# A System for Semantic Data Fusion in Sensor Networks

Alex Wun, Milenko Petrovic, Vinod Muthusamy, and Hans-Arno Jacobsen

Department of Electrical and Computer Engineering
University of Toronto
wun,petrovi,vinod,jacobsen@eecg.utoronto.ca

## Introduction

Emerging sensor network technologies are expected to substantially augment applications such as environmental monitoring, healthcare, and home/commercial automation [1]. By nature, sensor networks will be highly customized for specific applications and use highly customized protocols and data formats. As sensor networks become more pervasive, there will be a need to hide their internals from client applications. This implies more than simply providing a declarative interface to access sensor network data, which would still burden applications with the task of interpreting the context and meaning of sensor data. For example, an application may simply be interested in whether a conference room is "crowded" and not the specific audio, temperature, and motion sensor readings used to determine the status of the room. However, much of the work today focuses mainly on posting sensor level data from isolated sensor networks to the Internet [2, 3]. There has been little emphasis on how this data should be presented to applications as *higher-level* phenomena, thereby enabling the kind of utility envisioned by grid and pervasive sensor networks. In order to infer these high-level events, sensor data needs to be filtered, aggregated, and correlated from heterogeneous (and possibly geographically dispersed) sensor networks. In other words, data fusion must occur across independent sensor networks.

This is especially true in large-scale collaborative sensor network deployments geared towards observing emergent phenomena. For instance, Neptune is one such joint project between the U.S./Canada whose goal is to establish a long-term sensor network observatory in the northeast Pacific Ocean [4]. They are interested in identifying large-scale events such as submarine volcanic eruptions by fusing data from a multitude of physical, chemical, and biological sensor readings. In practice, distinguishing such events will require interpreting diverse sensor readings based on the knowledge of domain experts. It must consequently be easy to define, redefine, and modify the semantics defining any given event in order to adapt to new and improving domain knowledge. In other words, data fusion must also occur based on highly specialized and dynamic application semantics. These semantics must be *independent* of application and sensor network interfaces in order to facilitate interoperation and integration.

The challenge is to give applications the ability to exploit the full utility of sensor networks without being burdened with sensor level data. This is difficult because it requires fusing data across heterogeneous networks under semantics that differ between application domains. Even within a domain, applications may need to access sensor networks in a way that reflects their own context without affecting the interfaces seen by coexisting applications.

In this demonstration, we present a novel system to address these challenges. Our approach allows applications to express interest in semantic events that are transformed into sensor network level events. The semantic events are "defined" independently of the application and sensor network interfaces. Our main contribution is a system that allows efficient semantic event evaluation to occur within the sensor network while providing flexible high-level interfaces to applications. Additionally, we use a publish/subscribe messaging model in order to take the event-driven nature of these scenarios into consideration.

## System Architecture

The system in this demonstration is based on two of our prior projects to develop a semantic publish/subscribe engine and a sensor network publish/subscribe middleware platform. We will first briefly overview publish/subscribe and these two prior projects before describing how they fit together in our overall system.

### Publish/Subscribe Background

Publish/Subscribe[1] is an event-based messaging model that delivers *event notifications* to clients based on their prior expressed interests. There are three roles

[1]For brevity, we will refer to publish/subscribe as just pub/sub hereafter

in a pub/sub system: subscribers, publishers, and event dispatchers. Subscribers express interest in receiving event notifications by issuing *subscriptions* that describe their interest to the event dispatching system (consisting of one or more event dispatchers). The dispatchers store all subscriptions they receive. Similarly, publishers issue *events* to the dispatching system which is responsible for matching events against known subscriptions and subsequently forwarding these events to the appropriate subscribers. Events are generated as sets of attribute-value pairs while subscriptions are expressed as value filters on a set of attributes[2]. However, a major problem with current pub/sub systems is that event matching is entirely based on strict syntax. Therefore, it is difficult to achieve the scenarios discussed in the introduction since they require semantic decoupling across domains.

## Semantic Publish/Subscribe Engine

Current matching engines require subscriptions and events to be expressed according to a strict syntax in order for correct matching to occur. Suppose a subscriber expresses interest in being notified of earthquakes in North America by issuing the subscription: *S: (alert = earthquake)* $\wedge$ *(region = North America)* $\wedge$ *(magnitude* $\geq$ *5)*. A publisher may issue an event in the form: *E: (alert, earthquake) (country, U.S.) (richter, 6.5)* In traditional pub/sub systems, this issued event would incorrectly fail to match the subscription filter above since neither the attributes *country* and *richter* nor the value *U.S.* are expressed using the same syntax as the subscription. S-ToPSS[3] is a system that extends event matching with semantic awareness using three independent and increasingly powerful methods collectively defined in a domain *ontology*.

Synonym and taxonomy translation are two simple methods for expressing equivalence and hierarchical relationships between terms, respectively. Synonym translation allows the matching engine to map attributes and values into "canonical" terms within an application domain. With taxonomy translation, events containing more specialized terms than those used in a subscription (from the same hierarchy) are considered to match. While events containing more generalized terms than those used in a subscription do not match.

While synonyms and taxonomies capture common data representation techniquues, the most powerful method involves arbitrary mapping functions that correlate one or more attribute-value pairs to one or more semantically related attribute-value pairs. Mapping functions allow domain experts to express arbitary relationships that otherwise cannot be accounted for by the previous two methods. For example, a mapping function could be defined to transform the event $E_1'$*: (station-A-seismic, 5.5)* $\wedge$ *(station-B-seismic, 2)* $\wedge$ *(station-C-seismic, 4)* into a semantically equivalent event *E: (alert, earthquake)* $\wedge$ *(region, North America)* $\wedge$ *(magnitude, 6.5)* that would match the high-level subscription issued above. In the context of sensor networks, mapping functions can be used to perform calculations on predicate values for capturing concerns such as uncertainty or sensor noise. For instance, an event could contain data readings obtained from multiple sensors around a phenomenon (such as a chemical hazard). A mapping function could easily be written by a domain expert to average, transform, or *fuse* these readings into a semantically equivalent event.

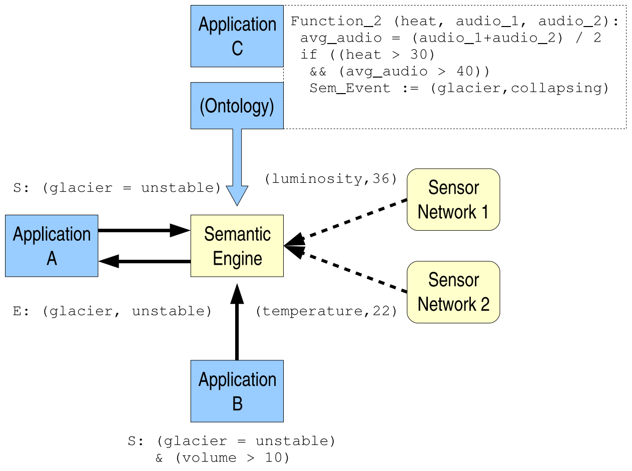

```
Function_2 (heat, audio_1, audio_2):
 avg_audio = (audio_1+audio_2) / 2
 if ((heat > 30)
  && (avg_audio > 40))
  Sem_Event := (glacier,collapsing)
```

Figure 1: Event translation through semantic engine

## Sensor Network Middleware

Sensor networks are primarily event-based, either generating data at predefined intervals or upon detecting specific events. To our knowledge, current sensor network projects tend to "hardcode" sensor behaviour into their networks. However, as [5] observed, cooperating sensor networks will need to run flexible data-centric middleware capable of directing sensor behaviour for achieving higher-level application goals. Micro-ToPSS[4] is a sensor network middleware that facilitates sensor level pub/sub capabilities. By extending the TinyScript [6] language and its associated VM to surface these new capabilities, we enable application awareness within the sensor network via a light-weight pub/sub system.

Applications in Micro-ToPSS are decomposed into a collection of *event handlers*, which are modular units of processing compiled from an application schema. Event handlers run on top of the Micro-ToPSS middleware issuing subscriptions to control sensor data flows in the network. Event handlers are also responsible for collecting sensor readings and issuing events through

[2]Simple examples of a subscription and a corresponding event that matches are:
*S: (temperature > 10)*
*E: (temperature, 25)*

[3]Semantic-Toronto Publish/Subscribe System

[4]Micro-Toronto Publish/Subscribe System

their local middleware. The handlers can be thought of as traditional pub/sub subscribers and publishers compacted into specialized, application-specific modules. The middleware is similar (in principle) to traditional pub/sub event dispatchers in that it stores subscriptions and performs matching of sensor events to subscriptions. Successfully matched events are routed back to the handler that originated the subscription. Sensor data readings are collected or transmitted only if an application (decomposed into handlers) interested in the data exists somewhere in the network.

**Semantic Fusion of Sensor Network Data**

In our system architecture, we assume that sensor networks are administered independently. By deploying our Micro-ToPSS middleware platform onto nodes inside the networks, we facilitate application-aware data collection within the network. External applications can specify the sensor data that is to be collected through an interface provided by the sensor networks. Application-specific data collection requests can be transformed into a collection of event handlers and deployed into the sensor network. As a result, the sensor network will only collect and transmit data that is relevant to client applications. In our system, we chose to implement the sensor network interface as a Web Service.

This interface abstracts sensor network protocol details, but the applications still receive sensor level data. This data may be processed (i.e. aggregated, filtered, averaged, etc.) but it is still expressed in the context of the sensor network. In other words, if the sensor network collects light and temperature readings, then the application will receive light and temperature readings in some form. As discussed in our introduction, applications may not be interested in the sensor level data but rather, the semantic meaning of the data. It is possible for the sensor network to provide high-level interfaces directly as well. But the utility of the sensor network then becomes much more restricted since the semantics of the high-level interface (as defined by the sensor network) may not necessarily match the semantics desired by all external applications.

By deploying a *semantic engine* (S-ToPSS) as a middleware intermediary between applications and sensor networks, we allow applications to exploit the full utility of sensor level data without unnecessarily encumbering the applications with sensor level contexts. Applications can make high-level declarative subscriptions to the semantic engine, which is responsible for semantically fusing data from disparate sensor networks. The semantics of the data fusion are defined entirely by an ontology. And the management of these ontologies is independent from the management of application and sensor network interfaces. For our demonstration, the interface to the semantic engine is also implemented as a Web Service.

Figure 2 illustrates a simple example using two sensor networks to monitor the stability of a glacial sheet. One network contains light sensors on the surface of the glacier, while the other contains temperature and audio sensors deeper within the glacier[5]. The two networks happen to be physically co-located on the same sheet for our example. The sensor networks each supply a *physical schema* to the semantic engine that describes the full range of actual sensing capabilities provided by the networks as shown in Figure 2(a). The semantic engine is also supplied with an ontology that defines synonymous terms and a mapping function. Given an ontology and physical schemas, the semantic engine can determine which sensor networks need to be "*tasked*" and what subscriptions are valid. In Figure 2(b) we have shown the networks being tasked as a result of the input ontology, but this can also occur only as subscriptions are received to preserve efficiency. With this example ontology, applications may issue a subscription involving any combination of synonyms and the high-level glacial state event. In Figure 1, application A is issuing a high-level subscription directly specified in the ontology. Application B may choose to augment the high-level subscription with a sensor reading relevant to its own context from the synonyms dictionary (sensor network 2 will need to be additionally tasked for audio sensing in this case). Application C may choose to supply additional domain expert knowledge to the semantic engine by describing new mapping functions, synonyms, or taxonomies. In this example, application C has defined a new mapping function that considers sensor noise by averaging two audio readings before mapping to a new semantic event. Ontologies may exist on a per-application basis, or a single ontology may pull together semantic definitions from across application domains.

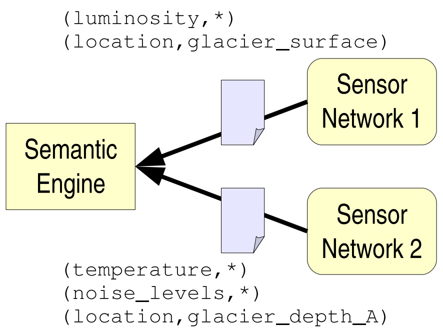
## Related Work

Mainwaring et al. [2] established a sensor network for monitoring seabird nesting environmental conditions on an island. Their approach makes environmental sensor data readings available through a standard Internet website. Our system is complementary to their project as we investigate allowing applications to semantically leverage sensor networks.

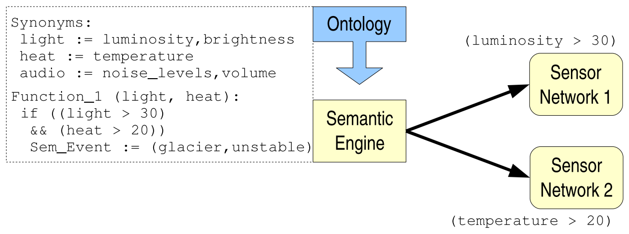
Similar to our project, Gaynor et al. [7] propose integrating geographically dispersed sensor networks. They investigate allowing applications to query sensor network data through a Data Collection Network consisting of host systems on the Internet. Translation between application queries and sensor queries occurs at Sensor Entry Points. Our semantic system would greatly improve flexibility and decoupling, complementary to their approach.

Both Madden et al. [8] and Yao and Gehrke [9] allow

[5]This example is for illustrating the utility of our approach. Our actual demo will instead be based on the environment of the demonstration area but will use similar sensors.



(a) Registration of physical schema



```
Synonyms:
 light := luminosity,brightness
 heat := temperature
 audio := noise_levels,volume
Function_1 (light, heat):
 if ((light > 30)
  && (heat > 20))
  Sem_Event := (glacier,unstable)
```

(b) Ontology definition and sensor network tasking

Figure 2: Dataflow within our system architecture

users to make declarative queries into sensor networks by modelling the networks as databases. As such, they focus on database-style queries such as aggregation, averaging, etc. However, the queries still primarily return sensor network relevant data. Applications will still need to interpret the data in an application context, which our semantic system facilitates by becoming flexible around query syntax. Our system is also based on a pub/sub rather than database model.

## Demonstration Description

In our research group, we are investigating the application of publish/subscribe messaging models and system architectures to sensor networks. Using a small number of basic sensors[6], we will setup a simple proof-of-concept sensor network to demonstrate "environmental monitoring" scenarios similar to those we discussed earlier. The sensor network interface will be a Micro-ToPSS aware Web Service running on a gateway node[7]. Although we will be limited by the environment of the actual demonstration area and sensor motes, a simple sensor network will be sufficient since we focus on demonstrating the data fusion system itself rather than the application scenario.
Our demonstration will use predefined ontologies describing different environmental monitoring domains (such as the glacier example). We will demonstrate how applications can issue high-level subscriptions through the S-ToPSS Web Services interface when subscribing to sensor network events. Three key features of our system will be highlighted in the demonstration: (1) Describing the same high-level event under different ontologies. For instance, an "unstable glacier" may imply high light and temperature readings or high audio readings depending on the ontologies used. (2) Interpreting the same sensor data in different ontologies. For instance, the same light, temperature, and audio readings can be interpreted differently as "collapsing glacier" or "crowded conference room" events under different ontologies. (3) Data fusion of sensor readings. For instance, the mapping function that defines an "unstable glacier" event can aggregate audio readings from multiple sensors to achieve more accurate results as part of the event mapping process. These key features can be transparently leveraged by applications in our system.

[6]We will be using Mica2 and Mica2dot Berkeley mote hardware equipped with basic sensor boards that provide light, audio, and temperature readings.

[7]The Stargate unit from www.xbow.com

## References

[1] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless Sensor Networks: a Survey," *Computer Networks*, vol. 38(4), pp. 393–422, 2002.

[2] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson, "Wireless Sensor Networks for Habitat Monitoring," in *MOBICOM*, 2002.

[3] T. Luckenbach, P. Gober, S. Arbanowski, A. Kotsopoulos, and K. Kim, "TinyREST: a Protocol for Integrating Sensor Networks into the Internet," in *REALWSN*, 2005.

[4] "Neptune," http://www.neptune.washington.edu/.

[5] Y. Yu, B. Krishnamachari, and V. K. Prasanna, "Issues in Designing Middleware for Wireless Sensor Networks," *IEEE Network*, vol. Jan/Feb, pp. 15–21, 2004.

[6] P. L. D. Gay and D. Culler, "Active Sensor Networks," in *NSDI*, 2005.

[7] M. Gaynor, S. L. Moulton, M. Welsh, E. LaCombe, A. Rowan, and J. Wynne, "Integrating Wireless Sensor Networks with the Grid," in *IEEE INTERNET COMPUTING*, 2004.

[8] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "TAG: a Tiny AGgregation Service for Ad-Hoc Sensor Networks," in *OSDI*, 2002.

[9] Y. Yao and J. Gehrke, "The Cougar Approach to In-Network Query Processing in Sensor Networks," in *SIGMOD*, 2002.