Finding Event Correlations in Federated Wireless Sensor Networks

Ismail Ari, Ömer F. Çelebi, *Member, IEEE*

***Abstract*—Event correlation engines help us find events of interest inside raw sensor data streams and help reduce the data volume, simultaneously. This paper discusses some of the challenges faced in finding event correlations over federated wireless sensor networks (WSNs) including high data volumes, uncertain or missing data, application-specific dependencies and widely varying data ranges and sampling frequencies. Analysis over real geo-tracking data of moving objects confirms some of these challenges. Federation at the data layer above the WSNs is presented as a feasible alternative.**

***Index Terms*—Event detection, Correlation, Middleware, Wireless Sensor Networks**

I. Introduction

WSN are used for real-time monitoring of physical environments. They help higher-level applications collect relevant data that can be transformed into actionable information. These applications include earthquake monitoring, asset tracking, traffic management, national security, green data centers [21], and recently regulatory hygiene-compliance tracking in hospitals [18]. People managing or using these applications are interested in detecting and even predicting concise "special events" (*e.g.* anomalies) upon which they can take an application-specific action.

Events are semantically different from primitive numeric sensor readings. For example, an event can refer to a numeric threshold violation or a more complex pattern such as an ordered (possibly nested) sequence of any datum. Finding complex event patterns in high-speed, unbounded, bursty data streams can be as challenging as finding a needle in a haystack. Sometimes checking only the "existence" of a simple reading in the stream may be of interest and sometimes we look for the "absence" of an event instead of its existence. Doing these becomes hard when the streams come from distributed sources. Ability to aggregate, order, join or correlate streaming data is the key to detecting many of these complex situations. Event correlation engines, some of which will be described here, help us describe these scenarios and find event patterns effectively. However, even the state-of-the-art systems cannot cope with today's real-time and distributed event processing challenges.

Assuring accurate environmental monitoring using Federated WSNs (FWSNs) and managing the scale is challenging. If only a few sensors are deployed, then the scale of the event-based application and its potential impact is limited. Also, when the density of the sensors over the area covered decreases the network becomes prone to disconnects. Alternatively, if millions of sensors are deployed (to increase the coverage and measurement accuracy), then both the WSNs and applications are faced with a data deluge, *i.e.* transferring, storing and near real-time processing of large data volumes. It is impractical to assume a fine-tuned homogeneous deployment model for FWSNs as done in cellular networks, since they are usually used in hostile environments and emergency conditions. If different phenomena are to be measured at different locations, then the cross-organizational nature of the network makes correlations impractical. As a result, while traditional WSNs provide continuous and relatively homogeneous data streams, FWSNs can carry numerous, heterogeneous, and possibly bursty data streams. Other challenges are listed at the end of this section.

Currently, many organizations still use Database Management Systems (DBMS) in an ad-hoc fashion to store and query sensor data. They face performance problems with time-window-based analysis over unbounded, high-volume streams, since DBMS architecture was designed for enabling offline analysis. An emerging system architecture called Data Stream Management System (DSMS) allows concurrent analysis over high-speed in-flight data with different continuous queries and is better suited for real-time applications. Complex Event Processing (CEP) middleware built on top of DSMS engines [1] promise a scalable alternative for WSNs data fusion or federation.

A. Motivating Scenario

Consider the data sample in Table 1 pertaining to only one vehicle collected from a real geo-tracking system. For this vehicle with the unique **Id** (00-123) the data shows the geo-location (**longitude and latitude**), **speed** and **time** information for every 20 seconds. However, due to intermittent disconnects or noise in the channel many data fields are prone to different types of errors. For example, while a normal value for the longitude and latitude fields would have 8 digits (*e.g.*

This work was supported in part by the Turkish National Institute of Science and Technology (TUBITAK) under Grant E190194, IBM Shared University Research program, and European Union FP7 Marie Curie Program under Grant BI4MASSES.

I. Ari and O. F. Celebi are at Ozyegin University, Istanbul, 34462 Turkey (Corresponding author: ismail.ari@ozyegin.edu.tr, +90-216-559-2331).

28,866,064, 41,052,856) we see that many entries lost several of their least-significant digits. Similarly, we find that the highly-varying speed information may also be erroneous. Note that some rows can be completely missing. For example, at minute 12/8/2009 7:26 only one measurement was recorded instead of three. Other potential data anomalies include accuracy errors and out-of-order arrivals [23]. This is only one data stream and yet there are thousands of vehicles and millions of objects that need to be tracked in FWSNs. Mass transportation administrators want the flexibility to be able to accurately track a single vehicle or average recordings from all vehicles on a certain route or certain region.

TABLE I
SAMPLE DATA FROM A REAL GEO-TRACKING APPLICATION

ID	LONGITUTE	LATITUDE	SPEED	DATE & TIME
00-123	28,863,169	4,105,348	42	12/8/2009 7:23
00-123	2,886,469	41,052,845	3	12/8/2009 7:23
00-123	28,866,064	41,052,856	26	12/8/2009 7:23
00-123	28,867,975	410,522	37	12/8/2009 7:24
00-123	2,886,879	4,105,189	1	12/8/2009 7:24
00-123	28,869,068	41,051,792	6	12/8/2009 7:24
00-123	28,869,884	41,051,376	16	12/8/2009 7:25
00-123	28,870,121	41,051,258	0	12/8/2009 7:25
00-123	2,887,055	41,051,044	16	12/8/2009 7:25
00-123	28,870,613	4,105,191	15	12/8/2009 7:26
00-123	28,868,597	4,105,249	46	12/8/2009 7:27
00-123	28,866,816	4,105,319	19	12/8/2009 7:27
00-123	288,657	41,053,898	20	12/8/2009 7:27

B. *Challenges*

Overall, the challenges and issues [6][13] in managing FWSNs data streams include:

- Limitations on communication range, power, CPU and memory of the wireless sensors and sensor networks resulting in broken data and out-of-order arrivals
- Need for real-time data cleansing and sanity checking and associated challenges
- Need to eliminate duplicate or unnecessary data to save resources without destroying the essence of information carried inside the streams and without missing critical events
- Finding correlations over high-speed raw or aggregated or sketched/summarized data streams [7][14][17]
- Having large number of streams to join and correlate; Holistic querying and monitoring over distributed streams
- Differences in observed data types (integer, float, string, date-time) and varying data values (0-1, 28868597, "BUS-00-130", etc.)
- Widely-varying data sampling frequencies (from microsecond, to seconds-minutes, to hours-days-months) among different sensor types.
- Lack of trust among organizations for sharing raw data. The need to operate over encrypted or compressed data (e.g. using fully homomorphic functions).
- Effective data visualization.

The rest of the paper is organized as follows. Section 2 overviews some modern event processing middleware that can be used to address some of these challenges. Section 3 describes the CEP concept, event query languages, and some details related to correlating geo-spatial stream data. Section 4 presents evaluation of CEP with event correlation operator and presents results with the vehicle tracking data. Section 5 concludes the paper and presents planned future work.

II. EVENT MIDDLEWARE

WSNs community generally holds the perspective that every computation can be and should be done at the network level, *i.e.* all data integration work and intelligence can be pushed onto the network. FWSNs show us one situation where this assumption can be questioned. Federation requires extreme planning either during sensor production, or by custom network/MAC protocol design or via organizational agreements. However, it would be hard to foresee potential future collaborations and data fusion needs. Therefore a data or event fusion middleware is necessary to provide flexible, extensible and high-performance WSNs federation.

Consider the following analogy: the human body has different types of sensors for vision, audio, smell, taste, and touch, but it takes a central entity (*i.e.* the brain) to correlate all the related inputs to make a real-time decision on how to respond to different situations. Similarly, Figure 1 shows different WSNs which represent different types of sensors and the event middleware level that represents the "enterprise nervous system". Following this analogy the real-time applications in Figure 1 would represent our arms, legs and speech that help us respond to external events.

Raw sensor data get transferred through wireless networks, message queues, raw sockets or other several protocols before

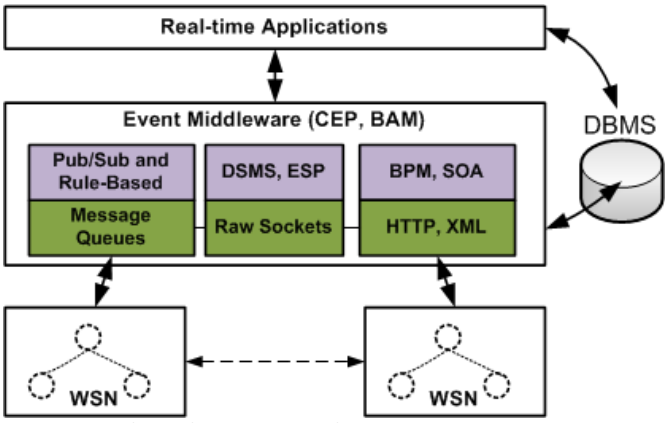


Figure 1. Preferred federation layer for the WSN is the event middleware from data fusion or correlation perspective. This layer is also referred to as the CEP layer. There are associated software products with these names.

reaching the event processing components. Event middleware software in Figure 1 are also called CEP or Business Activity Monitoring (BAM) products. Inside these products one usually finds other mainstream software engines. Publish-Subscribe and Rules-Based Systems allow users to describe certain conditions with IF-THEN rules. Registered rules get checked against all published events and subscribers are notified when there are matching events. DSMS [2] or Event Stream Processing (ESP) [11] systems allow querying over unbounded, streaming data. They have high-level query languages similar to SQL, but they also provide sliding-window semantics. Business Process Management (BPM) engines provide workflow and adapter capabilities for integrating different streams and services. When real-time event detection capabilities are coupled with Service-Oriented Architectures (SOA), the result is called an Event-Driven Architecture.

III. COMPLEX EVENT PROCESSING

We use a CEP engine called Esper [11] that is based on the DSMS architecture. DSMS systems parse, optimize and execute queries written in Continuous Query Language (CQL) [4]. CQL syntax and semantics are quite similar to that of Structured Query Language (SQL), but there are additional clauses such as WINDOW and EVERY that support sliding or tumbling window-based analysis over streams. CQL has basic stream filtering (SELECT x,y FROM Stream WHERE) queries as well as clauses for algebraic (COUNT [22], SUM, AVERAGE) and holistic (MIN, MAX) aggregation. More complex aggregation functions such as TOP-K [16], DISTINCT, QUANTILES, and SKYLINE can also be included in the CQL. DSMS engines underlying the CEP system provide effective queuing, scheduling, time-window support, and fast in-memory processing of high-speed streams [1][9][10]. CEP engines over DSMS add other correlation or sequence detection clauses into their query language to deliver concise and meaningful results. This new language is sometimes called an Event Processing Language (EPL) [3]. We start event detection with a basic statistical CORRELATION operator in this paper. For example, to find the correlation between the two moving vehicles we write the following queries in Esper:

```
SELECT Correlation1
FROM PairStream.
WIN:length(50).stat:correl(a.long, b.long)

SELECT Correlation2
FROM PairStream.
WIN:length(50).stat:correl(a.lat, b.lat)
```

where a PairStream is constructed by joining two streaming datasets (a sample was shown in Table 1). Then, out of these two datasets statistical Correlation of the latitude and longitude fields are calculated over a sliding window of 50 events and continuously output as a result stream. Esper supports both

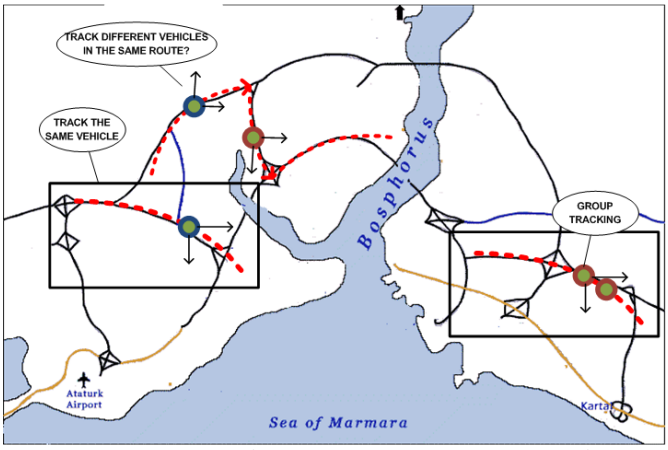


Figure 2. Detecting and enforcing in-route or group movement of vehicles using statistical correlations.

sliding and tumbling windows. Sliding windows can be time-based (*e.g.* capture events that occurred in the last 30 seconds) or count-based (*e.g.* last 50 events). The window can move smoothly (*e.g.* sliding every 1 second) or make big jumps (e.g. tumbling in 30 second increments) before publishing any results.

Figure 2 illustrates different uses of lat-long correlation information on a city map. Using statistical correlation, a moving vehicle can be correlated to its assigned route to assert in-route movement (a correct reference path is recorded beforehand), multiple vehicles on the same route can be correlated to spot erratic driving behavior, or assets moving in a supply-chain can be grouped together to assure intact delivery of goods to the distribution centers or retailers. However, we should carefully understand the issues with the new operator (*i.e.* Correlation) before we can use it for complex event detection. For example in Figure 2, two vehicles that are moving in different parts of the city (shown in rectangular boxes) on a very similar trajectory can have high correlation values since the latitude and longitude vectors are the same except only a spatial shift. Additional domain specific information may have to be used to detect whether these two vehicles are actually the same vehicle, are on the same route or belong to an asset group. Applying range queries (like the boxes in Figure 2) may be useful for assuring *spatial* relations.

The other challenge is related to the *temporal* component of the correlation. Vehicles on the same route pass from same points at different times and possibly move along the same trajectory with varying time-scales. Time-shifting and time-scaling is necessary to associate vehicles with their routes and other vehicles on the same route. We applied a time shift manually for the data used in this paper, but we plan to use either the Regressions or B-Splines techniques to create feature vectors for the routes and automatically correlate time-shifted and time-scaled events series in real-time. Finally, running these correlations over broken data is a big challenge and the only-solution is to do real-time data cleaning, pre-processing over the stream.

To summarize, correlation over geo-spatial streaming data is quite insensitive to spatial shifts, but is very sensitive to time shifts and broken data. We observe that running separate regressions on each variable can both correct broken fields and help us estimate missing or wrong values.

A. *Pearson Product-Moment Correlation*

To find the correlation between two scalar variables we use the Pearson Product-Moment Correlation Coefficient (PMCC) [25]. PMCC is defined as the covariance of the two variables divided by the product of their standard deviations. Its range is [-1,+1], +1 denoting a high positive correlation, 0 denoting no correlation and -1 denoting high negative correlation.

Correlation operation is "referential", *i.e.* it compares the movements of two numeric streams with respect to each other. Therefore, the results are relatively insensitive to the scale of the numbers. Other streaming data processing operators such as REGRESSION (calculated over a single vehicle and route) could also be prone to this "insensitivity to small changes in large numbers" problem.

Some complex situations also occur when certain events arrive at a preset order [15][24]. This type of event correlation uses an EPL clause called SEQUENCE. The associated query would have the following template:

SELECT SEQUENCE(A,B,C)
FROM Stream
WINDOW Length

where A,B,C are ordered events of interest. Note that many CEP engines either do not support the SEQUENCE operator at all or the supported version does not allow negations, SEQUENCE(A,**!B**,C), or nesting **SEQ**(A,**SEQ**(B,C),D) over the data streams.. In this paper, we only focus on statistical correlations and we discuss the design and implementation of sequence based event correlations in our related work [18].

IV. EVALUATION

GPS data from a single bus (00-130) moving along the same route on two different days (denoted as 7 and 8) was used for the evaluations in this section. Two different buses that move along the same route on the same day or different dates could also be used. We expect to find high correlations for vehicles that go over the same route, share sub-routes, or move together. Because the correlation is tracked over streaming data we will be able to detect emerging patterns such as traffic congestions or unique violations in real-time and publish short alerts as "interesting events to watch" to the related people. Note that some of this data may have to be stored or buffered before being correlated with others as well.

The experiments were executed by running the Correlation queries given in Section 3 over the open-source Esper engine on a personal computer with Intel i5 processor and 3GB memory. For visual confirmation we marked the movements of vehicles on Google Maps as shown in Figure 6. Each mark in the figure shows where each vehicle was during a specific

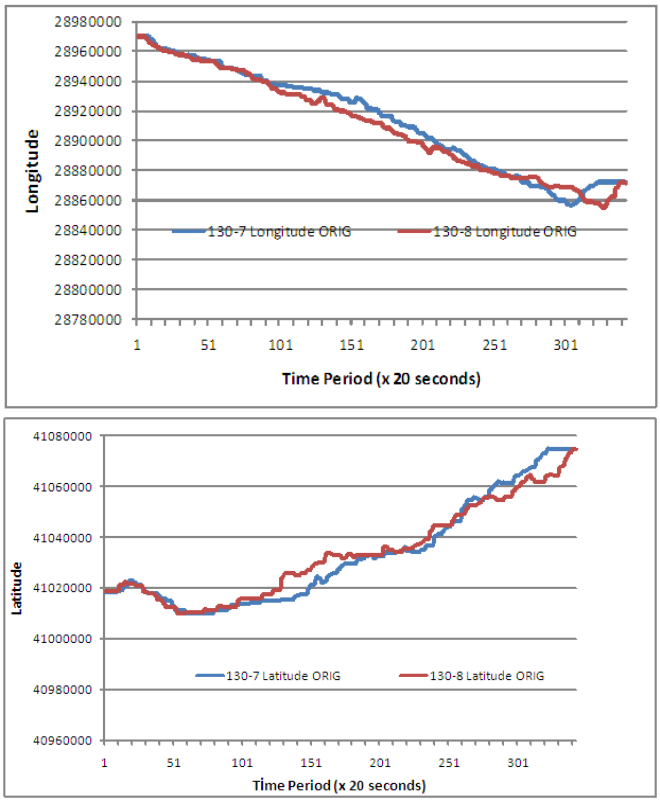


Figure 3. Longitude and Latitude of the same moving vehicle over the same route on different days and start times.

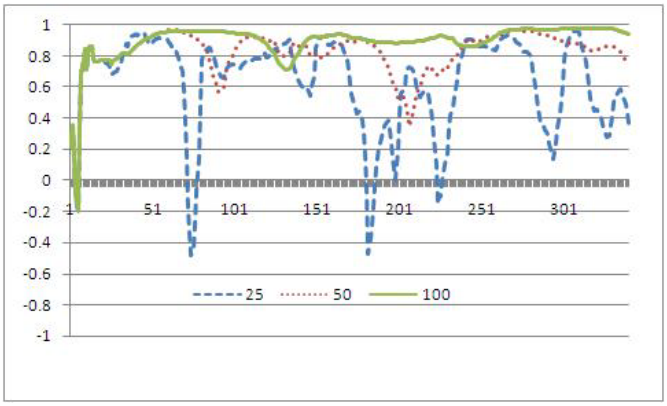


Figure 4. The effect of changing sliding window size on the correlation coefficient.

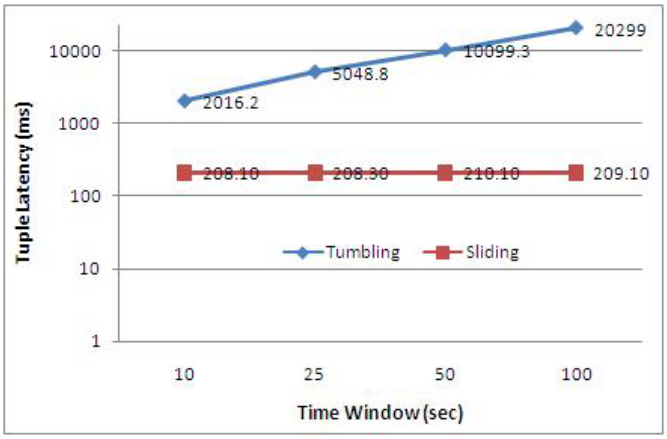


Figure 5. Performance comparison of sliding and tumbling windows at different window sizes.

recording. This also allows us to compare how a single bus moved at different times along the same route or two different buses moved with respect to each other - faster or slower- on the same path. Figure 3 shows the extracted longitude and latitude information from the two routes. These are the two values that are being separately correlated, respectively. Correlation coefficient is our statistical tool for finding event correlations (*e,g,* group asset tracking or route enforcement). To test its sensitivity, we investigate how changing the sliding window size affects the coefficient value and the processing performance (average latency of getting the results). Ultimately, we don't want to depend on any magic parameters.

Figure 4 shows the effect of window size on the correlation coefficient. Smaller window sizes have less data to compare, therefore when buses move differently with respect to each other the correlation value can drop sharply for that period. For larger window sizes like 50-100 this effect is compensated for, as one bus usually catches up with the other (or the same bus compensates for its transient delay over the same route at different times). To reduce the amount of output produced, we could use tumbling windows which only publish results at the end of a time or count period. A tumbling window is basically a discrete version of the continuous sliding window. Table II shows the correlation results for the tumbling windows. The results are similar to their corresponding sliding windows (on average higher for the larger windows), but they are published less frequently. While tumbling compensates for some of the jitter, the cost to calculate results increases as the window size increases as shown in Figure 5. Our future work includes running these queries over our high-end IBM Blade HS22 servers and testing performance over multi-core and distributed resources.

Figure 5 shows that changing the sliding window size does not affect the output tuple (correlation result) latency. This is because the each component of the correlation operation can be done incrementally. For example, we can maintain a running average of the scalar values by using the formulas:

TotalValNew = TotalValOld – ValueOut + ValueIn;

NewAve = TotalValNew/WindowSize;

For tumbling windows the delay increases logarithmically as the window size also increases logarithmically (shows linear on log-log scale), because the data is collected and processed at once for the time interval at the end of that time period.

V. Conclusions and Future Work

We focused on statistical correlation as a promising type of event detection and volume reduction technique over sensor streams in this paper. We used one type of sensor data and investigated one type of related application, which is real-time traffic and transportation management. We would like to extend our experiments with different types of FWSN data when we obtain them as finding the optimal parameters can be application dependent. Our analysis in this paper highlights some of these issues. Our future work will include automated methods for the correcting streaming data using regressions and adjusting the window parameters for correlations. Successful implementation of the proposed CEP system will satisfy the real-time or near real-time data analysis requirements of the mission-critical data stream applications. While we were only concerned with wireless sensor data in this paper, the techniques discussed can be used with wired sensor data (*e.g.* border protection with sensor-equipped fences) and software-based (non-sensor) monitoring in financial, e-trade, computer network applications as well.

TABLE II
Calculating Correlations Using Tumbling Windows

	Window Size		
Time	25	50	100
25	0.7725		
50	0.9293	0.9101	
75	0.1714		
100	0.7335	0.7469	0.9528
125	0.8016		
150	0.5505	0.7989	
175	0.5594		
200	0.1531	0.6363	0.8842
225	0.2692		
250	0.8886	0.8978	
275	0.8927		
300	0.3465	0.9086	0.9726
325	0.4321		
Average	0.577	0.8164	0.9365

Acknowledgment

We would like to thank Istanbul Bus Transport Agency (*abbreviated as* IETT) for providing us with the sample bus geo-tracking traces used in this paper.

References

[1] D. Abadi, D. Carney, U. Cetintemel, M. Cherniack, C. Convey, S. Lee, M. Stonebraker, N. Tatbul, and S. Zdonik. Aurora: A new model and architecture for data stream management. VLDB Journal, 12(2):120–139, August 2003.

[2] D. Abadi, Y Ahmad, M Balazinska, U Çetintemel, et al The design of the Borealis stream processing engine, In CIDR 2005

[3] E. Albek, E. Bax, G. Billock, KM Chandy, I. Swett, An Event Processing Language (EPL) for Building Sense and Respond Applications, In IEEE IPDPS, April 2005, Page136.

[4] A. Arasu, S Babu, J Widom, The CQL continuous query language: semantic foundations and query execution, VLDB Journal 15(2): 121-142, 2006

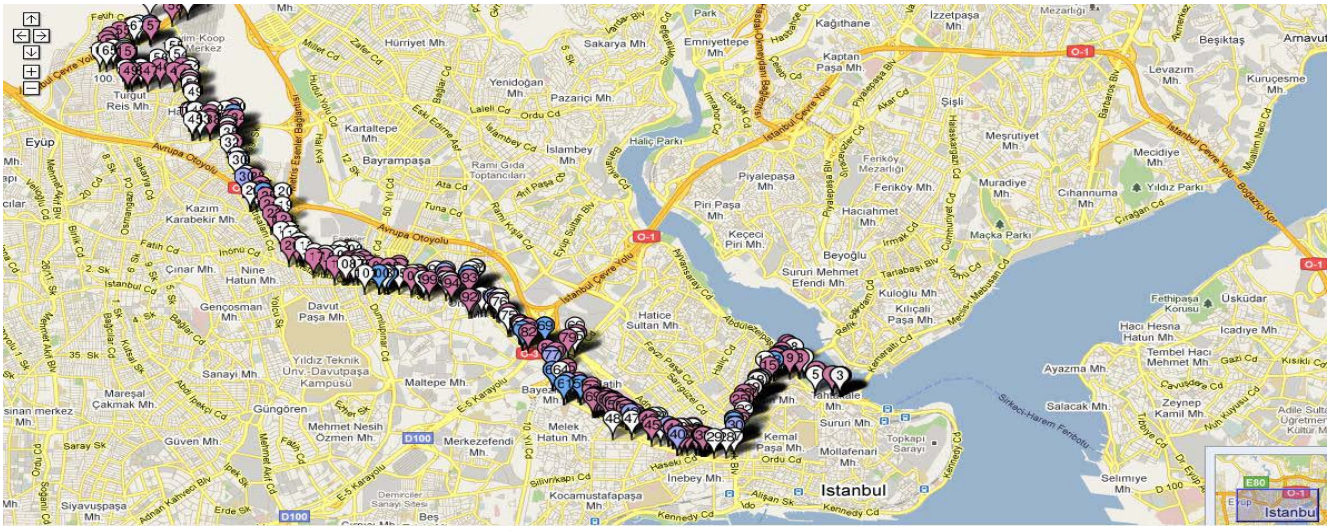
[5] K. Akkaya, I. Ari, In-network Data Aggregation in Wireless Sensor Networks, The Handbook of Computer Networks, Volume 2, Part 3, John-Wiley & Sons

[6] B. Babcock, S. Babu, M. Datar, R. Motwani, J. Widom, Models and issues in data stream systems, ACM PODS, 2002, June, pp 1-16.

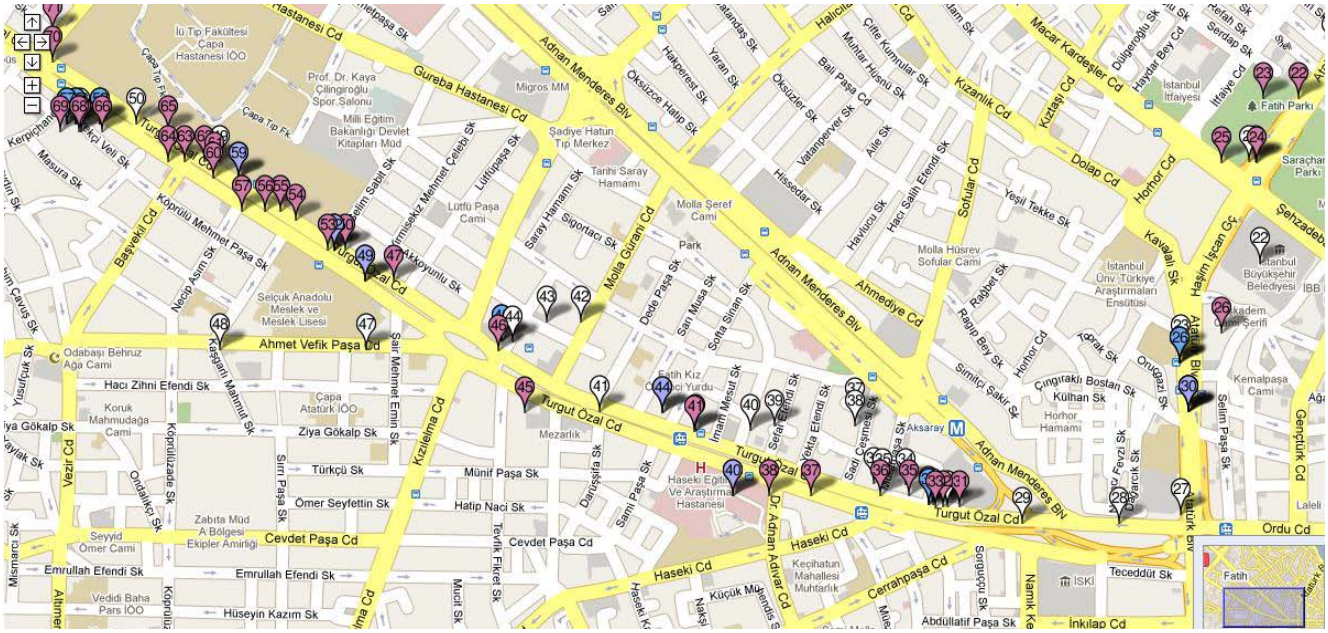
[7] A. Bulut, A. K. Singh,SWAT: Hierarchical stream summarization in large Networks, In ICDE 2003.

[8] Y. Chen, G. Dong, J. Han, BW Wah, J. Wang, Multi-Dimensional Regression Analysis of Time-Series Data Streams, In VLDB 2002.

[9] S Chandrasekaran, O. Cooper, A Deshpande, MJ Franklin, TelegraphCQ: Continuous dataflow processing for an uncertain world, CIDR 2003.
[10] A. J.Demers, J. Gehrke, B. Panda, M. Riedewald, V. Sharma, W. M. White: Cayuga: A General Purpose Event Monitoring System. CIDR 2007: 412-422
[11] EsperTech Inc. Event Stream Intelligence, http://www.espertech.com/
[12] J. Gehrke, F. Korn, D. Srivatava, On computing correlated aggregates over continual data streams, In ACM SIGMOD conference, pp 13-24, 2001.
[13] L. Golab, M. T. Ozsu, Issues in data stream management, SIGMOD Record Vol 32.No 2, June 2003
[14] S. Guha, C. Kim, K. Shim, XWAVE: Optimal and Approximate Extended Wavelets for Streaming Data, Proceedings of the 30th VLDB Conference, Toronto, Canada, 2004
[15] D. Gyllstrom, E. Wu, H-J Chae, Y. Diao, P. Stahlberg, G. Anderson, SASE: Complex Event Processing Over Streams, CIDR 2007.
[16] C. Jin et al, Sliding window Top-K queries on uncertain streams, In VLDB 2008, August, 2008
[17] T. Li, Q. Li, S. Zhu and M. Ogihara, A survey on wavelet applications in data mining, SIGKDD Explor. Newsletter, 4(2), 49-68, 2002.
[18] M. Liu, E. Rundensteiner, K. Greenfield, C. Gupta, S. Wang, I. Ari, A. Mehta, E-Cube: Multi-Dimensional Event Sequence Analysis Using Hierarchical Pattern Query Sharing, ACM SIGMOD 2011.
[19] Y. Mei, S. Madden, Zstream: A cost-based query processor for adaptively detecting composite events, In SIGMOD 2009.
[20] R. Motwani, et al., Mining frequent sets in streams, In VLDB 2002.
[21] D. Patnaik, M. Marwah, R. Sharma, N. Ramakrishnan, Sustainable operation and management of data center chillers using temporal data mining, In SIGKDD,2009, 1305-1314
[22] A. Ünal, Y. Saygın, Ö. Ulusoy, Processing count queries over event streams at multiple time granularities, Information Sciences 176 (2006)
[23] M. Wei, M. Liu, M. Li, D. Golovnya, E. Rundensteiner, K. Claypool: Supporting a spectrum of out-of-order event processing technologies: from aggressive to conservative methodologies. In SIGMOD 2009: 1031-1034
[24] E. Wu, Y. Diao, S. Rizvi. High-performance complex event processing over streams, In SIGMOD 2006, pp. 407-418
[25] PMCC in Wikipedia, http://en.wikipedia.org/wiki/Pearson_product-moment_correlation_coefficient



(A) The complete route for the same bus on two consequtive days is tagged and correlated.



(B) Zooming into one of the subjections of the route. Satellite data is dispersed (accuracy is lost) when vehicles are around buildings.

Figure 6: Tracking vehicles over Google Maps and correlating sensor readings to automatically detect which vehicles and routes are related and how.