# Heterogeneous Stream Processing and Crowdsourcing for Urban Traffic Management

Francois Schnitzler[1], Alexander Artikis[2], Matthias Weidlich[3], Ioannis Boutsis[4], Thomas Liebig[5], Nico Piatkowski[5], Christian Bockermann[5], Katharina Morik[5], Vana Kalogeraki[4], Jakub Marecek[6], Avigdor Gal[1], Shie Mannor[1], Dermot Kinane[7], and Dimitrios Gunopulos[8]

[1] Technion - Israel Institute of Technology, Haifa, Israel
[2] Institute of Informatics & Telecommunications, NCSR Demokritos, Athens, Greece
[3] Imperial College London, United Kingdom
[4] Department Informatics, Athens University of Economics and Business, Greece
[5] Technical University Dortmund, Germany,
[6] IBM Research, Dublin, Ireland
[7] Dublin City Council, Ireland
[8] Department of Informatics and Telecommunications, University of Athens, Greece

**Abstract.** We give an overview of an intelligent urban traffic management system. Complex events related to congestions are detected from heterogeneous sources involving fixed sensors mounted on intersections and mobile sensors mounted on public transport vehicles. To deal with data veracity, sensor disagreements are resolved by crowdsourcing. To deal with data sparsity, a traffic model offers information in areas with low sensor coverage. We apply the system to a real-world use-case.

**Keywords:** smart cities, crowdsourcing, event pattern matching, traffic, stream processing, big data

## 1 Introduction

New technologies related to mobile computing combined with sensing infrastructures distributed in a city or country are generating massive, heterogeneous data and creating opportunities for innovative applications. Levering such data to obtain a detailed and real-time picture of traffic, water or power networks, to name a few, is a key challenge to achieve better management and planning.

In this context, the goal of the INSIGHT project[9] is to support city or country managers in the detection of interesting events. The present work, originally presented in [3], gives a high-level overview of a traffic monitoring application in Dublin City, Ireland. Two particularly interesting features of this work for the machine learning and data mining communities are as follows.

– We present the general framework of an advanced smart city monitoring system leveraging large scale and heterogenous streams of sensor measurements, and the challenges that come up from a real application.

[9] `www.insight-ict.eu/`

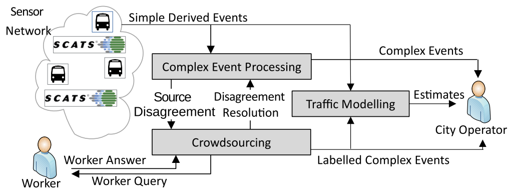


**Fig. 1.** Architecture overview

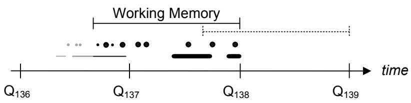


**Fig. 2.** RTEC event recognition

- We used real data streams coming from the buses and vehicle count SCATS sensors of Dublin city that we made publicly available[10]. The bus dataset includes 942 buses. Operating buses emit every 20-30 seconds. The SCATS dataset includes 966 sensors transmitting information every few minutes. They were collected during January 2013 and totalize 13GB of data.

The system architecture is schematized in Fig. 1. **Inputs** consist in the aforementioned sensors. Additional inputs can be requested from volunteering citizens through a **crowdsourcing** component (Sec. 4). The system **outputs**, in real time, a set of **complex events** (CEs) (Sec. 3), and **congestion estimates** for every intersection (Sec. 5). The architecture is implemented as a streaming system, using the **Streams framework** (Sec. 2).

## 2 Stream Processing

The *Streams* framework [4] is the backbone of our system. It provides a XML-based language to describe data flow graphs that work on sequences of data items (key-value pairs, i.e. attributes and their values). Nodes of the data flow graph are processes that comprise a sequence of processors. Processes take a stream or a queue as input and processors apply a function to the data items in a stream. These concepts are implemented in Java. Adding customized processors is realized by implementing the appropriate interfaces of the Streams API.

## 3 Complex Event Processing

For complex event processing, we use the Event Calculus for Run-Time reasoning (RTEC) [1, 2], a Prolog-based engine. Event Calculus is a logic programming language to represent and reason about events and their effects.

In RTEC, event types are represented as n-ary predicates of the form *event(Attribute1,...,AttributeN)*. The occurrence of an event E at time T is modeled by the predicate $\mathsf{happensAt}(E, T)$. The effects of events are expressed by means of *fluents*, i.e. properties that may have different values at different points in time, for example $\mathsf{holdsAt}(F = V, T)$.

In collaboration with domain experts, CEs have been defined over the input streams. For example, an intersection is congested if at least $n$ ($n > 1$) of its

[10] www.dublinked.ie

SCATS sensors are congested, or if busses suffer a high delay. CEs are modeled as logical rules defining event instances, for example,

$$\begin{aligned}\mathsf{happensAt}(\mathit{delayIncrease}(\mathit{Bus}),\ T) \leftarrow\ & \mathsf{happensAt}(\mathit{move}(\mathit{Bus}, \mathit{Delay}'),\ T'),\\ & \mathsf{happensAt}(\mathit{move}(\mathit{Bus}, \mathit{Delay}),\ T),\\ & \mathit{Delay}-\mathit{Delay}' > d,\quad 0 < T-T' < t.\end{aligned}$$

A $\mathit{delayIncrease}(\mathit{Bus})$ CE is recognized when the delay value of a $\mathit{Bus}$ increases by more than $d$ seconds in less than $t$ seconds.

At query times $Q_i$, RTEC recognizes CEs within a specified 'working memory' (WM) interval, based on data items received during the WM. Overlapping WMs allow to process, at $Q_i$, data items generated in $[Q_i - WM, Q_{i-1}]$ but arrived after $Q_{i-1}$. This is illustrated in Fig. 2. We performed both 'static' recognition, taking into consideration all sources, and 'self-adaptive recognition', where noisy sources are detected at run-time and temporarily discarded.

## 4 Crowdsourcing

We use crowdsourcing to ameliorate the veracity problem of the data. When the bus and SCATS sensors disagree about a congestion, the CE processing component requests additional inputs from the crowdsourcing component that queries human volunteers, or 'workers', close to the location of the disagreement.

Workers are presented with a set of possible labels (such as 'no congestion' or 'traffic jam') and select one. A key problem is to estimate the reliability of each worker, which we model by $p_i$, the probability that worker $i$ provides a wrong label. Estimating $\Theta \equiv \{p_i\}_i$ is typically done in batch mode, for example using the Expectation-Maximization (EM) algorithm. In order to estimate $\Theta$ on streaming data, we use an online EM based on stochastic approximation.

We employ the MapReduce programming model to communicate queries to the workers without effort from the user to reach him and to achieve real-time and reliable communication [5]. MapReduce allows processing parallelizable tasks across distributed nodes by decomposing the computational task into two steps, namely map and reduce. In our system, the crowdsourcing query engine communicates the queries to the workers (map task), and aggregates the results (reduce task). The interface of the mobile application is illustrated in Fig. 3.

## 5 Traffic Modeling

Large parts of the city are not covered by the sensors available. A Gaussian Process regression provides operators with a picture on the entire city [6, 7] .

To each vertex $v_i$ in the traffic graph $\mathcal{G}$ corresponds a latent variable $f_i$, the true traffic flow at junction $v_i$. We assume that any finite set $\mathbf{f} = f_j$ has a multivariate Gaussian distribution $P(\mathbf{f}) = \mathcal{N}(0, \hat{K})$. $\hat{K} = \left[\beta(L + I/\alpha^2)\right]^{-1}$ is the regularized Laplacian kernel function, with hyperparameters $\alpha$ and $\beta$. Zero

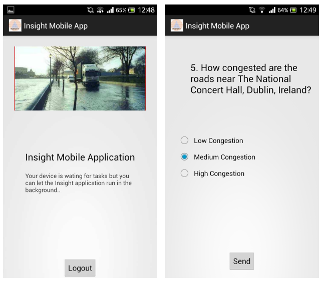


**Fig. 3.** Interface of the mobile crowd-sourcing application


**Fig. 4.** Traffic Flow estimates. Green dots indicate low traffic, red dots congestions.

mean is assumed without loss of generality. $L = D - A$ is the Laplacian, $A$ the adjacency matrix of $\mathcal{G}$, and $D$ a diagonal matrix with entries $d_{i,i} = \sum_j A_{i,j}$.

We also assume observations are affected by Gaussian noise: $y_i = f_i + \epsilon_i$, $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$. A joint distribution over observed and unobserved traffic flows can be defined, and the distribution of the unobserved flows conditionally on the observed ones computed. Results visible to operators are illustrated in Fig. 4.

**Acknowledgments.** This work is funded by the following projects: EU FP7 INSIGHT (318225); ERC IDEAS NGHCS; the Deutsche Forschungsgemeinschaft within the CRC SFB 876 "Providing Information by Resource-Constrained Data Analysis", A1 and C1.

## References

1. Artikis, A., Sergot, M., Paliouras, G.: Run-time composite event recognition. In: DEBS. pp. 69–80. ACM (2012)
2. Artikis, A., Weidlich, M., Gal, A., Kalogeraki, V., Gunopulos, D.: Self-adaptive event recognition for intelligent transport management. In: Big Data. pp. 319–325. IEEE (2013)
3. Artikis, A., Weidlich, M., Schnitzler, F., Boutsis, I., Liebig, T., Piatkowski, N., Bockermann, C., Morik, K., Kalogeraki, V., Marecek, J., Gal, A., Mannor, S., Gunopulos, D., Kinane, D.: Heterogeneous stream processing and crowdsourcing for urban traffic management. In: EDBT. pp. 712–723 (2014)
4. Bockermann, C., Blom, H.: The streams framework. Tech. Rep. 5, TU Dortmund University (12 2012)
5. Kakantousis, T., Boutsis, I., Kalogeraki, V., Gunopulos, D., Gasparis, G., Dou, A.: Misco: A system for data analysis applications on networks of smartphones using mapreduce. In: MDM12. pp. 356–359 (2012)
6. Liebig, T., Xu, Z., May, M., Wrobel, S.: Pedestrian quantity estimation with trajectory patterns. In: Flach, P.A., Bie, T.D., Cristianini, N. (eds.) Machine Learning and Knowledge Discovery in Databases, vol. LNCS 7524, pp. 629–643. Springer (2012)
7. Schnitzler, F., Liebig, T., Mannor, S., Morik, K.: Combining a gauss-markov model and gaussian process for traffic prediction in dublin city center. In: EDBT/ICDT Workshops. pp. 373–374 (2014)