A Semantic Approach for Building Pervasive Spaces

Daniel Massaguer Sharad Mehrotra Nalini Venkatasubramanian
Donald Bren School of Information and Computer Science
University of California, Irvine
{dmassagu,sharad,nalini}@ics.uci.edu

ABSTRACT

Large and pervasive sensing, communications, and computing infrastructures are enabling the realization of pervasive spaces. Enabling such spaces, however, encompasses a set of challenges. First, programming each application such that it connects to each sensor and it interprets the data being sensed requires a concentration of expertise that is rarely available. Second, achieving a wise and fair usage of the infrastructures is impossible with current approaches due to their lack of awareness of domain and application semantics. This paper summarizes a PhD dissertation that focuses on designing and implementing a middleware that addresses these challenges and overcomes the limitations of previous approaches by featuring a distributed streaming architecture and by being aware of the semantics of the space and applications. Namely, we focus on (i) the design and implementation of the overall system architecture and its underlying programming and execution model, (ii) a set of mechanisms to provide the right level of abstraction to applications, and (iii) a set of mechanisms that are able to protect privacy due to the inclusion of semantics in the middleware.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

Keywords

Pervasive spaces, middleware, SQL-programming, privacy

1. INTRODUCTION

Large and pervasive sensing, communications, and computing infrastructures are enabling the creation of pervasive spaces that offer new possibilities, conveniences, and functionalities. Instrumented pervasive spaces that allow observing people and other entities in a given physical space enable a rich set of applications ranging from surveillance and situational awareness to collaborative applications. Consider for instance, an office environment. Here, collaboration can be greatly enhanced if members of a team know where teammates are, what they are doing, and if they are available for discussions. Consider also the same office during an emergency such as a fire. Emergency response can benefit from knowing where the office occupants and firefighters are and what their health status is. These two types of applications are our main motivating applications for the work here presented.

Building these types of applications, nonetheless, carries a set of challenges including:
1.- Programming abstraction. Building these applications from scratch is an almost impossible task due to the large amount of heterogeneous sensors. Programming each application such that it connects to each sensor and it interprets the data being sensed requires a concentration of expertise that is rarely available. A middleware for pervasive spaces needs to provide a programming abstraction such that applications can be programmed independently from how events are detected.
2.- Privacy. Some of the entities being sensed are people. This makes privacy an issue. The responsability of keeping sensitive data from being inferred and misused cannot be trusted to the applications themselves—they are build to achieve a specific goal which does not necessarily involve preserving privacy—nor it can be trusted to the sensors—these know about bits and bytes but do not understand the concepts of people, sensitive data, and inference channels. The right place to deal with privacy is at the middleware level.

This dissertation focuses on a middleware that addresses the above challenges by embedding semantics in the middleware itself. Namely, we focus on (i) the design and implementation of such a middleware's overall system architecture and its underlying programming and execution model, (ii) a set of mechanisms to provide the right level of abstraction to applications, and (iii) a set of mechanisms that are able to protect privacy due to the inclusion of semantics in the middleware. Programming abstraction is provided by an extension to the entity-relationship model and an interface that allows users to pose SQL-like queries to the middleware regarding the objects in the physical space. In turn, this layer of abstraction along with further application semantics enable addressing the challenges of privacy. Privacy is protected with a utility-based framework that understands privacy as negative utility and maximizes the utility of the information being released with an algorithm based on distributed simulated annealing. All these mechanisms are embedded in a middleware that features a shared code repository for reusability and a distributed execution environment based on mobile agents for scalability and fault-tolerance.

The rest of the paper is organized as follows. Section 2 describes the overall architecture of the system and the concepts of virtual sensors and operators. Section 3 describes an extension to the entity-relationship diagram for designing pervasive space applications as well as the mechanisms

MDS'09, November 30, 2009 Urbana Champaign, Illinois, USA

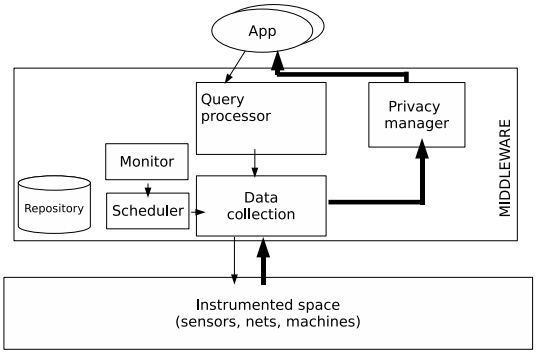

Figure 1: System architecture

to answer SQL-like queries. Section 4 summarizes our work on privacy-preservation. Section 5 describes both the system implementation and applications that have been implemented with our system. Section 6 summarizes the related work. Finally, Section 7 summarizes the paper.

2. ARCHITECTURE

Pervasive space applications are interested in semantically meaningful observations such as where people are and what they are doing. Raw sensor observations, however, do not always produce such domain-dependent observations but rather detect things like temperature, activity in a room, and location of a cellphone. Our middleware (SATware [2]) bridges the gap between the applications' interests and the raw sensor streams with the concept of *virtual sensors*. Virtual sensors are a set of transformations that when applied to a set of raw sensor streams produce a semantically meaningful stream at the level of abstraction at which applications reason. In order to answer a query from an application, SATware instantiates a set of virtual sensors, the output of which is forwarded to the application. Note that, beyond programming simplification, virtual sensors also allow for more flexibility: the middleware can decide to swap a virtual sensor if a better one becomes available without even notifying the application.

Fig. 1 depicts the overall system architecture. Applications pose continuous queries to the Query Processor. The Query Processor selects a set of virtual sensors that combined with relational operators (e.g., join and selection) will provide the answers to the queries. Each virtual sensor is in fact described as a graph of operators. The query plans formed by the virtual sensors and necessary relational operators are passed to the Data Collection module, which executes each operator (implemented as a mobile agent) in one of the machines from the underlying pervasive computing infrastructure. The streams outputted by the virtual sensors, prior to being forwarded to the application, pass through the privacy module, which modifies them to ensure that privacy is not violated. While data is being collected, the Monitor module forwards statistics which passes to the Scheduler module. The Scheduler module adapts the operator parameters in order to optimize resource usage. All modules consult a repository which contains a snapshot of the current infrastructure state, domain information, and code. The next two sections summarize the techniques behind the Query Processor and the Privacy Manager.

3. PROGRAMMING ABSTRACTION

From the point of view of the applications, a pervasive space is a physical space in which activities and objects are embedded. In this space, there are 3 types of objects: (1) spatial objects such as rooms, floors, and buildings, (2) people such as Mary, Peter, and Alice, and (3) inanimate objects such as coffee pots, recycle bins, and refrigerators. Each of these objects have attributes such as name, occupancy level, location, salary, level of coffee, and so on. These attributes are either static or dynamic (i.e., they change as a function of time). For instance, name and salary are static whereas location is static for spatial objects but dynamic for people. We call *observable attributes* the subset of attributes that can be sensed by the pervasives space. For example, a pervasive space with video-based people counters and RFID readers can detect both the level of occupancy of a room as well as recognize the people in it.

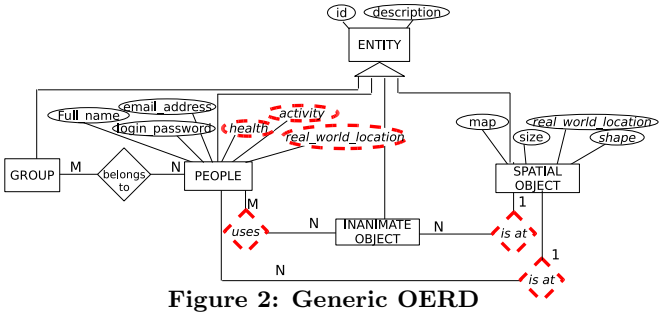

Figure 2: Generic OERD

3.1 Observable E-R

To model a pervasive space, we extend the Entity Relationship Diagram (the *de-facto* standard for designing databases) with new data and relationship types: observable attributes and observable relationships. We denote these new types with dashed circles and rombes and call the new diagram the Observable Entity Relationship Diagram (OERD). Fig. 2 depicts the OERD for a pervasive space. We call this diagram the *Generic OERD*. The Generic OERD describes a generic pervasive space. It contains the entities *People*, *Spatial Object*, *Inanimate Object*, and *Group*. Each entity has a set of basic attributes. For instance, *People* has static attributes that identify an individual (name, login, and password), and observable attributes that identify its state (location, activity, and health).

Whereas the Generic OERD represents a model of the space shared by any application, it does not include domain-specific details. For example, a data model for an application to improve collaboration between students needs to include a *Student* entity with a *StudentId* attribute, and a data model for an emergency response application needs to include entities such as *Firefighters* and *Oxygen Tanks*. If necessary, the Generic OERD can be extended for each application to add new roles that people take (e.g., students and firefighters), inanimate objects (e.g., backpacks and oxygen tanks), and spatial objects (e.g., floors and buildings). Fig. 3 depicts an example where it has been extended with the *Student* entity.

The OERD is translated to the relational model (i.e. tables) applying the same standard procedures that are used to translate an Entity Relationship Diagram to its relational model. Entities become tables with one column per attribute, N:M relationships become tables with one column for each entity taking part of the relation and attribute, and so on. The only difference is that when the tables are populated, only static attributes are inserted.

3.2 Continous Queries

Applications query SATware as they would query a database. Continuous queries are posed on the relational model following an SQL-like syntax extended with window operators as in the CQL stream query language [1]. SATware's

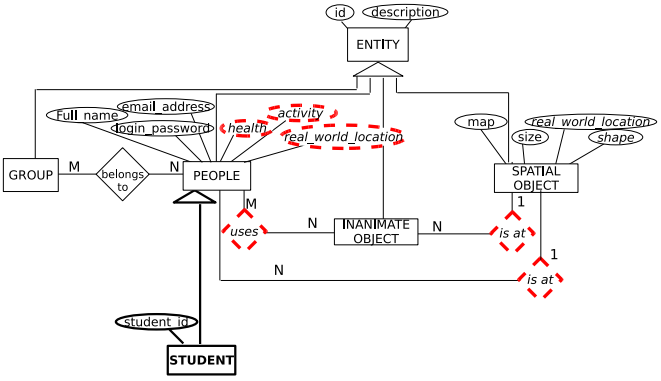

Figure 3: Generic OERD extended for a specific application

Query Processor translates each query to a query plan (i.e., a graph with selections, joins, and other operators). For each source table in the query plan, SATware selects virtual sensors to sense the observable attributes of each entry in the table.

Note that all virtual sensor's output streams adhere to one of 2 schemas: the observable attribute schema *(entityId, attributeId, attributeValue, time)* or the relationship schema *(relationshipId, entitySet, attributeId, attributeValue, time)*. This constraint, along with the existence of a generic OERD, provides SATware submodules with common grounds for semantic reasoning (e.g., see the privacy section below). This constraint also allows SATware to keep information in its repository regarding which virtual sensors can sense which attributes/relationships of which entities. With this information, and given a cost for each virtual sensor, the Query Processor is able to select a subset of virtual sensors that answer a continuous query at a minimum cost.

Once the virtual sensors are selected, the query plan along with the virtual sensor descriptions (recall that a virtual sensor is a graph of operators) are forwarded to the Data Collection module, which (i) decides in which machine each operator executes and instantiates them, (ii) decides to share operators if possible, and (iii) connects the last operator of the query plan to the application [1].

4. PRIVACY

The task of pervasive space applications is to provide answers to users' queries. While the utility of a query response is maximized for the observer when the data is in "the most precise" form, the utility may be quite the opposite for the target of the query. For instance, if location privacy is a concern then revealing accurate information about location is certainly detrimental for the target. There is often such a conflict between the "positive" and "negative" utilities associated with a piece of information that comprises a query response. Traditional access control mechanisms are geared towards deciding between the binary options of *granting* and *denying* access to a piece of information. In contrast, we consider a much larger set of options where the same information is revealed to the observer at a different granularity. Another important feature of our privacy analysis is that we factor in the information disclosed due to inference. In the remainder of this section we summarize how we model background knowledge, observer and target utilities, and the information release problem as a constrained optimization problem in a utility theoretic framework. We also summarize the algorithm we propose to solve the maximization problem as well as the implementation of our approach and some experimental results. For more details, the reader can check our paper in [36].

[1]Note that SATware is a distributed streaming system where query answers are forwarded directly to the application, without storing them into an intermediate database.

4.1 Background Knowledge Model

We model an observer's background knowledge (BK^{obs}) as a set of probabilistic first-order Datalog (pDatalog) clauses [28]. An example of a rule is:

$$Tuple(Alice, Location, l, t) : p * 0.8 \leftarrow Tuple(Mary, Location, l, t) : p \quad (1)$$

A data element is represented as a multi-attribute tuple of the form: $Tuple(objectId, attributeName, attributeValue, time) : certainty$. The rule above states that if Mary is at location l at time t with probability p, then Alice is also present at the same location with a probability of $0.8*p$. The knowledge base consists of rules of the above form along with some other auxiliary information in the form of hierarchies and facts.

4.2 Privacy vs Utility

We argue that the potential use (or misuse) of information is what defines the expected utility of information. Given this premise, we formulate our objective as follows. Let Y_{req} be the answer provided by the Data Collection module, Y_{rel} be the information being released by the Disclosure Control, $Y_{derived}$ the information inferrable from Y_{rel}, $\preceq$ the generalization relationship, and Y^i_{rel} a subset of Y_{rel} such that no piece of information in Y^i_{rel} allows one to infer a piece of information in $Y_{derived} - Y^i_{rel}$ and vice-versa. We cast the problem as a maximization problem where the objective is to find a generalization $Y^i_{rel} \preceq Y^i_{req}$ for each Y^i_{rel} that maximizes the observer aggregated expected utility of the information released while ensuring that the largest negative utility of all the information pieces the observer can infer, given his background knowledge and the information being released, is not greater than the largest positive utility of all the information pieces the observer can infer given the generalization hierarchy and the information being released. Formally it can be stated as:

$$\max_{Y^i_{rel}} EU_O(Y^i_{rel}) \quad (2)$$

such that

$$min\_EU_T(Y^i_{rel}) + max\_EU_O(Y^i_{rel}) \geq 0.0 \quad (3)$$

$$Y^i_{rel} \preceq Y^i_{req} \quad (4)$$

where

$$EU_O(Y^i_{rel}) = \sum_{\forall y_r \in GH(Y_{rel^i})} EU_O(y_r)$$

$$min\_EU_T(Y^i_{rel}) = \min_{y_d \in Y^i_{derived}} EU_T(y_d)$$

$$max\_EU_O(Y^i_{rel}) = \max_{y_r \in GH(Y^i_{rel})} EU_O(y_r) \quad (5)$$

and $EU_O(y)$ and $EU_T(y)$ are the expected observer and target utilities for releasing to the observer a piece of information regarding the target. These expected utilities are defined as the product of how much the observer believes y to be true, the probability of y being true, and the pre-specified utility of y being used/misused by the observer.

See Table 1 for a summary of the meaning of the symbols.

4.3 Solution

If an observer poses N continuous queries, it is possible that at any point in time, N distinct tuples might need to

Symbol	Description
Y_{req}	tuples before discl. control
Y_{rel}	tuples after discl. control
$Y_{derived}$	info inferrable from Y_{rel}
$GH(Y_{rel})$	info inferrable from y_{rel} based on gen. hierarchies
$\prec, \preceq$	generalization relations
EU_O	Observer's expected utility
EU_T	Target's expected utility

Table 1: Symbols

```
Y_rel = findMinIndPartitions(Y_req, BK^obs)
for each(Y^i_rel ∈ Y_req)
  do n times in parallel
    SimulatedAnnealing(Y^i_rel)
  enddo
endfor
```

Figure 4: Maximization algorithm

be generalized. The algorithm therefore has to search for a joint generalization scheme for these N tuples. If there are m levels of generalization per attribute (on average) and N tuples, the number of different generalization schemes is $O(m^N)$. Two important properties of the objective function are: (i) minimal independent partitions can be solved independently and in parallel, and (ii) the utility of a piece of information is never smaller than its generalization. Given the exponential size of the feasible region, the need for real-time solutions, the parallel nature of the problem formulation, and the distributed computing affinity of sentient systems, we propose a distributed stochastic solution based on distributed simulated annealing (Fig. 4). We use the Rete algorithm [19] to compute potential inferences. By making a few assumptions on the form of the knowledge base and setting the parameters for the simulated annealing appropriatly our algorithm has a time complexity polynomial w.r.t. the size of the knowledge base and number of queries.

4.4 Implementation

The high-level design for integrating the Privacy Manager into SATware is shown in Fig. 5. In the Policy Manager, privacy and utility policies are specified by users through the Policy Editor, validated by the Policy Processor, and stored into the Privacy DB. The policy editor obtains the users policies by adapting a utility elicitation process [13] based on having the user (i) specify a utility preference network and (ii) anchor the utility of some pieces of information according to the scale from Fig. 6. The knowledge base represents the background knowledge of users and it is partly populated by system and space administrators and partly learned (on-the-fly) by the system using the BK Generator, and stored into the BK-DB. The Disclosure Control analyses the possible information (using the proposed distributed simulated annealing technique) that the observer could infer and, with the active policies, decides which information should be generalized and how.

4.5 Experiments

The main goal of the disclosure control submodule is to be able (i) to produce good results (high utility with adequate privacy) and (ii) to do so in real-time. To test the parallel simulated annealing based approach (PSA), we compared it to two simpler and centralized approaches: a brute force search (BF) and a brute force search based on a less expressive anonymity-based definition of privacy (minGen). Some of the results are summarized in Fig. 7 and Fig. 8, the rest are published in [36][2]. We instantiated 5 variations of PSA.

[2]The results here presented are an average over 33 runs on a dual-core machine featuring an AMD Turion 64 X2 at 2.0Ghz, with 3GB of memory, and running Linux

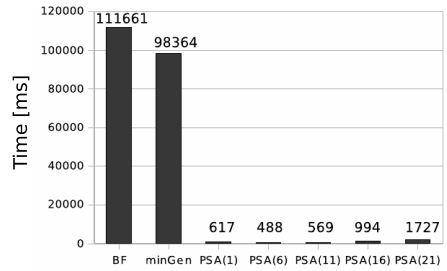

Figure 7: Avg. time overhead of PSA with different concurrent explorations vs BF and minGen

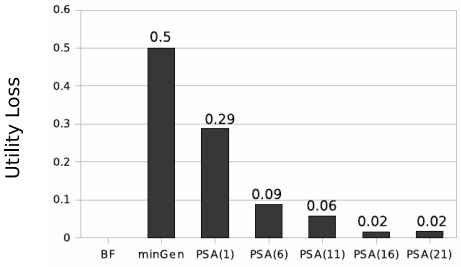

Figure 8: Avg. utility loss of PSA with different concurrent explorations vs BF and minGen

These instantiations differed on the degree of concurrent exploration (i.e., number of threads) being used. We call these PSA(x) where x is the number of concurrent explorations. Fig. 7 and Fig. 8 show that while the BF approach always finds the optimal, it takes far more time than the PSA approaches (minutes *versus* hundreds of milliseconds), which manage to return a solution very close to the optimal when incrementing the amount of concurrent exploration and with less than a second.

5. IMPLEMENTATION

SATware is being developed in the context of the Responsphere infrastructure at the UC Irvine campus which is a unique publicly accessible testbed for interdisciplinary research in situation monitoring and awareness in the context of emergency response applications. Currently, Responsphere includes more than 200 sensors of 10 different types deployed over the geographical space that covers about half of the UCI campus. The sensors range from network pan-tilt-zoom fixed video cameras, microphones, wireless motes with accelerometers, temperature sensors, and motion sensors to RFID tag readers and networked people counters.

The Responsphere-SATware framework has been and is being used to test and develop applications such as privacy-preserving video surveillance [24, 7, 45], situational awareness for firefighters (SAFIRE) [6], building visitor tracking [5], pedestrian traffic analyzer [26], technology-induced recycling behaviour, fresh coffee alerts, and others. For up-to-date videos and demos of these and other applications the reader is referred to the SATware website [8].

6. RELATED WORK

Stream processing engines (SPEs) like TelegraphCQ [15], STREAM [11], S3 [12], Cayuga [16], Aurora [14], Borealis [9], and MedSMan [31], focus on answering continuous queries on streams. With the exception of Aurora and Borealis, these systems focus on providing support for SQL-like queries with sliding windows. Aurora and Borealis focus on a "Box-and-Arrow" programming model where one describes

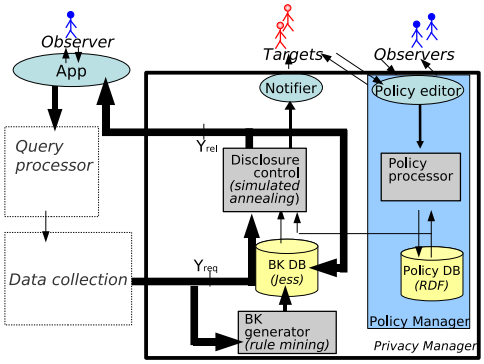

Figure 5: PrivacyManager

Target	
Labels	**Utilities**
Extremely Sensitive	−1.00
Very Sensitive	−0.75
Sensitive	−0.50
Somewhat Sensitive	−0.25
Not Sensitive	0.00
Observer	
Labels	**Utilities**
Don't Care	0.00
Information Curiosity	0.25
Information Useful	0.50
Information Needed	0.75
Always Needed	0.99

Figure 6: Utility Scales

queries as a graph of operators with a series of parameters. SPEs usually execute the queries on a centralized server and thus the main challenges orbit around overcoming a constraint memory and CPU with static solutions such as chain scheduling [14] and semantic load shedding [43] as well as dynamic solutions such as per-tuple dynamic routing [15, 29].

In contrast with SPE's architectural design, some work on wireless sensor networks (WSN) have focused on in-network processing. There has been two main themes in this area. Some work has focused on improving ad-hoc programming of sensor networks by providing ways of uploadiong new code to each node [18, 30]. Others focused on providing a database-like view of the sensor network and pushing the execution of the relational operators into the WSN notes [34, 33].

Service-oriented middleware (SOM) for pervasive spaces like Gaia [3], Oxygen [4], PICO [27], Scooby [38], and Aura [41], takes an approach closer to the "Box-and-Arrow" from Aurora and Borealis. Namely, applications are described as graphs of services. Each device in the pervasive/ubiquitous space advertises its capabilites as services. The main challenges then include how to optimally perform a QoS-based service discovery and composition [21, 23, 20, 27], proactive and reactive failure resilience [22], and dynamic swapping of services and service graphs.

None of the previous approaches, however, provides the level of abstraction desired for programming pervasive spaces. All of them still require the application to specify *how* to answer a query. SPEs require applications to specify *which streams* to connect. WSNs expect applications to specify *which sensed* data they are interested it. SOMs require applications to specify *which services* are needed. Our system, on the other hand, allows applications to specify *what* they are really interested in and SATware decides *which sensors* and *streams* to instantiate. Moreover, knowing *what* the applications are interested in allows the system to take more sensible decisions regarding system optimization, reliability, privacy, and the like.

Regarding privacy, several privacy and anonymity definitions and metrics along with specific solutions have been proposed in the literature. For instance, in the statistical data publishing domain, metrics such as k-anonymity [39] [42], l-diversity [32], and others [35] have been proposed. In [17] the author proves the impossibility for absolute privacy in statistical databases and defines the alternative metric of *differential privacy*, which is a metric relative to the risk of a user participating in a statistical database. Nonetheless, none of the previous definitions applies to our scenario: we need a non-binary definition regarding information that is not useful in an anonymous/statistical manner.

Our privacy work here complements QoS-related work in semantic load shedding in SPEs in that whenever a tuple needs to be dropped, we dropped the one with the least utility. The difference is that the need to drop tuples is triggered by potential privacy violations whereas in traditional SPEs systems is triggered by rescource scarcity.

Privacy in pervasive spaces has been researched at multiple levels. At the network layer, [10] combines hop-to-hop routing based on handles with limited public-key cryptography to preserve privacy from eavesdroppers and traffic analyzers. At the architectural level, solutions such as Cricket [37] and Place Lab [40] protect a user's (private) location by limiting the execution of location algorithms and location-based services to the user's carry-on device. In contrast, we assume that the sensor might not have enough context and resources to compute observations nor it is the final recipient of information. Other work regarding privacy in pervasive spaces includes our own work on private event detection with trusted sensors but untrusted storage infrastructure [24].

7. CONCLUSIONS

The dissertation whose current status is summarized in this paper focuses on the design and implementation of a middleware to enable pervasive-space applications. Namely, we propose an architecture that provides database-like views of the space. Applications query these views as they would a database. These queries are usually composed of static attributes and relationships (e.g., a firefighter's rank) and observable attributes and relationships (e.g., the level of O_2 in a firefighter's oxygen tank). The middleware provides continuous answers to the observable part of the query by deploying the necessary virtual sensors, each of which is a pre-specified set of operators that interprets raw data coming from sensors to obtain the more abstract (and semantically meaningful) value of an observable attribute or relationship.

The virtual sensor abstraction not only enables programming at a higher level, but it also enables the middleware to reason at a semantic level and provide more meaningful privacy protection. Before disclosing the query answer to the application, the privacy submodule modifies it to meet the user specified privacy policies while maximizing the utility of the answer for the application users. In constrast with more traditional binary definitions where information is either public or private, we propose a utility-based definition where information is associated with a positive utility for the querier and a negative utility for the target of the query. Moreover, further information that the querier might be able to infer is also associated with a negative utility. With this definition, we propose a framework where the system has to decide, at every time instant, which information should be generalized and how much, such that privacy is preserved and utility for the querier is maximized. We solved the maximization problem with an algorithm based on distributed simulated annealing. To realistically instantiate such an ap-

proach, we also had to address the problem of obtaining and representing the utility functions and obtaining and representing a user's background knowledge. We proposed solutions for both problems.

Acknowledgements

We would like to thank the SATware research group, the rest of the PhD committee of the dissertation here described, and the anonymous reviewers of this and other papers that are part of the dissertation. Their valuable input and their contributions on coauthoring papers and code has made it possible for this research to come to live. Special thanks to Roberto Gamboni, Jay Lickfett, Jonathan Cristoforetti, Alessandro Ghigi, Francisco Servant, Ronen Vaisenberg, Shengyue Ji, Hojjat Jafarpour, Minyoung Kim, Jooyoung Park, Kyoungwoo Lee, Mamadou Diallo, Bijit Hore, Chris Davison, Jon Hutchins, Utz Westermann, Gloria Mark, Ramesh Jain, and Don Patterson.

This work has been partially supported by the NSF under award Numbers 0331707, 0331690, and 0403433.

8. REFERENCES

[1] A. Arasu, S. Babu, and J. Widom, J. The CQL continuous query language: Semantic foundations and query execution. The VLDB Journal, 2006.

[2] B. Hore, H. Jafarpour, R. Jain, S. Ji, D. Massaguer, S. Mehrotra, N. Venkatasubramanian, and U. Westermann. Design and Implementation of a Middleware for Sentient Spaces. In ISI'07, 2007.

[3] Gaia OS, September 2007.

[4] Oxygen project, September 2007.

[5] RFID Tag lookup. http://www.ics.uci.edu/community/events/openhouse/rfid.php, 2009.

[6] SAFIRE. http://www.ics.uci.edu/%7Eprojects/cert/verticals.html, 2009.

[7] SATrecorder. http://www.ics.uci.edu/%7Eprojects/SATware, 2009.

[8] SATware website. http://satware.ics.uci.edu, 2009.

[9] D. J. Abadi, Y. Ahmad, M. Balazinska, U. Çetintemel, M. Cherniack, J.-H. Hwang, W. Lindner, A. S. Maskey, A. Rasin, E. Ryvkina, N. Tatbul, Y. Xing, and S. Zdonik. The Design of the Borealis Stream Processing Engine. In *Proceedings of the 2005 CIDR Conference*, 2005.

[10] J. Al-Muhtadi, R. Campbell, A. Kapadia, M. Mickunas, and S. Yi. Routing through the Mist: Privacy Preserving Communication in Ubiquitous Computing Environments. In *ICDCS*, volume 22, pages 74–83, 2002.

[11] 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, Book chapter, March 2004.

[12] L. Brown, A. Hampapur, J. Connell, M. Lu, A. Senior, C.-F. Shu, and Y. Tian. IBM Smart Surveillance System (S3): An open and extensible architecture for smart video surveillance (demo). In *ICCV05*, 2005.

[13] S. Buffett and M. Fleming. Applying a Preference Modeling Structure to User Privacy. *NRC Publication Number: NRC 49372*, 2007.

[14] D. Carney, U. Cetintemel, M. Cherniack, C. Convey, S. Lee, G. Seidman, M. Stonebraker, N. Tatbul, and S. Zdonik. Monitoring streams–a new class of data management applications. Technical Report CS-02-04, Brown Computer Science, February 2007.

[15] S. Chandrasekaran, O. Cooper, A. Deshpande, M. J. Franklin, J. M. Hellerstein, W. Hong, S. Krishnamurthy, S. Madden, V. Raman, F. Reiss, and M. Shah. TelegraphCQ: Continuous Dataflow Processing for an Uncertain World. In *Proceedings of the 2003 CIDR Conference*, 2003.

[16] A. J. Demers, J. Gehrke, M. Hong, M. Riedewald, and W. M. White. Towards expressive publish/subscribe systems. In *EDBT*, pages 627–644, 2006.

[17] C. Dwork et al. Differential privacy. *Proc. ICALP*, 2006.

[18] C.-L. Fok, G.-C. Roman, and C. Lu. Rapid development and flexible deployment of adaptive wireless sensor network applications. In *ICDCS'05*, 2005.

[19] C. Forgy. Rete: a fast algorithm for the many pattern/many object pattern match problem. *IEEE Computer Society Reprint Collection*, pages 324–341, 1991.

[20] X. Gu and K. Nahrstedt. Dynamic qoS-aware multimedia service configuration in ubiquitous computing environments. In *ICDCS*, pages 311–318, 2002.

[21] X. Gu and K. Nahrstedt. Distributed multimedia service composition with statistical qoS assurances. *IEEE Transactions on Multimedia*, 8(1):141–151, 2006.

[22] X. Gu and K. Nahrstedt. On composing stream applications in peer-to-peer environments. *IEEE Transactions on Parallel and Distributed Systems*, PDS-17(8):824–837, Aug. 2006.

[23] X. Gu, K. Nahrstedt, R. N. Chang, and C. Ward. QoS-assured service composition in managed service overlay networks. In *ICDCS*, page 194. IEEE Computer Society, 2003.

[24] B. Hore, J. Wickramasuriya, S. Mehrotra, N. Venkatasubramanian, and D. Massaguer. Privacy-preserving event detection in pervasive spaces. In *PerCom 2009*, 2009.

[25] G. Hsieh, K. Tang, W. Low, and J. Hong. Field deployment of IMBuddy: A study of privacy control and feedback mechanisms for contextual im. *Lecture Notes in Computer Science*, 4717:91, 2007.

[26] A. Ihler, J. Hutchins, and P. Smyth. Adaptive event detection with time-varying poisson processes. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 207–216, New York, NY, USA, 2006. ACM Press.

[27] S. Kalasapur, M. Kumar, and B. Shirazi. Dynamic service composition in pervasive computing. *IEEE Trans. Parallel Distrib. Syst*, 18(7):907–918, 2007.

[28] M. Kifer and A. Li. On the semantics of rule-based expert systems with uncertainty. *LNCS on ICDT*, 88:102–117, 1988.

[29] I. Lazaridis and S. Mehrotra. Optimization of multi-version expensive predicates. In *SIGMOD 07*, New York, NY, USA, 2007. ACM Press.

[30] P. Levis and D. Culler. Mate: A tiny virtual machine for sensor networks. In *ASPLOSX*, 2002.

[31] B. Liu, A. Gupta, and R. Jain. MedSMan: A Streaming Data Management System over Live Multimedia. In *MM'05*. ACM, 2005.

[32] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. l-Diversity: Privacy Beyond k-Anonymity.

[33] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. Tag: a tiny aggregation service for ad-hoc sensor networks. In *USENIX OSDI*, 2002.

[34] S. R. Madden, M. J. Franklin, and J. M. Hellerstein. TinyDB: An Acquisitional Query Processing System for Sensor Networks. *ACM Transactions on Database Systems*, 2004.

[35] D. Martin, D. Kifer, A. Machanavajjhala, J. Gehrke, and J. Halpern. Worst-Case Background Knowledge for Privacy-Preserving Data Publishing. In *ICDE*, 2007.

[36] D. Massaguer, B. Hore, M. H. Diallo, S. Mehrotra, and N. Venkatasubramanian. Middleware for pervasive spaces: Balancing privacy and utility. In *Middleware 2009*, 2009.

[37] N. Priyantha, A. Chakraborty, and H. Balakrishnan. The Cricket location-support system. In *MobiComp*, pages 32–43. ACM Press New York, NY, USA, 2000.

[38] J. Robinson, I. Wakeman, and T. Owen. Scooby: middleware for service composition in pervasive computing. In *MPAC '04: Proceedings of the 2nd workshop on Middleware for pervasive and ad-hoc computing*, pages 161–166, New York, NY, USA, 2004. ACM Press.

[39] P. Samarati and L. Sweeney. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. Technical report, In Proceedings of the IEEE Symposium on Research in Security and Privacy, 1998.

[40] B. Schilit, A. LaMarca, G. Borriello, W. Griswold, D. McDonald, E. Lazowska, A. Balachandran, J. Hong, and V. Iverson. Challenge: ubiquitous location-aware computing and the" place lab" initiative. In *WMASH*, pages 29–35. ACM New York, NY, USA, 2003.

[41] J. Sousa and D. Garlan. Aura: An architectural framework for user mobility in ubiquitous computing environments, 2002.

[42] L. Sweeney. Achieving k-Anonymity Privacy Protection Using Generalization and Suppression. *Int'l journal of uncertainty fuzziness and knowledge based systems*, 10(5):571–588, 2002.

[43] N. Tatbul, U. Çetintemel, S. B. Zdonik, M. Cherniack, and M. Stonebraker. Load shedding in a data stream manager. In *VLDB*, pages 309–320, 2003.

[44] A. Toninelli, R. Montanari, L. Kagal, and O. Lassila. A semantic context-aware access control framework for secure collaborations in pervasive computing environments. In *International Semantic Web Conference*, pages 473–486, 2006.

[45] J. Wickramasuriya, M. Datt, S. Mehrotra, and N. Venkatasubramanian;. Privacy protecting data collection in media spaces. In *ACM Multimedia 2004*, 2004.