Data Management Issues for Intelligent Transportation Systems (Position Paper) *

Federica Mandreoli[1,3], Riccardo Martoglia[1], Wilma Penzo[2,3], and Simona Sassatelli[1]

[1] DII - University of Modena e Reggio Emilia, Italy
{federica.mandreoli, riccardo.martoglia, simona.sassatelli}@unimo.it
[2] DEIS - University of Bologna, Italy
{wilma.penzo}@unibo.it
[3] IEIIT-BO/CNR, Bologna, Italy

Abstract. In this paper we discuss the technical challenges of devising a Data Stream Management System (DSMS) in the intelligent transportation scenario considered in the PEGASUS project, where the final aim is to provide reliable and timely information to improve the safety and the efficiency of vehicles' and goods' flows.
The system should collect and integrate the large amounts of geo-located stream items coming from On Board Units (OBUs) installed on vehicles, with the aim of producing real-time maps including traffic and Points Of Interest (POIs) information to be then distributed to OBUs. OBUs' smart navigation engines will exploit these maps to enhance mobility and provide user-targeted information.
We propose a two-tiered GIS DSMS architecture where stream items are pulled from the source input stream, processed and stored in a result container to be further pulled by other operators. The system reduces the data acquisition costs by adopting communication-saving policies, supports ad-hoc strategies for reducing the storage management costs (lowering response times and memory consumption), and provides the required data access functionalities through an SQL-like query language enhanced with stream, event, spatial and temporal operators. OBU stream items are also exploited to detect Events Of Interest (EOIs) such as jams and accidents and to support a collaborative mechanism for user-powered POI management and rating. EOIs and POIs are modeled through specific ontologies which allow for a flexible and extensible data management and guarantee data independence from the raw streams.

1 Introduction

Under the all too ordinary state of traffic congestion in our cities, the increase of fuel consumption and pollution (noise and emissions) due to stop-and-go conditions, the increase of accident rates and, in turn, the generation of congestion

* This work is partially supported by the Industria 2015 funded PEGASUS project.

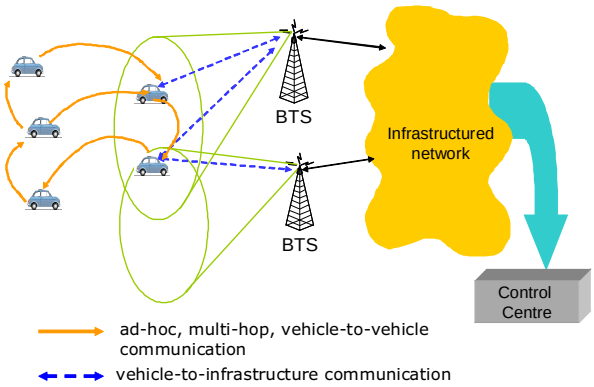

Fig. 1. The PEGASUS project reference scenario

because of accidents, are well-known problems which commonly deteriorate people's quality of life. The development of new transport and mobility concepts have indeed been promoted in the EU 7th Framework Programme with the aim of developing innovative and effective initiatives, bringing together all elements of a clean, energy-efficient, safe and intelligent transport. Thus, the support to sustainable urban mobility of people and goods in a territory has become one of the major challenges which has recently gained much interest in several ICT research areas, such as GIS [22], networking [8], operations research [19], wireless sensor networks [25], and telecommunications [16].

In this context, the PEGASUS[4] project aims at employing infotelematics systems to provide mobility solutions for an efficient and effective traffic management. The reference scenario is shown in Figure 1. Vehicles are equipped with sensor-based devices called On-Board Units (OBUs) which send stream items retrieved from sensors (e.g. average speed, sudden deceleration, etc.) to a data Control Centre. Data communication is performed in two steps: 1) vehicles are dynamically self-organized in a clustered V2V (Vehicle to Vehicle) -based communication system, where possibly aggregation of stream items is carried out, and 2) cluster heads broadcast the collected data towards base transceiver stations (BTSs) which compose an infrastructured network connected to the Control Centre (V2I - Vehicle to Infrastructure - communication). The Control Centre collects, integrates, and analyzes the large amounts of geo-located stream items coming from the OBUs, manages Events Of Interest (EOIs) (e.g. crashes, traffic jams) and produces real-time maps including traffic and Points Of Interest (POIs) information (e.g., gas stations, parking lots, cinemas, restaurants, aso) which is distributed to the OBUs according to user's location and personal profile. The OBUs' smart navigation engine exploits this information for providing end users with various services to enhance mobility: traffic congestion

[4] PEGASUS: Mobility management project through infotelematics systems for urban areas, passengers vehicles and goods safety (`http://pegasus.octotelematics.com/`).

prevention and warnings, alternative route prompting, crash monitoring, road weather conditions detection, parking availability, gas station cheapness, aso. The major objective of the PEGASUS project is thus to provide an Intelligent Transportation System (ITS) which provides reliable and timely information to improve the safety and the efficiency of vehicles' and goods' flows, as well as to make transportation a smart experience.

One of the major challenges in PEGASUS is the management of the multitude of stream items which originates from vehicles. The Control Centre is in charge of storing real-time geo-located stream items which must be accessed and manipulated efficiently to promptly answer users' requests. Scalability, modularity, and geographic data management capabilities, are thus essential requirements which characterize the Control Centre.

In this paper we discuss the technical challenges of devising a Data Stream Management System (DSMS) in the intelligent transportation scenario considered in the PEGASUS project. Several issues need to be faced. First of all, the use of scalable storage mechanisms which separate data of present interest (e.g. accidents, jams, existing at the present time) from historical data to be used for statistics, personalization, as well as for traffic predictions. As to this point, a crucial concern arises over the way data should be organized, both logically and physically, in order to make the use of the database the most efficient as possible. Furthermore, data acquisition of huge volumes of stream items is a heavyweight operation which could/should benefit of communication-saving techniques such as aggregation of stream items coming from the same geographic area. Finally, in order to allow for a flexible and extensible management of points of interests and events, the system should abstract from the specific characteristics these may have in different scenarios. For this purpose, we propose an ontology level where POIs and EOIs are conceptualized and made independent from the raw stream items coming from the OBUs and from the specific manipulation procedures of the raw data.

We propose a two-tiered GIS DSMS architecture satisfying the above requirements. The paper is organized as follows. Section 2 presents the architecture of the Control Center, including a short description of the POIs and EOIs ontologies. The challenging issues addressing efficient data acquisition are described in details in Section 3, whereas data storage is deeply dealt with in Section 4. Finally, Section 5 compares with related work and concludes.

2 Control Centre Architecture

In the PEGASUS project, the Control Centre is in charge of managing and exploiting stream items coming from vehicles for the purpose of delivering infomobility services (smart navigation, urban mobility, safety) to users. The Control Centre is thus composed of two main modules: the Data Stream Management System and the Service Module, as shown in Figure 2. The Service Module interacts with the DSMS to obtain information which is used for answering users' service requests through the Communication Manager; the latter takes advantage

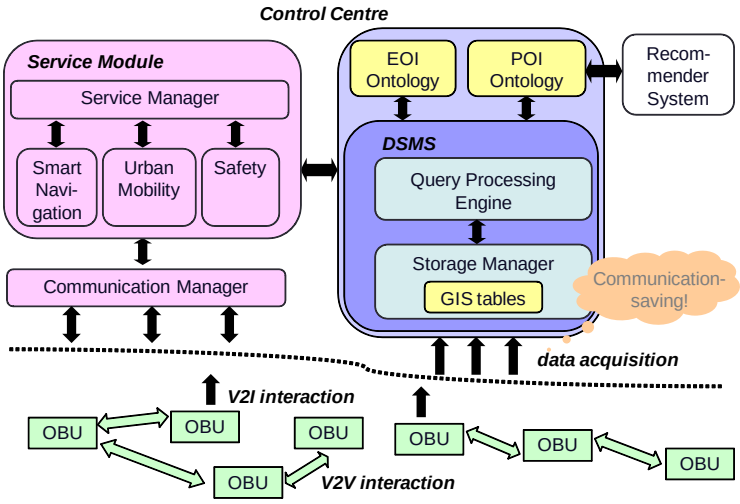

Fig. 2. PEGASUS Control Centre architecture

of efficient and effective delivering policies, for instance by providing information only to vehicles in a given region where an event has occurred.

In this paper we mainly focus on the DSMS which is responsible for stream item acquisition, storage, and manipulation. In order to satisfy the scalability requirements needed by the PEGASUS scenario, the DSMS should employ communication-saving policies for data acquisition, as well as flexible mechanisms for data storage to differentiate between fresh data which should be timely available, thus needing main memory allocation, and historical data which can be stored on disks and cached when needed.

The DSMS makes use of ontologies for the management of POIs and EOIs. The decoupling of the DSMS from the ontology level strengthens the independence of the system from "wired" implementation solutions tailored for specific scenarios. The ontology level allows for a flexible and extensible management of points of interest and events since it introduces an abstract level where they are conceptualized and made independent from the raw stream items coming from OBUs and from the specific manipulation procedures of the raw data. Ontologies are indeed well suited for modeling continuously evolving entities like POIs and EOIs since, in any moment, new concepts can be created and populated with new data instances.

The POI ontology deals with historical data and it is populated by means of the stream items coming from OBUs and which describe users' behavior, e.g., registering the restaurants where users stopped in the past. More precisely, firstly the ontology is initialized with a number of concepts (e.g. restaurant, car-park, etc...) imported from the available maps. Then, concepts are populated (and possibly new concepts are created) on the basis of OBUs' history. The final goal is to support a collaborative recommender system [2], which also exploits user profiling techniques (e.g. [20]), for POI management and rating in such way that

it is possible to equip the real-time maps distributed to OBUs with user-targeted POI information.

The EOI ontology models a variety of events of interest (e.g., accidents, traffic jams) which are characterized by the occurrence of specific traffic conditions. Events are detected by exploiting data collected by OBUs, i.e. the trend of specifically measured quantities like vehicles speed and acceleration, and other information sources like simulation models and traffic maps. Relying on such information, events can be discovered by applying commonly used techniques like Dynamic Probabilistic Models (DPM) and, in particular, Hidden Markov Models (HMM) (as in [21]). Specifically, the ontology describes the events and the rules regulating their generation. An event is thus detected when data collected by OBUs meets the conditions specified in the ontology and, as a consequence, it triggers the execution, on the GIS DSMS data tables, of event-based queries associated with it. These queries monitor at high rate the event area until the event is resolved with the aim of providing the users with infomobility services. Similarly to event start, event resolution is detected according to the corresponding conditions specified in the ontology and it causes the stopping of all the associated event-based queries.

A more detailed description of the DSMS and of the data acquisition process is given in the following sections.

3 Communication-saving Data Acquisition

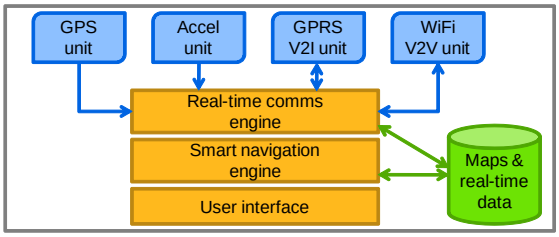

Fig. 3. On-Board Unit architecture

In our envisioned system, each of the participating vehicles is equipped with an OBU device, which is responsible for collecting, by means of V2V interaction, the data that has to be sent to the Control Centre (V2I interaction). The OBU is a small and "intelligent" mobile computing device (see Figure 3) which acquires data through the GPS and Accelerometer units, and performs real-time communications through the GPRS (for V2I) and WiFi (for V2V) units; all I/O units interact with the *real-time communication engine*, which orchestrates all the data acquisition and communication operations and on which this section will be focused. The device has also a flash-memory unit where static road network maps are stored, together with the real-time information acquired from the Control Centre and/or from neighboring vehicles; such storage is accessed by the smart navigation engine in order to provide the driver with the required services. The data which is acquired and managed by an OBU is:

– position, velocity, travel time and other GPS-acquired data (useful for real time traffic services);
– accelerations read from the accelerometer (used for assisting emergency detection e.g. in case of an accident);
– POI/EOI notifications and POI ratings as notified by the user.

OBUs update the Control Centre in real time with timely POI/EOI (or accident) notifications and with continuous streams of GPS-derived data. Since GPRS communication costs are still significantly high, each OBU follows an innovative hybrid communication strategy, where V2V communications between vehicles do not try to completely replace V2I (as in most of the available literature [10, 11, 15], still an infeasible scenario for our country), but are instead exploited to reduce the V2I payload. Several V2I and V2V communication-saving techniques are dynamically selected and combined on the basis of the specific Control Centre requirements and conditions, allowing the system to exploit the best of both worlds in order to minimize the global costs and maximize the usefulness and timeliness of the provided services.

In-node V2I communication-saving techniques. While POI/EOI data can be simply sent following the driver requests without specific in-node elaborations, several server update policies are available for transmitting GPS-derived data. Sampling-based policies allow the Control Centre to keep track of each vehicle's position over time, for instance for insurance and emergency services: besides *simple sampling*, which sends data at regular time/travelled distance intervals (e.g. each minute or each 2Km), *map-based sampling* allows for smarter communication choices on the basis of the vehicle position on the map (e.g. frequent street-level updates in navigating an urban area, higher rates in regular highway cruising). Information-need policies, on the other hand, prove valuable for minimizing communications in real time traffic informations services, where one of the main Control Centre goals is typically to maintain a sufficiently precise estimation of the current average travel time for the different road segments (highways to streets). In the *deterministic information-need* policy, each OBU dynamically receives such broadcasted velocity v_b from the Control Centre for each segment of interest and, for each travelled segment, it transmits the measured v_m if and only if $|v_b - v_m|$ exceeds a given threshold T [11]. Regardless of this rule, in the *probabilistic information-need policy*, each vehicle transmits the information with a given probability p, which can be updated from the Control Centre in order to guarantee a given confidence in the average speed computation [6]. We envision that such policies will coexist and be made available through a dynamic selection. Finally, for V2I interaction, we also plan to adapt some promising techniques which are commonly exploited in wireless sensor networks, such as *packet merging* [24] (sending one larger packet is less expensive than sending multiple smaller packets) and *linear regression* [13] (exploiting the possibly significant amount of redundancy in readings from a vehicle over time).

In-network V2V communication-saving techniques. In our vision, the vehicles exploit the "free" WiFi communication channel, if available, to organize themselves in clusters and to aggregate their data, so to minimize V2I communications. This *self-organization* is essential in a very dynamic, mobile and

faulty wireless environment, where data loss and collisions are very common and the communication system could easily clog up in so-called "broadcast storms". Similarly to [10], neighboring OBUs traveling a segment are dynamically self-organized into a cluster, thus forming a clustering-based multi-channel V2V communication system. Cluster Members (CMs) communicate to Cluster Heads (CHs) inside a cluster, and, in our case, CHs communicate the results to the Control Centre in V2I mode (optionally also performing inter-cluster communication for further communication minimization). Intra-cluster WiFi communications are almost immediate and allow OBUs to provide a fast reaction in case of emergency (for instance, an EOI accident notification could be instantaneously broadcasted to the CMs, allowing for a fast reaction, and CH could send a single aggregated EOI notification to the Control Centre). Moreover, for traffic monitoring purposes, the execution of *dynamic distributed aggregation* protocols allows for the execution of aggregation functions to estimate useful measures inside a cluster: such protocols, like dynamic counting (e.g. for the number of vehicles) and distributed averaging (e.g. for mean velocity) [15], are essential in our scenario, since, differently from most aggregation solutions, they neither assume sufficient connectivity to establish a routing infrastructure, nor they assume that the potentially frequent host failures are visible.

4 Data Storage

From a data management point of view, the application scenario considered in the PEGASUS project can be viewed as a data-intensive streaming application with temporal and spatial requirements. To this end, we envision a temporal GIS DSMS equipped with an SQL-like query language for time-geo-located data streams acquisition and access.

One of the main data management constraints of the PEGASUS project is that stream items must be retained beyond their real-time processing as the Control Centre should be able to process not only continuous queries but also any ad-hoc query and, in case, OLAP analysis which could be issued to the system, also after data acquisition. Therefore, we cannot adopt the "standard" DSMS data management solution where data storing is usually tightly coupled with query processing [1, 4, 9]: records are acquired only as needed to satisfy the queries and stored only for a short period of time or delivered directly out of the network, unless the query flow explicitly requires persistent storage. On the contrary, the GIS DSMS we propose founds on a two-tiered architecture where stream items are pulled from the source input stream, processed and stored in a result container to be further pulled for query processing. For ease of presentation, we model this situation through two operators (Producer and Consumer) and a store between them [7].

Such a store contains the table `obu` which logically has one row per OBU report, with one column per attribute that OBUs can produce:

```
obu(id,time,position,velocity,acceleration,travel_time...,
    poi,poi_type,...,eoi,...)
```

Q1:	**Q2:** ON EVENT crash(eoi.position):
SELECT *	SELECT o.velocity
FROM observations	FROM observations AS o, eoi
SAMPLE PERIOD 10 s	WHERE distance(eoi.position,o.position)<100 m
	SAMPLE PERIOD 1 s
Q3:	**Q4:**
SELECT time, COUNT(*)	CREATE STORAGE POINT rest_hst AS
from obu	(SELECT o.id, o.time, p.id, o.rating
WHERE intersect(position,s)	FROM obu AS o, POI AS p
GROUP BY SCALE(time AS HOUR)	WHERE o.poi=p.id
SAMPLE PERIOD 1 h	AND o.type="restaurant")

Table 1. Query samples

Although we impose the same schema on the items produced by every OBU, some of the attribute values could be NULLs. For instance, NULLs are the default values for the POI and EOI attributes, unless drivers report useful information about points or events of interest by explicitly interacting with the OBU navigator. Moreover, the road network maps are stored together with two tables: `poi(id,position,type,...)` containing the points of interest and `eoi(id,position,type,...)` maintaing the current events of interest. It is worth noting that while `obu` is a "stream" relation containing a time-varying bag of tuples [4], all the other tables are conventional stored relations.

The Producer performs regular append-only insertions of new OBU reports in the `obu` table. Observations arrive at well-defined sample intervals specified through the query language clause `SAMPLE PERIOD`. For instance, the continuous query Q1 shown in Tab. 1 specifies that each OBU should report its observation (contained in the virtual table `observations`) once each 10 seconds. Q1 is a standing query which runs continually, unless it is explicitly stopped. Alternatively, queries can be limited to run for a specific time period via a `FOR` clause (e.g. `SAMPLE PERIOD 5 s FOR 1 h`).

Sometimes, it could be necessary to alter the default data acquisition processes, for instance when a particular event occurs. To this end, we support event-based queries. For instance, query Q2 of Tab. 1 could be used to monitor the velocity of vehicles near an accident. The events of interest (e.g. accident, queue, etc.) are defined in the EOI Ontology. When an event is detected, the corresponding tuple is added to the `eoi` table and all the queries associated to that event are triggered. Then, each of these queries is issued to the system for the event period, i.e. it terminates whenever the tuple is dropped from `eoi`. Generally speaking, event-based queries are essential as they allow the system to perform specific operations only when some external condition occurs. For instance, an accident notification service would be better supported by an event-based query on the `obu` table which identifies all the vehicles entering a segment near an accident. Indeed, a fast notification of the accident to those vehicles would allow them to exit the expressway and avoid the resulting congestion [5].

Any query referring to the `obu` table is a Consumer in that it uses the query results that are generated by the Producer. Continuous queries on the `obu` table

follow the usual stream-based semantics: they are time-based sliding window queries which incrementally produce new answers by considering only those items which have arrived in the last time period. For instance query Q3 delivers a stream of values representing the number of vehicles traveling on the segment `s` per hour. The `SAMPLE PERIOD` argument of query Q3 states both that the query delivers updates every hour and that at each run the tuples used to compute the aggregate record all belong to the time period between the current hour and one hour ago. Alternatively, it is possible to specify a different time period by using the `WINDOW` operator. For instance, appending the clause `WINDOW 2 h` to query Q3 means that it still delivers updates each hour but the reference time period for data items is no longer one hour but two hours.

Finally, similarly to many streaming systems [1, 4, 9], projections and/or transformations of tuples from the stream table may be stored in materialization points. Such materialization points accumulate a small buffer of data that may be used in other queries. For instance, query Q4 is a one-shot query which registers the restaurants where users stopped in the past. This kind of queries is exploited for populating the POI ontology, which is responsible for the maintenance of the history of the OBU's behavior w.r.t. the visited POIs.

Vehicle reports usually contain geo- and temporal-referenced data. Therefore, we are particularly interested in supporting those predicates which GISs commonly support for a spatial analysis of geographic data. For instance, Q2 contains the spatial predicate `distance` which returns the distance between the crash position and the vehicle position, both represented as points while Q3 uses `intersect` to select the vehicles to count. Moreover, `obu` is a time-varying relation. With reference to the temporal literature [14], the timestamp `time` of each tuple, which represents the time at which the position report was emitted, is a valid time value. As for the spatial case, the support to temporal querying constructs is essential for a time-aware access to data streams. For instance, the TSQL2 operator SCALE used in query Q3 moves the `time` value "up" to the hour granularity.

When a query is issued to our temporal GIS DSMS, the system is responsible for delivering the query results which consist in one single result set, in case of ad-hoc query, or in a stream of result sets, in case of continuous query. In line with the DSMS design principle argued in [7], we envision a system where the storage manager is fully decoupled from the Query Processor Engine (QPE). In this way, we can implement a storage mechanism which assists the query processing engine in query processing and optimization and offers support for reducing the storage management costs (lowering response times and memory consumption). At the same time, the QPE can abstract from the specific data access issues and exclusively focus on producing efficient query plans for the expressive query language briefly introduced above.

More precisely, the storage manager will provide the required data access functionalities and implement a scalable storage mechanism which efficiently supports the wide variety of storage requirements emerging from the intelligent transport application considered in the PEGASUS project. We briefly outline the

main ones. First, unlike standard DSMSs, data never expire, that is they never go out of the system. This means that the storage manager must be able to manage very large amounts of data. At the same time, some of the services we would like to support, such as the accident alert service, have real-time requirements. These services usually rely on continuous queries which can specify windows of different sizes on the same data. Therefore, the storage manager should ensure fast access to the most recently reported data values. Event-based queries need to store events for a given time period and to introduce synchronization mechanisms on the store contents. Finally, it is well known that processing non-conventional predicates, such as the spatial and temporal ones, is particularly time consuming. In this context, the storage manager should provide some index structures to speed up response time.

Similarly to other DSMSs [1, 4], the `obu` tuples required by the windowed operations will be mainly maintained in main-memory data structures, such as circular queue [1] or linked lists [7]. Each sliding window will be associated to two pointers bounding the list of tuples which haven't been processed yet. Whenever a tuple has been consumed by all the current sliding windows it becomes an "historical" item. Historical items will be maintained in secondary-memory data structures. In this context, instead of moving one historical item at a time, we will garner added efficiency by introducing lazy removal approaches which move multiple marked items from main-memory to secondary-memory. Finally, higher level index structures will be built on top for an efficient support to the different kinds of predicates.

Similarly to [7], the store manager will put at the QPE disposal an access interface consisting in per-item and per-attribute data access operations such as read and update. Such operations will be used to construct query plans. In doing this, the QPE will implement various query optimization approaches, from grouping similar continuous queries [9], to the exploitation of the update patterns of the current continuous queries [12] and operator scheduling [4].

5 Discussion and Concluding Remarks

Intelligent Transportation Systems present challenges which cover several research areas, such as GIS, wireless sensor networks, and telecomunications, just to mention a few.

Recent studies in the US show that the current range of WiFi technology (100-300m in outdoor scenarios) coupled with an OBU market penetration of 3%-10% are enough to guarantee an effective V2V communication [11]. In Italy, the number of past-generation (V2I-only communication) OBU-equipped vehicles is constantly increasing and has recently reached 750,000 installed units[5] (nearly 2% penetration). A gradual transition towards our innovative hybrid V2I/V2V communication strategy, appears not only feasible but also a promising scenario, where V2V would significantly cut-down the currently very high V2I communication costs. In particular, we plan to adapt those advanced communication-saving

[5] http://traffico.octotelematics.it

policies, such as dynamic aggregation and self-organization solutions, which have been successfully employed in different EU and US scenarios. For the first time, such policies will be combined and made available through a dynamic selection on the basis of the Control Center requirements and on an Italian ITS.

To the authors' knowledge however, no proposals exist which are explicitly devoted to the data management issues of ITSs. The only work which focuses on traffic information is [5], which however only specifies a benchmark for DSMS. The research areas more related to the PEGASUS application context are the ones of DSMS and event-driven query processing. In recent years, there has been an extensive amount of work in the field of stream processing. Several research prototypes for DSMS have been built (e.g. Aurora [1], STREAM [4], NiagaraCQ [9]), each one proposing its own data and query models and query processing solutions. Moreover, many other papers (e.g. [12]) delve into the problem of processing continuous queries. However, all of these approaches have a tight coupling between query processing and storage management by embedding storage management within the stream processing engine. This data management solution does not comply with our research context where data streams must be retained beyond their real-time processing. As to our knowledge, [7] is the only paper which discusses the option of a loosely-coupled system design. We adopted this solution and discussed in details the issues related to a data store containing stream items which never expire.

A large amount of literature is available regarding event detection and management in several research fields such as RFID [21, 23], sensor networks [18], DSMS [9], continual queries [17] and publish/subscribe systems [3]. As to the event-driven features of our system, we borrowed ideas from [18, 17]. Differently from these works, we consider events as high level concept which are formally defined in the event ontology and we model the relationship between such an ontology and the underlying DSMS.

Finally, the query language we sketched in Sec. 4, mainly through examples, draws inspiration from the TinyDB language [18] to which we added the `WINDOW` clause.

References

1. D. J. Abadi, D. Carney, U. Çetintemel, M. Cherniack, C. Convey, C. Erwin, E. F. Galvez, M. Hatoun, A. Maskey, A. Rasin, A. Singer, M. Stonebraker, N. Tatbul, Y. Xing, R. Yan, and S. B. Zdonik. Aurora: a new model and architecture for data stream management. *VLDB J.*, 12(2):120–139, 2003.
2. G. Adomavicius and A. Tuzhilin. Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. *IEEE TKDE*, 17(6):734–749, 2005.
3. M. Aguilera, R. Strom, D. Sturman, M. Astley, and T. Chandra. Matching Events in a Content-Based Subscription System. In *Proc. of PODC*, pages 53–61, 1999.
4. A. Arasu, B. Babcock, S. Babu, J. Cieslewicz, M. Datar, K. Ito, R. Motwani, U. Srivastava, and J. Widom. STREAM: The Stanford Data Stream Management System. In M. Garofalakis, J. Gehrke, and R. Rastogi, editors, *Data Stream Management: Processing High-Speed Data Streams*. Springer, 2007.

5. A. Arasu, M. Cherniack, E. F. Galvez, D. Maier, A. Maskey, E. Ryvkina, M. Stonebraker, and R. Tibbetts. Linear Road: A Stream Data Management Benchmark. In *Proc. of VLDB*, pages 480–491, 2004.
6. D. Ayala, J. Lin, O. Wolfson, N. Rishe, and M. Tanizaki. Communication Reduction for Floating Car Data-based Traffic Information Systems. In *Proc. of Advanced Geographic Information Systems, Applications and Services*, pages 44–51, 2010.
7. I. Botan, G. Alonso, P. Fischer, D. Kossmann, and N. Tatbul. Flexible and scalable storage management for data-intensive stream processing. In *Proc. of EDBT*, pages 934–945, 2009.
8. R. Bruno, M. Conti, and E. Gregori. Mesh networks: commodity multihop ad hoc networks. *IEEE Communications Magazine*, 43(3):123–131, 2005.
9. J. Chen, D. DeWitt, F. Tian, and Y. Wang. NiagaraCQ: A Scalable Continuous Query System for Internet Databases. In *Proc of SIGMOD*, pages 379–390, 2000.
10. R. Ding and Q. Zeng. A Clustering-based Multi-channel Vehicle-to-Vehicle (V2V) Communication System. In *Proc. of ICUFN*, pages 83–88, 2009.
11. S. Goel, T. Imielinski, and K. Ozbay. Ascertaining Viability of WiFi based Vehicle-to-Vehicle Network for Traffic Information Dissemination. In *Proc. of IEEE ITSC*, 2004.
12. L. Golab and M. Özsu. Update-Pattern-Aware Modeling and Processing of Continuous Queries. In *Proc. of SIGMOD*, pages 658–669, 2005.
13. C. Guestrin, P. Bodik, R. Thibaux, M. Paskin, and S. Madden. Distributed Regression: an Efficient Framework for Modeling Sensor Network Data. In *Proc. of IPSN*, pages 1–10, 2004.
14. C. Jensen and R. Snodgrass. Temporal data management. *IEEE TKDE*, 11(1):36–44, 1999.
15. O. Kennedy, C. Koch, and A. J. Demers. Dynamic Approaches to In-network Aggregation. In *Proc. of ICDE*, 2009.
16. U. Lee and M. Gerla. A survey of urban vehicular sensing platforms. *Computer Networks*, 54(4):527–544, 2010.
17. L. Liu, C. Pu, and W. Tang. Continual Queries for Internet Scale Event-Driven Information Delivery. *IEEE TKDE*, 11(4):610–628, 1999.
18. S. Madden, M. Franklin, J. Hellerstein, and W. Hong. TinyDB: An Acquisitional Query Processing System for Sensor Networks. *ACM TODS*, 30(1):122–173, 2005.
19. A. Melki, S. Hammadi, Y. Sallez, T. Berger, and C. Tahon. Advanced approach for the public transportation regulation system based on cybercars. *RAIRO-Oper. Res.*, 44(1):85–105, 2010.
20. S. Middleton, N. Shadbolt, and D. D. Roure. Ontological User Profiling in Recommender Systems. *ACM TOIS*, 22(1):54–88, 2004.
21. C. Ré, J. Letchner, M. Balazinska, and D. Suciu. Event queries on correlated probabilistic streams. In *Proc. of SIGMOD*, pages 715–728, 2008.
22. L. Speičvcys, C. Jensen, and A. Kligys. Computational data modeling for network-constrained moving objects. In *Proc. of GIS*, pages 118–125, 2003.
23. F. Wang, S. Liu, and P. Liu. Complex RFID event processing. *The VLDB Jour.*, 18(4):913–931, 2009.
24. Y. Yao and J. Gehrke. Query Processing in Sensor Networks. In *Proc. of CIDR*, 2003.
25. P. Zhuang, Q. Qi, and Y. Shang. Wireless sensor networks in intelligent transportation systems. *Wireless Communications and Mobile Computing*, 9(3):287–302, 2009.