The distance threshold used for the neighbouring sensors computation was set to 250 meters. Data from April were used in order to calculate the historical correlations, cross-correlations as well as the ARIMA models. On the other hand, data from May were used for different experimental runs (see below).

For the Pearson Correlation method we have stored the historical correlations of the neighbour-pairs in the MongoDB component. 7116 neighbour pairs were identified under the distance threshold from a set of 900 SCATS sensors. The correlation value ranged from almost perfect correlation, for sensors of the same junction under different lane, to no correlation at all for more distant sensors. Negative correlation values between nearby sensors were also observed. This could be explained by the opposite direction of the lane the sensors are responsible for. In Figure 4, the correlation matrix for a set of 30 nearby sensors is presented. As expected, clusters are formed by adjacent sensors that are highly correlated. Thus, it is reasonable to argue that when the expected correlation is not observed there might be a problem with the sensor. For the Cross-Correlation method apart from storing the correlation value itself we have also stored the time lag that maximizes the pair-wise sensor correlation. The time lag range was set to a maximum of 10 minutes since the sensors are quite close to each other and larger time lags are unlikely to significantly favor the correlation value. In addition, the larger the time lag range is, the more computationally demanding the method will be. As it was expected, in most cases the highest cross-correlation was observed with no time lag at all, since most sensor pairs are responsible for different lanes of the same highway junction. However, for more distant sensor-pairs responsible for different highway junctions, small time lags gave a boost on their correlation value. One way to understand this is because vehicles require a short time to reach consecutive junctions. In addition, this behaviour could be also explained by the operation of traffic lights that transfer the traffic from junction to junction on fixed time intervals. Figure 5 depicts the distribution of the optimal time

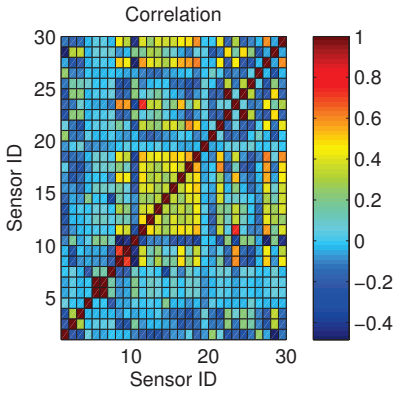

Figure 4. The correlation matrix of a set of 30 neighbors

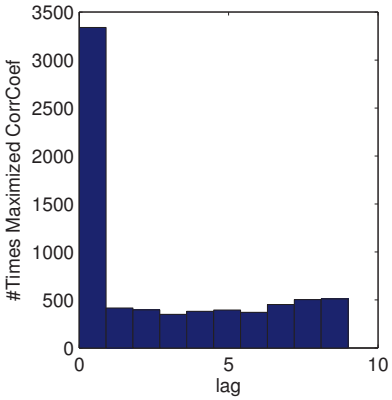

Figure 5. Distribution of the optimal time lag value over all the neighbour-pairs

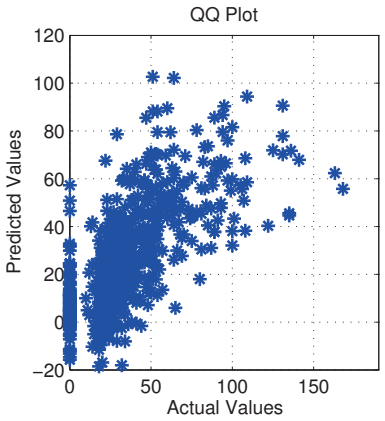

Figure 6. The predicted and the actual degree of saturation values over the validation dataset

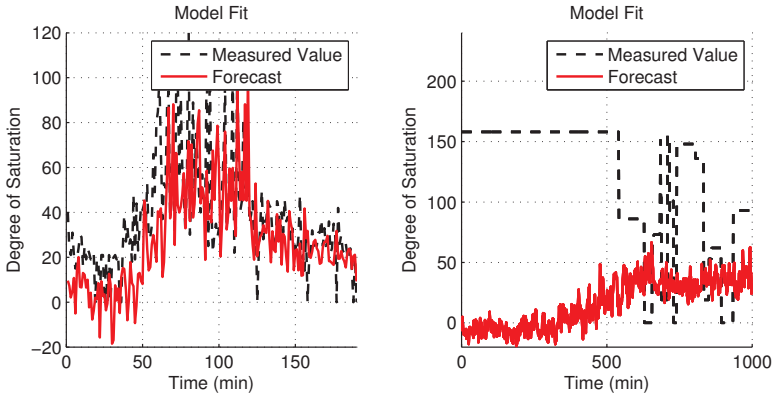

Figure 7. On the left there is a sensor whose values agree well with our forecast. On the right there is a noisy sensor whose values diverge significantly from the predicted and it is considered as faulty

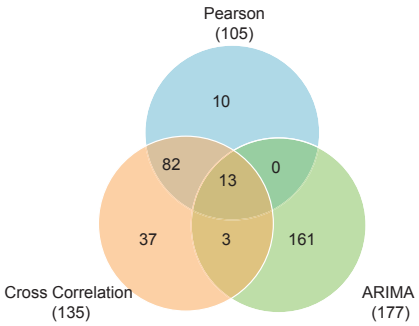

Figure 8. Venn diagram for the three proposed methods

lag value over a sample of neighbor pairs.

In terms of the multivariate ARIMA method, a different model was fitted to each sensor using as features all the neighbor sensors. The performance of all the models was aggregated and measured in terms of Mean Absolute Error (MAE), Root Mean Squared Error(RMSE) and Correlation Coefficient (CC) using a validation dataset. The results follow on Table 2 with a low MAE value of 16.18 indicating a decent fit.

Figure 6 shows the forecasting performance of the ARIMA model on the validation dataset and Figure 7 gives an example on forecasting two different sensors. Sensors such as the one presented in Figure 7 (left) will be considered as non-faulty since the deviation between the observed measurements and the expectation is not significant. On the other hand, sensors such as the one in Figure 7 (right), given that it reports maximum values for a long period of time, it is likely that it is faulty. These sensors are flagged by our system for further manual evaluation or inspection from the traffic operators.

The three methods were compared in terms of the number of faulty sensors they identify. In addition, since the Correlation and the ARIMA approaches focus at a very different aspect of the same problem we measured the overlap between their results. Figure 8 displays the Venn diagram of the results obtained over the period of one day during May of 2014. As it was expected, the results of Pearson Correlation and Cross-Correlation are highly overlapping since for many sensors the optimal time lag is zero. On the other hand, the ARIMA method identified different sensors as erroneous suggesting that the methods are complementary to each other. Interestingly enough, 13 sensors were identified as erroneous from all methods indicating that sensors operate in an unexpected way in many settings and are more likely to be faulty.

Metric	Result
CC	0.68
MAE	16.8
RMSE	23.18

Table 2. The measurements that indicate the performance of the multivariate ARIMA model

6. Related Work

Traffic monitoring has been a field of great interest in the scientific community (Biem et al., 2010), (Patroumpas & Sellis, 2012). These works detect unusual events based on pre-defined *rules* so any updates to the traffic conditions overtime is not taken into account. In contrast, our proposal exploits historical data for updating the expected sensor correlations and detects events only when the real-time conditions deviate significantly from the expected. Authors in (Ma et al., 2013) propose a novel city transportation application that enables sharing of taxi rides in a large city. Their goal was to develop an application that is beneficial for both the citizens and the taxi drivers.

There has been significant work in traffic monitoring in the use-case of Dublin. (Artikis et al., 2014) proposed a traffic management system, based on heterogeneous data, which used Crowdsourcing in order to resolve conflicting sensors reports. (Zygouras et al., 2015) focused on monitoring the traffic conditions of the city by considering the metrics reported from sensors mounted on top of public buses. While (Liebig et al., 2014a) and (Liebig et al., 2014b) perform individual trip planning that considers future traffic hazards in routing. Furthermore, their approach estimates the expected traffic flow in areas with low sensor coverage.

Anomaly detection methods have been widely applied for mining data streams including techniques such as data clustering (Guo et al., 2009), principal component analysis (PCA) (Lakhina et al., 2004), wavelet transform (Novakov et al., 2013) and many others. Some detection methods follow a time series analysis perspective and focus on forecasting methods such as ARIMA (Zare Moayedi & Masnadi-Shirazi, 2008; Fujimaki et al.). ARIMA models are a wide family of analysis and forecasting models that are used widely in forecasting urban traffic time series data (Lee & Fambro, 1999; Williams et al., 1998). This makes ARIMA models suitable for our scenario. (Niennattrakul et al., 2010), used distance-based outlier detection techniques, reducing the size of the original database, in order to efficiently identify outliers in massive streaming datasets. (Schettlinger et al., 2010) proposed an online time series filter, using repeated median regression, which is able to smooth the data and keep intact the signal's trend. (Branch et al., 2013) developed a distributed and in-network model in order to detect outliers on network exchanges among neighboring nodes. (Fried et al., 2015) proposed a Bayesian approach to model time series of counts, using Metropolis-Hastings algorithm in order to estimated the parameters of the model.

7. Conclusions

In this paper we presented an efficient approach for resolving whether irregular sensor measurements are due to faulty sensors or unexpected traffic. Our approach exploits sensors' past measurements and the crowd's wisdom for decision making. We implemented our proposals using the Lambda-Architecture for processing real-time and historical data, and an Android application for extracting answers from the human crowd. We applied three different outlier detection techniques that identified complementary set of faulty sensors. Finally, our detailed experimental evaluation indicates that our approach can effectively resolve the source of irregular measurements in real-time.

Acknowledgments

This research has been co-financed by the European Union (European Social Fund ESF) and Greek national funds through the Operational Program Education and Lifelong Learning of the National Strategic Reference Framework (NSRF) - Research Funding Program: Thalis-DISFER, Thalis-CompGeom, Aristeia-MMD Investing in knowledge society through the European Social Fund, the FP7 INSIGHT project and the ERC IDEAS NGHCS project.

References

Artikis, A., Weidlich, M., Schnitzler, F., Boutsis, I., Liebig, T., Piatkowski, N., Bockermann, C., Morik, K., Kalogeraki, V., Marecek, J., Gal, A., Mannor, S., Kinane, D., and Gunopulos, D. Heterogeneous Stream Processing and Crowdsourcing for Urban Traffic Management. *in Proc. 17th International Conference on Extending Database Technology (EDBT), Athens, Greece, March 24-28, pp. 712-723*, 2014.

Biem, Alain, Bouillet, Eric, Feng, Hanhua, Ranganathan, Anand, Riabov, Anton, Verscheure, Olivier, Koutsopoulos, Haris, and Moran, Carlos. IBM Infosphere Streams for Scalable, Real-time, Intelligent Transportation Services. *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, 2010.

Bockermann, C. and Blom, H. The streams framework. *Technical Report 5, TU Dortmund University*, 2012.

Boutsis, Ioannis and Kalogeraki, Vana. On task assignment for real-time reliable crowdsourcing. In *ICDCS*, pp. 1–10, Madrid, Spain, June 2014.

Branch, JoelW., Giannella, Chris, Szymanski, Boleslaw, Wolff, Ran, and Kargupta, Hillol. In-network outlier detection in wireless sensor networks. *Knowledge and Information Systems*, 34(1):23–54, 2013. ISSN 0219-1377. doi: 10.1007/s10115-011-0474-5. URL http://dx.doi.org/10.1007/s10115-011-0474-5.

Dou, Adam, Kalogeraki, Vana, Gunopulos, Dimitrios, Mielikainen, Taneli, and Tuulos, Ville H. Misco: a mapreduce framework for mobile systems. In *PETRA*, June 2010.

Dou, Adam Ji, Kalogeraki, Vana, Gunopulos, Dimitrios, Mielikinen, Taneli, Tuulos, Ville, Foley, Sean, and Yu, Curtis. Data clustering on a network of mobile smartphones. In *SAINT*, pp. 118–127, Munich, Germany, July 2011.

Fried, Roland, Agueusop, Inoncent, Bornkamp, Bjrn, Fokianos, Konstantinos, Fruth, Jana, and Ickstadt, Katja. Retrospective bayesian outlier detection in ingarch series. *Statistics and Computing*, 25(2): 365–374, 2015. ISSN 0960-3174. doi: 10.1007/s11222-013-9437-x. URL http://dx.doi.org/10.1007/s11222-013-9437-x.

Fujimaki, Ryohei, Yairi, Takehisa, and Machida, Kazuo. An anomaly detection method for spacecraft using relevance vector. In *Learning, The Ninth Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD*, pp. 785–790. Springer.

Guo, Feng, Yang, Yingzhen, and Duan, Lian. Anomaly detection by clustering in the network. In *Computational Intelligence and Software Engineering, 2009. CiSE 2009. International Conference on*, pp. 1–4. IEEE, 2009.

Kakantousis, Theofilos, Boutsis, Ioannis, Kalogeraki, Vana, Gunopulos, Dimitrios, Gasparis, Giorgos, and Dou, Adam. Misco: A system for data analysis applications on networks of smartphones using mapreduce. In *MDM*, pp. 356–359, Bengaluru, India, July 2012. IEEE.

Kinane, D., Schnitzler, F., Mannor, S., Liebig, T., Morik, K., Marecek, J., Gorman, B., Zygouras, N., Katakis, Y., Kalogeraki, V., and Gunopulos, D. Intelligent synthesis and real-time response using massive streaming of heterogeneous data (insight) and its anticipated effect on intelligent transport systems (its) in dublin city, ireland. In *ITS*, Dresden, Germany, November 2014.

Lakhina, Anukool, Crovella, Mark, and Diot, Christiphe. Characterization of network-wide anomalies in traffic flows. In *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pp. 201–206. ACM, 2004.

Lee, Sangsoo and Fambro, Daniel B. Application of subset autoregressive integrated moving average model for short-term freeway traffic volume forecasting. *Transportation Research Record: Journal of the Transportation Research Board*, 1678(1):179–188, 1999.

Liebig, Thomas, Piatkowski, Nico, Bockermann, Christian, and Morik, Katharina. Predictive trip planning-smart routing in smart cities. In *EDBT/ICDT Workshops*, pp. 331–338, 2014a.

Liebig, Thomas, Piatkowski, Nico, Bockermann, Christian, and Morik, Katharina. Route planning with real-time traffic predictions. In *Proceedings of the 16th LWA Workshops: KDML, IR and FGWM, Aachen, Germany*, pp. 83–94, 2014b.

Ma, Shuo, Zheng, Yu, and Wolfson, Ouri. T-Share: A Large-Scale Dynamic Taxi Ridesharing Service. *ICDE*, 2013.

McCreadie, Richard, Macdonald, Craig, Ounis, Iadh, Osborne, Miles, and Petrovic, Sasa. Scalable Distributed Event Detection for Twitter. *BigData Conference: 543-549*, 2013.

Niennattrakul, V., Keogh, E., and Ratanamahatana, C.A. Data editing techniques to allow the application of distance-based outlier detection to streams. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pp. 947–952, Dec 2010. doi: 10.1109/ICDM.2010.56.

Novakov, Stevan, Lung, Chung-Horng, Lambadaris, Ioannis, and Seddigh, Nabil. Studies in applying pca and wavelet algorithms for network traffic anomaly detection. In *High Performance Switching and Routing (HPSR), 2013 IEEE 14th International Conference on*, pp. 185–190. IEEE, 2013.

Patroumpas, Kostas and Sellis, Timos. Event Processing and Real-time Monitoring over Streaming Traffic Data. *Web and Wireless Geographical Information Systems Lecture Notes in Computer Science Volume 7236, pp 116-133*, 2012.

Schettlinger, K., Fried, R., and Gather, U. Real-time signal processing by adaptive repeated median filters. *International Journal of Adaptive Control and Signal Processing*, 24(5):346–362, 2010. ISSN 1099-1115. doi: 10.1002/acs.1105. URL http://dx.doi.org/10.1002/acs.1105.

Williams, Billy M, Durvasula, Priya K, and Brown, Donald E. Urban freeway traffic flow prediction: application of seasonal autoregressive integrated moving average and exponential smoothing models. *Transporta-*

tion Research Record: Journal of the Transportation Research Board, 1644(1):132–141, 1998.

Yi, B.-K., Sidiropoulos, N.D., Johnson, T., Jagadish, H.V., Faloutsos, C., and Biliris, A. Online data mining for co-evolving time sequences. In *Data Engineering, 2000. Proceedings. 16th International Conference on*, pp. 13–22, 2000.

Zare Moayedi, H. and Masnadi-Shirazi, M.A. Arima model for network traffic prediction and anomaly detection. In *Information Technology, 2008. ITSim 2008. International Symposium on*, volume 4, pp. 1–6, Aug 2008.

Zygouras, Nikolas, Zacheilas, Nikos, Kalogeraki, Vana, Kinane, Dermot, and Gunopulos, Dimitrios. Insights on a Scalable and Dynamic Traffic Management System. *EDBT*, 2015.