VIRTUAL SENSORS FOR SERVICE ORIENTED INTELLIGENT ENVIRONMENTS

Raja Bose, Abdelsalam (Sumi) Helal, Vishak Sivakumar and Shinyoung Lim
Mobile and Pervasive Computing Laboratory
Department of Computer and Information Sciences and Engineering
University of Florida, Gainesville, FL, U.S.A.
{rbose, helal, vss, lim} @cise.ufl.edu

ABSTRACT

Service-oriented architecture (SOA) is a popular industry standard that provides an elegant model to create and compose interoperable applications. In the case of sensor networks, SOA is considered to be a very attractive enabler for sensor programmability and self-integration. However, SOA cannot be used without additional mechanisms to enhance the reliability and availability of the sensors behind the services. In the absence of such mechanisms, sensor-based SOA could provide a false promise of service whose, quality and availability are highly questionable and cannot be guaranteed. In this paper, we propose a Virtual Sensor framework for service-oriented sensor networks with the aim of enhancing reliability, prolonging availability and enabling automatic service composability. We classify and model three useful virtual sensor concepts and present compensation algorithms for coping with sensor failure. To quantify the merit of our approach, we developed a Data Quality Indicator measure for expressing the perceived quality of virtual sensor readings.

KEYWORDS *– Distributed Sensor Networks, Virtual Sensors, Service-Oriented Architecture, Reliability, Availability, Service Composability*

1. Introduction

Some of the technical issues faced by real-life sensor network deployments are fault tolerance, scalability, production cost, and power consumption [1]. From the perspective of application developers and service providers, sensor networks also need to provide mechanisms which make them easy to access, program and customize. For this reason, service-oriented sensor networks have evolved to allow easy and efficient implementation of services and applications. The Atlas Platform [2, 3] is one such commercially available sensor and actuator platform which enables the creation and deployment of service-oriented sensor networks.

However, by their very nature, sensor networks are built using minimal cost components and processes which make them intrinsically unreliable and prone to failures [1, 4, 5]. They are also built as limited processing elements without rich software stacks that allow them to integrate and interoperate with traditional computing elements such as servers and mainframes. In this paper, we highlight three requirements that arise when sensor networks are deployed in a real environment.

- **Automatic Composition of Sensor Services.** Due to the massive proliferation of applications based on sensor networks, there are many scenarios where a single physical deployment of sensors has to service multiple applications. Hard coding sensor services to cater to specific applications or even having the applications themselves, specify exactly how a particular service composition should be performed is neither scalable nor feasible. Instead there needs to be a mechanism which allows automatic composition of sensor services in response to an application's needs.
- **Reliability.** Services offered by individual sensors provide no guarantee about their quality. There are rarely any built-in mechanisms within a sensor service which provides Quality of Service (QoS) for data originating from that sensor. Furthermore**,** services need to be aware when their functionality drops below a certain level. This enables them to either alert their client applications or simply go offline instead of potentially providing unreliable data.
- **Availability.** As mentioned previously**,** individual sensors are prone to high rates of failure. Hence, there is an urgent need for a sensor model which can cope with failure of multiple sensors and still be able to provide reasonable quality of data.

To address these issues, we propose a framework model that allows programmers to define and utilize "virtual" sensors, which are superior to physical sensors as they can provide guarantees and the quality of service as well. Our proposed framework is based on the Atlas sensor platform, whose service-oriented architecture facilitates the implementation of such a value-added concept. Specifically, the proposed Virtual Sensor framework enhances the reliability and availability of sensor services. It provides minimum guarantees of functionality and is capable of estimating the reliability of data originating from the sensors. It also has mechanisms for recovering from failures of multiple physical sensors and detecting degradation in a sensor's performance. Additionally, it allows for the automatic composition of sensor services in response to an application's needs, without requiring it to specify exactly, which and how sensor services need to be composed together.

This paper is organized as follows: Section 2 covers related research in this area. Section 3 provides a classification of virtual sensors along with their model specifications. Section 4 describes our framework model for virtual sensors including a compensation algorithm for maximizing availability and reliability. Section 5 gives a comparative analysis of data quality of sensor readings based on different levels of compensation. Section 6 provides a brief description of our implementation of the framework using the Atlas Platform. Finally, in Section 7 we give an outline of our on-going research and conclusions.

2. Related Research

A number of different approaches have been undertaken to resolve the problems of reliability, availability, and service composability in sensor networks. Among them are sensor fusion and failure detection, machine learning, data mining, architectural changing, using clustering algorithms, and collaborative processing infrastructure. Several recent research papers have focused on virtual sensors for abstracting data from physical sensors [6], estimating sensed data by considering noise on the image that is captured by image sensors [7], sensor fusion and failure detection [8], and multi-sensor management and information fusion [9]. Apart from these, other approaches for enabling virtual sensors have also been used such as those based on infrastructure and architectural changes. Gu et. al. [5] propose a formal and ontology-based context model using rules which describe context-aware behaviors. Hardy and Maroof [10] give an architectural model for virtual sensor integration which provides a framework for designing and constructing virtual sensors using a layered architecture. Chin et. al. [4] propose an ontology based programming-by-example approach which uses deconstructed appliance model, task oriented programming and ontology in conjunction with sets of rules.

3. Virtual Sensors

We define a virtual sensor as a software sensor service entity consisting of a group of sensors along with associated knowledge which enables it to provide services which are beyond the capabilities of any of its individual member sensors. Virtual sensors may be composed of a group of physical sensors or other virtual sensors.

We classify virtual sensors into three categories, namely, *Singleton virtual sensor*, *Basic virtual sensor* and *Derived virtual sensor.*

Each category of virtual sensor is modeled as a set of tuples which represents the knowledge possessed by them. One set of attributes (which we represent in bold type) is set at run-time when a service object representing a particular virtual sensor is created. The other set of attributes is set statically and stored as part of the sensor model definition in the knowledge base as described in Section 3.1.

3.1 Singleton Virtual Sensor

This type of virtual sensor represents a single physical sensor. It contains specific knowledge about the sensor in form of attributes such as location, the phenomena it detects, unit of measurement etc. For our purposes we assume that a physical sensor outputs a numeric value which is then converted using a conversion formula into a real-world measurement. We model a singleton virtual sensor as a 5-tuple: S = <**I**, **L**, T, U, F> where,

I – Sensor ID
L – Location of sensor
T – Type of phenomena detected
U – Unit of measurement for the phenomena
F – Conversion Formula where F = f(x) where x is a numeric reading from the physical sensor.

3.2 Basic Virtual Sensor

A basic virtual sensor is composed of a group of singleton virtual sensors of the same type. A basic virtual sensor detects the same type of phenomena which is detected by its member singleton virtual sensors. For example, a basic virtual temperature sensor can be composed of multiple singleton virtual sensors of type temperature. A basic virtual sensor provides greater reliability and availability of sensor data as compared to a corresponding singleton virtual sensor. It achieves this through the use of a compensation algorithm along with a Data Quality Indicator (DQI), details of which are presented in Section 4.4. We model a basic virtual sensor as a 7-tuple: B = <**I**, **L**, T, U, F, Q, **S**>, where,

S – Set of singleton virtual sensor instances which are members of this basic virtual sensor.
F – Aggregation Formula where $F = f(\{R_s\}_{s \in S})$ where R_s is a reading from singleton virtual sensor $s \in S$.
Q – QoS Formula
I, L, T and U have their usual meanings defined above.

3.3 Derived Virtual Sensor

A derived virtual sensor is composed of a group of basic and/or other derived virtual sensors of heterogeneous types. The type of phenomena sensed by this category of virtual sensor is more abstract than what can be detected by any of the member sensors. A Weather Sensor is an example of a derived virtual sensor since it can consist of basic temperature, pressure and humidity virtual sensors. These basic virtual sensors are themselves composed of multiple singleton temperature, pressure and humidity sensors respectively. A derived virtual sensor is modeled as a 8-tuple: D = <**I**,**L**,Y,**Z**,F,Q,M,R> ; where,

Y – Set specifying the basic/derived virtual sensors which are required by this derived sensor.
Z – Set of basic/derived virtual sensor instances which are members of this derived sensor.
F – Formula for integrating various sensor readings where $F = f(\{R_z\}_{z \in Z})$ where R_z is a reading from

basic/derived virtual sensor z ϵ Z.

M: F → R where R is the set of possible responses that can be given by this derived virtual sensor.

I, L and Q have their usual meanings as previously defined.

4. Framework Model for Virtual Sensors

In this section, we propose a framework model for managing the life cycle of virtual sensors. Our framework resides inside an existing implementation of service oriented architecture (SOA) such as OSGi [11] and hence, has access to its various built-in mechanisms for discovery, delivery, creation, modification and deletion of services. The framework architecture is shown in Fig. 1.

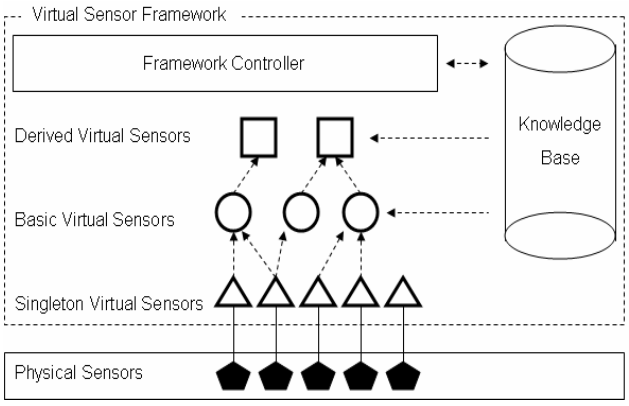

Fig. 1 Framework architecture

4.1 The Knowledge Base

The knowledge base is a central repository of information in the framework model. It is responsible for managing the storage, addition, modification and deletion of information records associated with virtual sensors running inside the framework. It stores the following types of information records which are available for lookup by both the framework controller and the virtual sensors:

- Sensor Model Definition: These records store the model information for each of the 3 types of virtual sensors defined in Sections 3.1, 3.2 and 3.3.
- Phenomena definition: These records store the definitions of various phenomena that can be detected by the sensor network. Examples include basic phenomena such as temperature and humidity and more complex phenomena such as weather, ambience and security. A phenomena definition is stored as a 2-tuple: P = <N, D> ; where,

  N – Name of the phenomena
  D – Model of the derived / basic virtual sensor which can detect this phenomena

  For each basic virtual sensor type T (defined in Section 3.1) created in the system, the corresponding phenomena definition <T, D> is automatically stored in the knowledge base.
- Reliability & Availability parameters: These records store some environment variables which are required by the virtual sensors for maximizing the availability and reliability of their data. These parameters are stored as 2-tuples: E = <P, V> ; where,

  P = N_T or P_T or a
  V = Value of the parameter
  N_T , P_T and a are defined in Section 4.4

4.2 Framework Controller

The framework controller is responsible for receiving queries from applications and then determining with the help of the knowledge base, which virtual sensors need to be created. It is also responsible for removing virtual sensors which are not being used by any of the user applications. We assume that a query from a user application is of the form: SELECT <phenomena_type> FROM <location>. The controller extracts the user's requirement from the query, namely, the type of phenomena and the location where it has to be detected. It then starts the process of creating and organizing virtual sensors in the framework to enable the sensor network to work towards fulfilling that query. This process is described in Section 4.3. The framework controller also keeps track of existing virtual sensors in the system and when they go offline. It does so by using the underlying mechanisms of the service oriented environment that it resides in. Last but not the least, the framework controller keeps track of sensors joining the network and for each new sensor, it creates a singleton virtual sensor. An important point to note here is the fact that basic and derived virtual sensors are created and destroyed based on user requirements but singleton virtual sensors are created whenever new physical sensors join the sensor network and are destroyed when the physical sensor they are representing fails or goes offline.

4.3 Enabling Service Composability

The virtual sensor framework performs automatic composition of sensor services in response to a user's requirements. Based on the type of phenomena that the user wants the sensor network to detect, the framework controller looks up the corresponding phenomena definition from the knowledge base. This definition contains a reference to the Sensor Model Definition of the basic or derived virtual sensor which can detect the phenomena. The framework controller retrieves the Sensor Model Definition from the knowledge base and sets some of the attributes in the model definition such as the sensor ID and sensor location. Based on the model definition, it then creates a new service object representing this virtual sensor. This newly created virtual sensor in turn, uses information from its model definition to create or access other service objects representing virtual sensors required by it and the process continues. Once all the virtual sensors have been created and organized the user is able to retrieve data from the sensor network. In case a virtual sensor fails to start due to a

software or hardware error, it returns an error message which is forwarded back up the hierarchy till it reaches the framework controller. The framework controller in turn notifies the user that its query cannot be completed at the present time.

We present an example scenario to demonstrate how all the components of the virtual sensor framework come together to enable automatic on-the-fly service composition. Consider a sensor network consisting of temperature and humidity sensors deployed in a large forested area which is divided into 4 grid boxes A,B,C and D as shown in Fig. 2.

We make the following assumptions regarding the sensor network setup: (1) The knowledge base contains the Sensor Model Definitions of the following virtual sensors: Singleton Temperature sensor (denoted by circle), Singleton Humidity sensor (denoted by square), Basic Temperature sensor (T), Basic Humidity sensor (H) and Derived Weather sensor (W). It also contains the Phenomena definition <'WEATHER', W> where W represents the model definition of the derived Weather sensor; (2) Singleton virtual sensors corresponding to each of the physical sensors have already been created by the framework controller but none of the basic and derived virtual sensors currently exist inside the framework and hence will need to be created on-the-fly in response to a user's query.

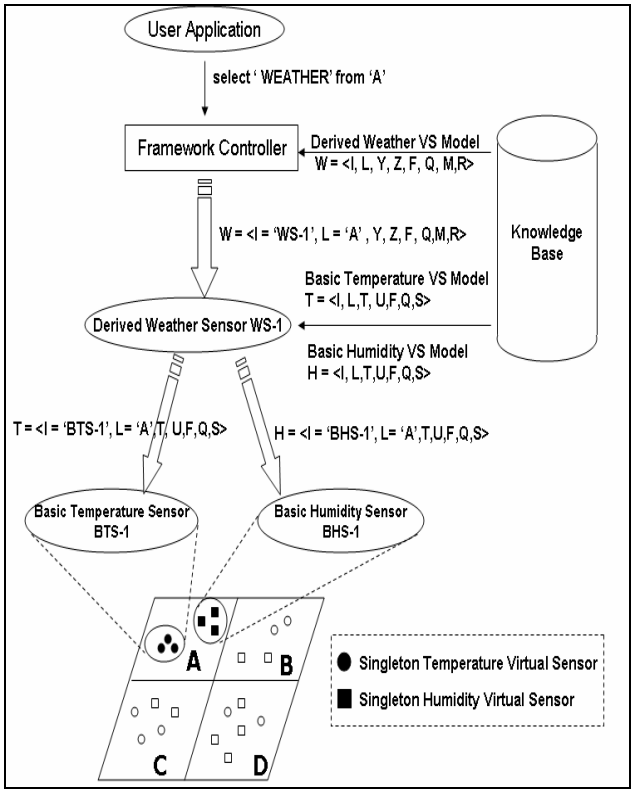

Fig. 2 Sensing Weather Phenomena

Suppose a user sends the following query to the framework: select 'WEATHER' from 'A'. The framework controller extracts the phenomena type, 'WEATHER' and the location, 'A' from the query. It looks up the knowledge base to retrieve the phenomena definition for 'WEATHER'. Using this it gets the sensor model definition for the derived Weather Sensor. It sets the location attribute in the model definition to 'A' and creates a service object representing a Weather sensor. From its model definition, the Weather sensor knows that it needs to get data from basic temperature and humidity virtual sensors located at 'A'. The Weather sensor then retrieves the sensor model definitions of these basic virtual sensors and sets their location attributes to 'A'. It then uses the model definitions to create a new service object representing each basic virtual sensor. Each of these basic virtual sensors then figure out from their model definitions that they need to aggregate readings from singleton virtual sensors of their respective types, which are located inside 'A'. Using mechanisms for service discovery and composition provided by the underlying SOA implementation, the basic virtual sensors are able to access and aggregate data from the singleton temperature and humidity sensors. Once all the virtual sensors are active and data starts flowing back up the chain, the controller notifies the user that it can now access weather information from location 'A' using services provided by the derived Weather sensor.

4.4 Availability and Reliability

We propose a compensation algorithm for Basic Virtual Sensors which enables them to increase their availability by adapting to failure of one or more Singleton Virtual Sensors. We also propose the concept of Data Quality Indicator (DQI) which allows a basic or derived virtual sensor to provide information regarding reliability of data originating from it.

4.4.1 Basic Virtual Sensor

A Basic Virtual Sensor consisting of N Singleton Virtual Sensors, outputs a single reading, which is an aggregate of the N readings obtained from them. For our purposes, the arithmetic mean of the readings is taken as the aggregate. In order to derive the compensation and Data Quality Indicator formulae for Basic Virtual Sensors, we define the following terms:

N = Number of Singleton Virtual Sensors which are initially members of this Basic Virtual Sensor

$S = \{s_i\}_{i = 1 \text{ to } N}$: Set of the Singleton Virtual Sensors

S_F = Set of dead Singleton Virtual Sensors

N_C = Number of Singleton Virtual Sensors currently alive

DQ_T = Minimum Data Quality Threshold. The Basic Virtual Sensor must provide readings of quality DQ_T or higher

T_{start} = Time when the sensor network was started up

R_{s_i} = Reading from sensor $s_i \in S$

N_t = Number of readings sampled till time t. See assumption (2) below

B = N x N matrix where B_{ij} = number of instances when $|R_{s_i} - R_{s_j}| < \varepsilon$ (ε is dependent on the sensitivity of the sensor hardware and is typically chosen between 5 and

20). B_{ij} gives the number of observed instances when sensors s_i and s_j exhibited similar behavior. B is updated whenever the sensors are sampled.

$P_{s_i-s_j} = {}^{Bij}\!/_{Nt}$. $P_{s_i-s_j}$ is the probability that sensors s_i and s_j behaved in a similar manner and depends on the time t at which it is calculated.

W_{s_i} = Probabilistic weight associated with R_{s_i} where,

$$W_{s_i} = \begin{cases} 1 \text{ ; if } s_i \text{ is alive} \\ W_{s_j}P_{s_i-s_j} \text{ ; if } s_i \text{ is dead and its readings are approximated using } R_{s_j} \text{ for some } j \neq i. \end{cases}$$

A_{s_i} = Set of sensors whose readings are being approximated using readings from sensor $_{s_i}$. Initially for all i = 1 to N, $A_{s_i} = \varnothing$.

$VS_{reading}$ denotes a reading from the Basic Virtual Sensor where, $VS_{reading} = (\sum_{s_i \in S} W_{s_i} R_{s_i}) / (\sum_{s_i \in S} W_{s_i})$

We also make the following assumptions:

(1) All the Singleton Virtual Sensors associated with a particular Basic Virtual Sensor represent physical sensors having the same characteristics such as sensitivity, error, range of output values etc.

(2) All Singleton Virtual Sensors are sampled together periodically in a synchronized fashion.

(3) All Singleton Virtual Sensors are alive at startup and hence, initially $W_{s_i} = 1$ for i = 1 to N.

Suppose S' is the set of Singleton Virtual sensors that fail at time t then the compensation algorithm is given as follows:

(1) Set $N_C = N_C - |S'|$.

(2) For each sensor $s_x \in S'$, compute the following:

(i) Calculate $P_{s_x} = \max \{\{0\} \cup \{P_{s_x-s_j} \text{ for all } s_j \in S - S' - A_{s_x} \mid P_{s_x-s_j} > P_T\}\}$ where P_T = minimum user-defined threshold probability. P_T ensures that the maximum observed probability of similar behavior is high enough not to be attributed to chance. Note that in case none of the probabilities exceed P_T, then $P_{s_x} = 0$. Suppose $P_{s_x} = P_{s_x-s_y}$. This implies that sensor $s_y \in S$ behaved most similar to s_x.

(ii) Set $W_{s_x} = W_{s_y}P_{s_x}$ and $A_{s_y} = A_{s_y} \cup \{s_x\}$. Hence, from now onwards, readings for sensor s_x will be approximated using readings from sensor s_y, i.e. $R_{s_x}=R_{s_y}$. We assume that the equations for calculating W_{s_x} and R_{s_x} are stored in the framework. This ensures that: (a) Subsequent to each sampling, sensor readings for s_x are generated using data obtained from sensor s_y and; (b) Whenever W_{s_y} changes (for example, when sensor s_y fails), W_{s_x} also gets updated.

(3) Update W_s for all $s \in A_{s_x}$ and set $S_F = S_F \cup \{s_x\}$.

(4) If $W_{s'} < P_T$, for any $s' \in S$, set $W_{s'} = 0$ & $S = S - \{s'\}$.

This compensation algorithm ensures that during computation of $VS_{reading}$ the absence of sensors is compensated by readings obtained from other live sensors thereby increasing availability of sensor data.

Each $VS_{reading}$ is associated with a Data Quality Indicator (DQI) which gives an indication of the reliability of the data. For the sake of simplicity, we assume that all live sensors give accurate readings and the only inaccuracies in $VS_{reading}$ are due to approximation of readings for dead sensors using the compensation algorithm.

$$DQI = 100 * (N_C + \sum_{s \in S_F} g(t - T_{start})W_s) / N; \quad DQI \in [0, 100]$$

If the value of DQI falls below DQ_T then this implies that the Basic Virtual Sensor is unable to meet the minimum data quality requirements set by the user. Whenever a Basic Virtual Sensor detects that its DQI falls below DQ_T it automatically goes offline and its service disappears.

$g(x) = 1 - e^{-(x/a)}$, where a is a constant greater than 0 whose value depends on the sensors and their deployment environment. It is the approximate number of seconds it takes the sensor network to operate in steady state. For example, if the sensor network takes 1 month to reach a steady state of operation and time is being measured in minutes then, $a \sim 45000$. g(x) is the weight function associated with W_s. If W_s is calculated based on observations taken over a time interval whose length is close to the amount of time taken by the system to reach a steady state of operation, then that value of W_s will be given more weight than say, a value of W_s which has been calculated based on observations taken over a relatively short period of time. Note, $g(x) \in [0, 1)$ and $\lim_{x \to \infty} g(x) = 1$.

4.4.2 Derived Virtual Sensor

Derived Virtual Sensors currently do not have the provision for a compensation algorithm to increase their availability and they go offline if any of the Derived or Basic Virtual Sensors associated with them, fails.

The Data Quality Indicator associated with readings originating from a Derived Virtual Sensor is calculated as the average of the Data Quality Indicators obtained from all the Derived or Basic Virtual Sensors associated with it.

5. Comparison Analysis

We consider a scenario where a Basic Virtual Sensor consisting of 7 Singleton Virtual sensors $\{s_i\}_{i=1 \text{ to } 7}$ is started up at time T_{start}. We assume that $DQ_T = 70$, $P_T = 0.5$, a=1 and for the sake of simplicity statically set $P_{s_i-s_j} = p$ for all $i \neq j$ where $i,j \in [1,7]$ and p = 0.0, 0.6 and 0.8. The different values of p represent the different degrees of similarity in behavior amongst the sensors, which in practical circumstances would depend on factors such as pattern of sensor deployment and the environment where they are operating. We also assume that the physical sensors (and hence the Singleton Virtual Sensors representing them) start failing individually starting at T_{start}+10 minutes. If a sensor fails at time t, then as per our assumptions, $t - T_{start} \geq 10$ and hence, $g(t - T_{start}) \sim 0.99$ for purposes of calculating data quality using the DQI

formula. The graph in Fig. 3 is a plot of DQI values calculated for different values of p as the physical sensors keep failing one-by-one over time. The plot for p = 0 corresponds to the case where the compensation algorithm is not applied on failure of a sensor. From the graph one can observe that application of compensation algorithm significantly increases the availability and reliability of sensor data in the face of sensor failure, even for a small number of sensors. In this particular scenario, for example, we observe that there is an average gain of 30% in reliability and a gain of 100% in availability when we compare the case where p = 0.8 to the case where no compensation is applied (p = 0).

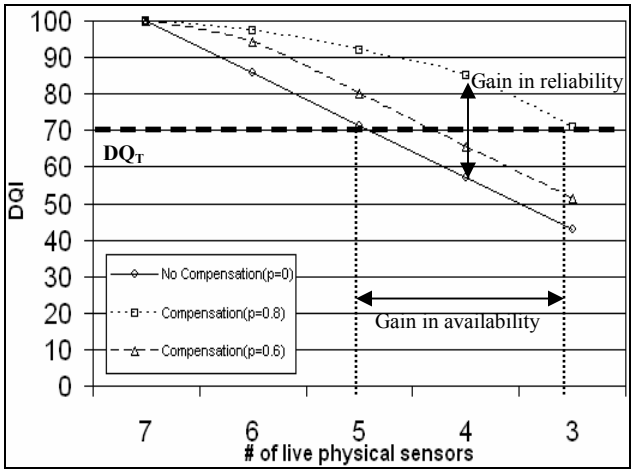

Fig. 3 Comparison between different levels of compensation

6. Implementation

We implemented the Virtual Sensor framework on top of the Atlas Platform which is a commercially available plug-and-play service oriented sensor platform. Atlas automatically exports sensors and actuators deployed in a physical space as OSGi software services. The virtual sensor framework consists of a collection of OSGi bundles. Singleton, Basic and Derived types of virtual sensors are implemented as OSGi bundles and represented as services inside the OSGi environment. The knowledge base and framework controller are also implemented as individual OSGi bundles and their services can be accessed by both user applications and virtual sensor services. The knowledge base service provides mechanisms for downloading and updating the virtual sensor model definitions, phenomena definitions and reliability and availability parameters. The framework controller manages the creation and deletion of virtual sensors. Using the service discovery and creation mechanisms available through OSGi, the framework controller detects when a new physical sensor joins the network and is able to dynamically create and deploy a Singleton Virtual Sensor service representing that sensor. On receiving a user or application query, the framework controller accesses the knowledge base and creates and deploys the necessary Basic and Derived Virtual sensor services required to fulfil the query.

7. Conclusion & Future Work

In this paper, we proposed a framework model for deploying virtual sensors in service-oriented sensor networks with the aim of enhancing reliability, prolonging availability and enabling automatic service composability. We classified virtual sensors into singleton, basic and derived categories and gave models for each of them. We further proposed ideas to enable the automatic composition of virtual sensor services for detecting more complex phenomena. We also described a compensation algorithm for coping with sensor failure and a Data Quality Indicator for expressing the quality of sensor readings.

Our on-going research includes the refinement of the framework model, construction of sensor failure models and factoring in the effects of deployment patterns of sensors on reliability and availability. We also plan on conducting both large scale simulations and smaller scale experiments using a real world test bed of sensors to study the performance quality of virtual sensors.

References

[1] D. Culler, D. Estrin, M. Srivastava, Overview of Sensor Networks, *IEEE Computer*, Aug. 2004.

[2] J. King, et. al,. "Atlas – A Service-Oriented Sensor Platform", Proceedings of the first IEEE International Workshop on Practical Issues in Building Sensor Network Applications , Tampa, Nov. 2006

[3] R. Bose, et. al., "Building Plug-and-Play Smart Homes Using the Atlas Platform,",The 4th Int'l Conf. on Smart Homes & Health Telematics, Belfast, 2006.

[4] J.S.Y. Chin et. al, Virtual Appliances for Pervasive Computing: A Deconstructionist, Ontology based, Programming-by-example Approach, The 2005 IEE International Workshop on Intelligent Environments.

[5] T. Gu, H. K. Pung, D. Q. Zhang, Towards an OSGi-based Infrastructure for Context-aware Applications, Pervasive Computing, Oct.-Dec., 2004.

[6] S. Kabadayi, A. Pridgen, C. Julien, Virtual Sensors: Abstracting Data from Physical Sensors, *Proceedings of the 2006 International Symposium on on World of Wireless, Mobile and Multimedia Networks*, Buffalo, NY, 2006.

[7] R. Costantini and S. Susstrunk, Virtual Sensor Design, *Proceedings of SPIE -- Sensors and Camera Systems for Scientific, Industrial, and Digital Photography Applications V*, San Jose, CA, 2004.

[8] T. W. Long, E. L. Hanzevack, W. L. Bynum, Sensor Fusion and Failure Detection Using Virtual Sensors, *Proceedings of the American Control Conference*, San Diego, June 1999.

[9] N. Xiong and P. Svensson, Multi-sensor management for information fusion: issues and approaches, Information Fusion, Vol. 3, 2002.

[10]N. Hardy and A. A. Maroof, ViSIAr, A Virtual sensor integration architecture, *Robotica 1999*, Vol. 17.

[11]OSGi Alliance – http://www.osgi.org