e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 05 Issue 12
April 2018

AN EFFICIENT AND RELIABLE COMMUNICATION IN WIRELESS BODY AREA NETWORK WITH THE CONCERN OF ENERGY HARVESTING

Manimegalai .P [a&c*], Pasupathy.N [b]

[a] Ph.D Research Scholar, Dept. of Electronics, Erode Arts and Science College, Erode, India.

[b] Associate Professor, Dept. of Electronics, Erode Arts and Science College, Erode, India.

[c] Asst. Professor, Dept. of Electronics, Sri Vasavi College, Erode, India.

*** Email id:** *manimegalaiap@gmail.com*

Abstract:

Wireless Body Area Networks (WBANs) are expected to achieve high reliable communications among a large number of sensors. The outage probability can be used to measure the reliability of the WBAN. In this paper, we optimize the outage probability with the harvested energy as constraints. Firstly, the optimal transmit power of the sensor is obtainted while considering a single link between an access point (AP) located on the waist and a sensor attached on the wrist over the Rayleigh fading channel. Secondly, an optimization problem is formed to minimize the outage probability. Finally, we convert the non-convex optimization problem into convex solved by the Lagrange multiplier method. Simulations show that the optimization problem is solvable. The outage probability is optimized by performing power allocation at the sensor. And our proposed algorithm achieves minimizing the outage probability when the sensor uses energy harvesting. We also demonstrate that the average outage probability is reduced with the increase of the harvested energy.

Keywords: wireless body area networks (WBANs); outage probability; energy harvesting; reliability

Introduction : Body Area Sensor Networks (BASNs) have been developed for a variety of applications, such as health and patient monitoring, medical care, tones and entertainment. Application-wise BASNs are maturing with an array of applications and innovations. However, mass deployment is hindered by a number of factors. First, most BASNs are special purpose designs with fixed applications, which implies that BASNs and their applications are tightly coupled in the development phase. Consequently, application developers need to be aware of these specifies, which leads to application awareness of details of data representation, sensor types and the likes. Second, the meaning of the data collected by the sensors and any prior knowledge essential for interpretation is not recorded, which renders the data useless for future processing . Prior knowledge may refer, for example, to driver details of the applied sensor, to details about how to convert voltage into a blood pressure value, or to assumptions about the influence of the operating environment on sensor accuracy. Third, many applications are currently closed, vendor specific and specialized services. The applications cannot access each other's information and in many cases, the data acquired is not even accessible to the users. For example, data from the movement sensor of the Philips Direct Life service that monitors people's physical activity levels are currently not available to other services. Another example is that a BASN may need to interoperate with other BASNs, backend services and even with electronic health record systems., we proposed a generic BASN platform named VITRUVIUS (see Figure 1), which integrates multiple sensors and handles different sensor configurations, allows BASN applications to be installed dynamically and run concurrently. A benefit of a platform like VITRUVIUS is that it provides capabilities for reuse and evolution of existing sensor net- works and applications, and for extensions by adding new sensors and applications. These capabilities bring benefits such as lower costs relating to application development and maintenance, a higher degree of comfort since the number of mounted devices remains limited (by reusing existing components and sensors), and a higher dependability because of shared platform evolution and the ability of sharing data and resources. For example, in the healthcare domain, an application that detects atria fibrillation and an application that detects epileptic seizures can use the same sensor for collecting ECG data. Interoperability among components of distributed systems refers to the ability to exchange services and data with one another . While interoperability is concerned with syntax and parameters of interfaces, semantic interoperability addresses the meaning of operations and exchanged data. It reduces dependencies, which is important for third parties to make system components that can be integrated into the platform easily. Semantic interoperability is, therefore, a necessary condition for flexibility and evolution of BASNs. We use the VITRUVIUS platform to analyze how semantic interoperability fits in the architecture and how to work effectively with semantics. Semantic interoperability implies that data is given a context for interpretation, which is either known by the receiver or part of the data. As it turns out such context has typically a local relevance. The architecture should support this locality to avoid burdening the entire system with irrelevant details. In this paper, we present the analysis, design, reference implementation and qualitative evaluation of a semantic layer added to VITRUVIUS with the purpose to improve the semantic interoperability among sensor networks and applications. We adopt the ontology based approach , in which the developed ontology on context aware examinations has the potential of not only supporting correct interpretation of sensor data but also ensuring its appropriate use in accordance with the purpose of a given application. The ontology can be regarded as a vehicle that unifies the data originating from different system components into a universal understanding, thereby achieving interoperability at the semantic level. We focus on the analysis and the resolution of problems regarding the architectural design and implementation of the application ontology, data storage and data sharing. Furthermore, we evaluate the BASN platform through aspects of semantic expressiveness, extensibility and data sharing. The platform is implemented on an Android phone connected to multiple sensor devices (Shimmer sensors and a DTI-2 wristband). The paper is organized as follows. Section 2 summarizes the existing VITRUVIUS platform and presents the ontology-based architecture. Section 3 gives our reference design and implementation of the platform, application ontology and data sharing. -

What's VITRUVIUS?

Versatile Interface for Trustworthy VItal User (oriented) Services

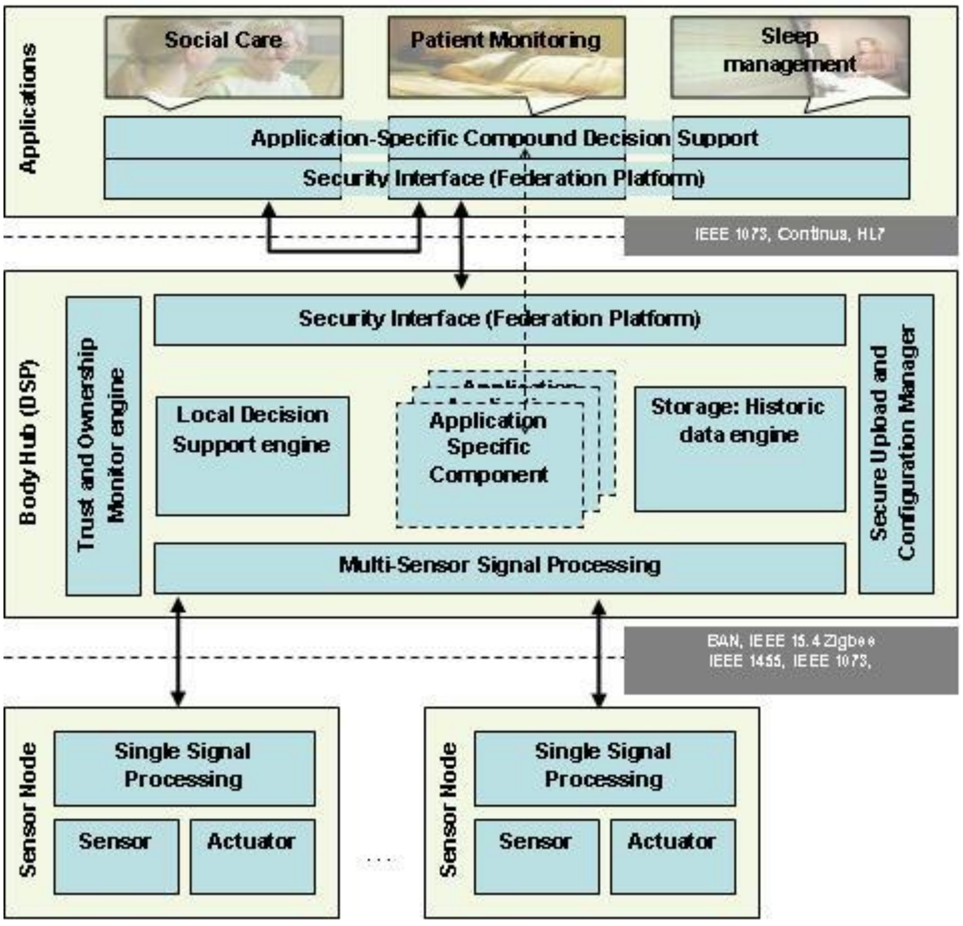
In future, detailed and fully up to date personal information, including data from real time body sensors, will be accessible for widespread use by multiple services and applications. This information can come from sources as diverse as certified medical exams carried out by qualified professionals, or mass market on body accelerometers casually used for computer gaming. The use of body sensor data can start early in people’s life, when baby phones measure the emotional state to gently guide the newborn into a peaceful sleep. Late in people’s life, body sensing can play an important role in assisting seniors to age in dignity in a safe natural home environment.

During the decades in between, sensor data can be used in applications ranging from sport coaching to intensive computer gaming and medical preferably ambulatory monitoring during periods of illness, or chronic need of attention. Yet the widespread use of body data, not only including people’s physiology but also their emotions, raises societal issues. It also comes with interesting scientific challenges, particularly because the technical constraints of node power consumption and wireless links and networks have to be taken into account. The VITRUVIUS project aims at exploring the underlying key consequences for the architecture of Body Sensor Networks (BSN) and the handling of information about the individual's body coming from power-constrained wireless sensor nodes.

The handling of medical information about the human is migrating from a hospital centric to a patient centric approach. Body data is not only used in healthcare but also in lifestyle and entertainment services, where the person centric thinking already is pervasive. Ownership and exercising access rights of personal privacy sensitive data is a delicate question, and has to be studied at architecture level, in the context of severe limitations in ultra low power sensors nodes with unreliable and bandwidth-limited sensor modules.

VITRUVIUS System Architecture

At the heart of the project is the architecture that provides essential body information to service providers in an accurate, reliable, robust and trusted privacy preserving manner. It relieves the application developer from the need to understand the details of the underlying sensor system.


Applications
Social Care
Patient Monitoring
Sleep management
Application-Specific Compound Decision Support
Security Interface (Federation Platform)
IEEE 1073, Continua, HL7
Body Hub (DSP)
Trust and Ownership Monitor engine
Security Interface (Federation Platform)
Local Decision Support engine
Application Specific Component
Storage: Historic data engine
Secure Upload and Configuration Manager
Multi-Sensor Signal Processing
BAN, IEEE 15.4 Zigbee
IEEE 1455, IEEE 1073,
Sensor Node
Single Signal Processing
Sensor
Actuator
Sensor Node
Single Signal Processing
Sensor
Actuator

VITRUVIUS System Architecture

At or near the body, a number of sensor nodes extract information from the body via dedicated sensors, sensor single signal processing modules which pre-process the data (through calibration, artifact rejection, signal validation and compression), and wireless (e.g. Zigbee) links that convey the information securely to a body hub. The sensor nodes may also contain actuators so that the interface between body hub and sensor nodes will, in general, be bidirectional.

Based on the sensor signals, the body hub estimates key physiological parameters (e.g. heart rate, velocity, temperature) by means of multi sensor signal processing, and subsequently extracts key diagnostic information by means of a hub local decision support engine. This processing is controlled by application specific components uploaded into the hub. A wireless PDA can serve as hub node, interacting with the user (e.g. perceived pain level, system feedback, privacy policy settings, authenticating trusted sensors, etc).

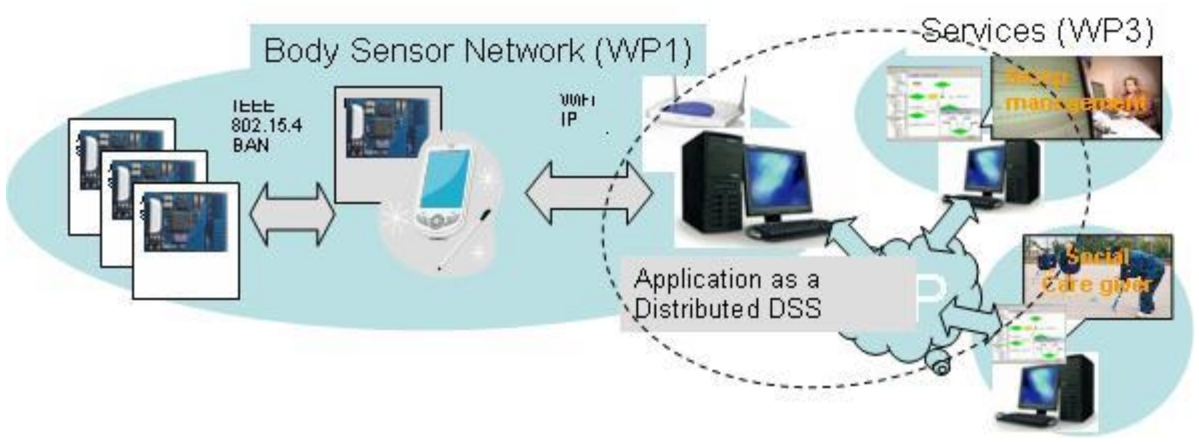
Diagnostic information is conveyed by a wireless link, preferably using existing data formats such as the Zigbee Medical Profile to the 'fixed world' where it is integrated with other information (e.g. medication status, electronic patient record, 'fixed world' sensors) into a compound decision support scheme that forms the heart of the application, and that is connected to the various caregivers. Again, the information that is send out is typically controlled by uploaded components .

Since the sensor nodes and body hub will, in general, be battery operated, it is essential to minimize the amount of information that is conveyed wirelessly, even if this requires many signal-processing and decision-support computations. This is so because Moore's law causes these computations to become ever more energy efficient, whereas wireless links are governed by fixed energy limits. A direct implication is that signal processing and decision support functions, which both act to reduce the amount of data, should be shifted as deeply as possible into the sensor nodes and hub. This efficiency consideration is perfectly consistent with our view of the user as being the owner of his/her body information, with no more information released to the outside world than strictly needed.

Signal processing algorithms, but even more so, decision support schemes are specific to an application or service, and hence need to be configurable and up loadable in a secure and trust preserving manner. To this end, the hub architecture contains a secure upload and configuration manager, trust and ownership monitor, and security interface. The first module provides means for run time upload and installation of application specific components. For instance, a service may want to use its own local decision support component specific only to this application, to get access to specifically tailored information. In this case, the application or service can request installation of the component on the hub. The secure upload and configuration manager checks the component certification details, verifies the future system integrity and installs it on a system. The trust and ownership monitor not only constantly examines the current security, performance and reliability properties of the system, but also predicts and verifies these properties at the system reconfiguration time, thereby authorizing configuration changes. The Security interface serves as a “body firewall” that shields the hub from the outside world, blocks body spy ware, and provides only limited privacy sensitive data to parties that are not entitled full access. For example, the user or his social care givers receive different information than doctors or professional sport coaches. Some parties may only be interested in whether medically certified products are used according to professional protocols (liability), others may be interested in physiological data (Sp02 saturation, respiration details), while others are best served with interpreted data (grandma has fallen). If necessary the hub can make the data “anonymous” for certain applications. For instance, heart rhythm and blood pressure data should, in case of monitoring a cardiac patient, be routed to a health monitoring application to be supervised by a hospital cardiologist. The same signals, however, can be used for sports or fitness monitoring and coaching. So, different applications, using data of the same sensors, (can) have different supervisors and/or service providers.

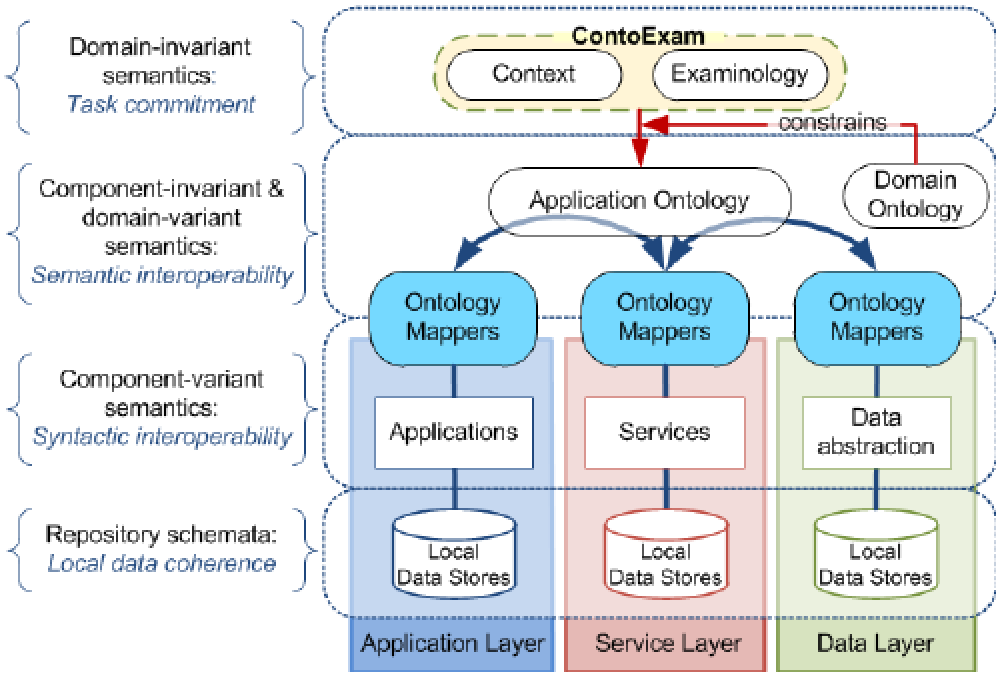
Within the hub, the raw incoming signals as well as the temporal evolution of the hub's state can be collected in a storage memory for inspection in special cases (e.g. liability disputes), or at special moments (e.g. when the hub is close to the fixed-world transceiver, permitting wireless communication at very low power).

To facilitate application development, it is essential that the medical or domain specialists can take the helm in the development of the decision-support scheme, without requiring knowledge engineers as go-betweens. This implies that an ontology based approach is needed, where specialists can freely define the ontologies that are relevant to the application at hand, along with guidelines and decision rules in terms of these ontologies. For this reason a state of the art ontology based decision support development framework will be taken as a starting point, and a separate back end will be developed to split the resulting decision support rules into executable 'fixed world's and 'hub based' modules. In this manner, service providers can quickly develop new applications and services based on high level body information, without requiring knowledge about specific sensor technology.



Ontology-based architecture

Crucial to the success of Body Area Sensor Networks is the flexibility with which stakeholders can share, extend and adapt the system with respect to sensors, data and functionality. The first step is to develop an interoperable platform with explicit interfaces, which takes care of common management tasks. Beyond that, interoperability is defined by semantics. This paper presents the analysis, design, implementation and evaluation of a semantic layer within an existing BASN platform for the purpose of improving the semantic interoperability among sensor networks and applications. We adopt an ontology-based approach but rather than having a single overall ontology, we find that using clear semantic domains and mappings between them improves composability and reduces interoperability problems. We discuss the design choices and a reference implementation on an Android phone and actual sensor devices. We show by a qualitative evaluation that this semantic interoperability indeed provides significant improvements in flexibility. Body Area Sensor Networks (BASNs) have been developed for a variety of applications [4, 12], such as health and patient monitoring, medical care, fitness and entertainment. Application-wise BASNs are maturing with an array of applications and innovations. However, mass deployment is hindered by a number of factors. First, most BASNs are special purpose designs with fixed



System architecture of the VITRUVIUS Body Area Sensor Network.

applications, which implies that BASNs and their applications are tightly coupled in the development phase. Consequently, application developers need to be aware of these specifics, which leads to application awareness of details of data representation, sensor types and the likes. Second, the meaning of the data collected by the sensors and any prior knowledge essential for interpretation is not recorded, which renders the data useless for future processing [13]. Prior knowledge may refer, for example, to driver details of the applied sensor, to details about how to convert voltage into a blood pressure value, or to assumptions about the influence of the operating environment on sensor accuracy. Third, many applications are currently closed, vendor-specific and specialized services. The applications cannot access each other's information and in many cases, the data acquired is not even accessible to the users. For example, data from the movement sensor of the Philips Direct Life service that monitors people's physical activity levels are currently not available to other services. Another example is that a BASN may need to interoperate with other BASNs, backend services and even with electronic health record systems. In [3], we proposed a generic BASN platform named VITRUVIUS (see Figure 1), which integrates multiple sensors and handles different sensor configurations, allows BASN applications to be installed dynamically and run concurrently. A benefit of a platform like VITRUVIUS is that it provides capabilities for reuse and evolution of existing sensor networks and applications, and for extensions by adding new sensors and applications. These capabilities bring benefits such as lower costs relating to application development and maintenance, a higher degree of comfort since the number of mounted devices remains limited (by reusing existing components and sensors), and a higher dependability because of shared platform evolution and the ability of sharing data and resources. For example, in the healthcare domain, an application that detects atrial fibrillation and an application that detects epileptic seizures can use the same sensor for collecting ECG data. Interoperability among components of distributed systems refers to the ability to exchange services and data with one another [9]. While interoperability is concerned with syntax and parameters of interfaces, semantic interoperability addresses the meaning of operations and exchanged data. It reduces dependencies, which is important for third parties to make system components that can be integrated into the platform easily. Semantic interoperability is, therefore, a necessary condition for flexibility and evolution of BASNs. We use the VITRUVIUS platform to analyze how semantic interoperability fits in the architecture and how to work effectively with semantics. Semantic interoperability implies that data is given a context for interpretation, which is either known by the receiver or part of the data. As it turns out such context has typically a local relevance. The architecture should support this locality to avoid burdening the entire system with irrelevant details. In this paper, we present the analysis, design, reference implementation and qualitative evaluation of a semantic layer added to VITRUVIUS with the purpose to improve the semantic interoperability among sensor networks and applications. We adopt the ontology based approach of [2], in which the developed ontology on context-aware examinations has the potential of not only supporting correct interpretation of sensor data but also ensuring its appropriate use in accordance with the purpose of a given application. The ontology can be regarded as a vehicle that unifies the data originating from different system components into a universal understanding, thereby achieving interoperability at the semantic level. We focus on the analysis and the resolution of problems regarding the architectural design and implementation of the application ontology, data storage and data sharing. Furthermore, we evaluate the BASN platform through aspects of semantic expressiveness, extensibility and data sharing. The platform is implemented on an Android phone connected to multiple sensor devices (Shimmer sensors and a DTI-2 wristband). The paper is organized as follows. Section 2 summarizes the existing VITRUVIUS platform and presents the ontology-based architecture. Section 3 gives our reference design and implementation of the platform, application ontology and data sharing. The evaluation is given in Section 4. Section 5 concludes the paper. Figure 1 shows the architecture of the VITRUVIUS system that was proposed in [3]. The system consists of body sensors together with a more powerful device (body hub) that is equipped with VITRUVIUS software platform. This hub is capable of storing data and running services and applications; it controls the BASN and acts as a single access point. In our reference design, we use a smart phone. We regard VITRUVIUS as a platform on which (concurrent) BASN applications developed by third parties are dynamically installed and configured and to which sensors developed by third parties can be connected. The platform is self-contained and can connect to back end systems for the purpose of app installation, data storage and analysis, monitoring and control in back end applications. A modular and layered architecture of the platform is proposed that provides a loose coupling between the applications and the detailed characteristics (e.g. the communication protocol, sensor value representation) of the sensors. Loose coupling can be achieved by separating the architecture into layers, each of them compounding certain concerns. We have applied three functional layers: data and sensor concerns, service concerns and application concerns, as follows. The Sensor abstraction layer (SAL) is an interface between the sensors and the other modules in upper layers of the body hub. This layer allows the system to easily adapt and handle new sensors and their configurations. The layer contains sensor drivers that drive and configure a sensor (e.g. enable/disable sensors or set the sampling rate) to obtain data according to a request from the service or the application. The Service layer contains the downloaded services and the basic services of the platform. A central database is used for the purposes of data storage and sharing among system components. A service is a program that operates only on the data space defined by the user of which the local BASN storage (e.g. a database) is part of. A service typically operates in the background without a user interface. For example, a service derives heart rates from raw ECG signals, or it controls a sensor to obtain data. The Application layer contains the downloaded applications and the GUIs. An application combines the results of services to achieve its functionality. Several applications can share the same service. An application interfaces with the outside world, at least through a user interface but possibly also by communication with other parties (e.g. back ends) than the body hub. For example, an application that facilitates the user to monitor his daily health conditions and a fitness application share the data generated by the heart- rate detection service. Both may communicate statistics to an external party. Although this results in a loose coupling in terms of functionality and system components, from a semantic point of view such an architecture still enforces a strict coupling between data schema, syntax and meaning, i.e., one big semantic silo. It is our objective to provide loose coupling also at the semantics level, facilitating true open sensor networks and apps that are truly interoperable. To that end we introduce the following ontology-based architecture. In order to provide loose coupling at the semantic level, first a separation needs to be achieved between semantic and functional concerns. Ontologies are widely recognized as a means to address semantic concerns. Stated in [8], an ontology is a formal, explicit specification of a shared conceptualization. By applying ontologies, i) concepts will become semantically grounded in the referred real-world entities, ii) by reasoning over these concepts, new facts can be inferred or validity of statements can be checked, and iii) by relating the data schemata to an ontology, stored data are enriched with the semantic conventions that apply in the domain of application. When integrating ontology's in the existing architecture of Figure 1, a foundation is laid for separating semantic from functional concerns. A prerequisite, however, is to fit those ontology's without disrupting the existing architecture. We therefore engage in an approach that addresses semantic concerns in a semantic architecture that is orthogonal to the existing (functional) architecture that already shows loose coupling between (functional) components. We further observe that not all semantic concepts that are available in the ontologies are relevant for all functional layers and, in addition, it pays in terms of efficiency to lump semantic closely related concerns within a semantic layer together. This leads to localized semantics that is translated or mapped at boundaries. We thus adopt an ontology based approach, shown in Figure 2, in which the semantic architecture, depicted as 4 horizontal layers, is orthogonal to the functional architecture that is depicted as 3 vertical columns. We can now focus on achieving semantic loose coupling without having to fear to destroy functional loose coupling, and do so by introducing two architectural principles that are fundamental to loose coupling, separation of concerns and transparency, and apply these at that semantic architecture. In software engineering, separation of concerns (SoC) is a design principle that refers to decomposing a software system into parts with as few dependencies as possible [11]. In [2], it is extended and specialized into two separate aspects, the functional SoC and the semantic SoC. As shown in Figure 2, the functional SoC is represented by three vertical layers and the semantic SoC is represented by four horizontal layers. The functional SoC takes care of the functional design of the system, while the semantic SoC simplifies achieving semantic interoperability since it allows separating the semantics into layers with different levels of abstraction. The semantic interoperability then can be achieved at a higher abstraction layer, where less heterogeneous concepts exist than at lower abstraction layers. The Repository schemata layer carries ample semantics, explicitly that is. Computers store data, just data, and there exists no way around that. Hence, semantics at this level are implicitly captured in data and meta-data, and to operate appropriately on that data requires prior knowledge. The inherent semantics is fully settled in their design and development phase. However, we observe that each component in any functional layer will have its own distinct semantics about its own particular data and data format, possibly even about their administration and access, which is relevant for that components operation and purpose.

Data storage and sharing

A common way to work with ontology data is to load the ontology components (e.g. classes, instances) at run-time from a set of source documents (e.g. OWL les). This is a very exible approach, but it has limitations for efficiently processing large amounts of instance data since the application must parse the source documents each time it is run. Another approach is to store the ontology data persistently in a database. This saves the loading overhead and means that we can store large volumes of ontology data, but it has the expense of a higher overhead to retrieve and update data. There are several approaches to store the ontology data.

We consider two architectural approaches for the purpose of data storage and sharing. In the Ist approach, a centralized ontology database is used to store both ontology based data (e.g. sensor data) and their respective meanings (ontologies). This approach is similar to the architecture in [3], albeit a more formal application ontology is employed. The use of the centralized database may facilitate the data access control and management, but it has some issues. Using a single schema impedes loose-coupling between applications and services, whilst the approach that accepts local schemata indeed requires mappers but provides a loosely coupled architecture. The database could become a performance bottleneck of the platform when there are too many modifications to the database. Extra effort is needed in developing the ontology database. Moreover, the procedure of updating ontology's is complex.

To address the above issues, an alternative approach of using decentralized data stores is proposed in Figure 5. This architecture conforms to the layered architecture of the existing platform. The grey box represents the components of the Basic Platform, which provide necessary services for other components (e.g. applications) running on the platform. For example, the configuration Manager is responsible for handling system configuration and installing applications, the Security Interface and the Trust & Ownership Manager are responsible for authenticating the downloaded software and evaluating the system trustworthiness.

In comparison to the former architectural approach, the ontologies are stored separately in the application ontology repository. This has advantages: i) a good performance of loading, retrieving, and updating data since the data storage is more lightweight; ii) no extra implementation e orts are needed; and iii) exibility of updating and modifying ontologies. Besides, each component (sensor driver, service or application) uses its local data store (not shown) for storing data. This avoids the issue of performance bottleneck. The medication of the data schema can be easily performed and authorized by the data store owner. Also, it simples the reuse of existing applications and services since they can maintain their original data storage mechanism (e.g. relational database or le system). In the former architecture, when a new application is installed, its data schema and tables need to be created in the centralized database.

This approach, however, has practical issues that we need to address: 1) how can data in local data stores be shared? and 2) how does a component know which component it contacts for a particular type of data? For example, the heart monitoring application needs to do a proper service that can provide heart rate data.

To address the Ist question, we use the publish subscribe pattern and message based communication. The publish subscribe pattern provides a loosely coupled form of interaction among system components. The publisher has useful data or events, on which interested subscribers can subscribe in order to get data or stay synchronized with the publisher. A publisher can share data to multiple subscribers by maintaining a list of subscribers for a certain subject. A subscriber can also subscribe to data from multiple publishers. For example, the epilepsy monitoring application can subscribe to data from both the accelerometer sensor driver and the heart rate detection service. The subscriber that wants to receive data from a publisher needs to have a permission granted by the platform. Moreover, the subscriber can implement alter method that accepts only interested data sent by authenticated publishers.

To address the second question, we apply the broker pattern, in which the broker mediates communication (e.g. data requests and responses) among the system components. The configuration Manager plays a role of the broker that varies and serves the requests from the subscribers (e.g. a data subscription) and the publishers (e.g. a publishing advertisement). Figure 6 shows the interaction among the publisher (service), the subscriber (application) and the configuration Manager. The verification process is performed with the help of the Security and Trust Manager (not shown). When a new publisher is added to the platform, the publisher advertises its metadata with the configuration Manager. If the publisher is varied, its information (e.g. publishing data) is stored into the application ontology. When a subscriber wants to subscribe to a particular data, it sends a request to the configuration Manager, which then queries the application ontology for the matching publisher of such request. Upon the response from the configuration Manager, the subscriber communicates directly to the publisher for the data subscription. The data messages exchanged between the publisher and the subscriber are handled by their ontology mappers (not shown), which translate between the local data format and the domain data format (from the syntactic perspective) and between the local semantics and the domain semantics (from the semantic perspective).

In comparison to the former architectural approach, the ontology's are stored separately in the application ontology repository. This has advantages: i) a good performance of loading, retrieving, and updating data since the data storage is more lightweight; ii) no extra implementation efforts are needed; and iii) edibility of updating and modifying ontology's. Besides, each component (sensor driver, service or application) uses its local data store (not shown) for storing data. This avoids the issue of performance bottleneck. The modification of the data schema can be easily performed and authorized by the data store owner. Also, it simplifies the reuse of existing applications and services since they can maintain their original data storage mechanism (e.g. relational database or le system). In the former architecture, when a new application is installed, its data schema and tables need to be created in the centralized database.

This approach, however, has practical issues that we need to address: 1) how can data in local data stores be shared? and 2) how does a component know which component it contacts for a particular type of data? For example, the heart monitoring application needs to do a proper service that can provide heart rate data.

IMPLEMENTATION:

Platform configuration

An example of the platform configuration consisting of the Android phone (body hub) and three sensor devices: the Shimmer ECG worn on the chest, the Shimmer accelerometer worn on the right arm and the DTI-2 wrist band worn on the left arm. We chose the Android platform for implementing the body hub since Android is an open source mobile operating system used by many devices on the market. The Android smart phone provides a wide range of communication protocols, e.g. WLAN and Blue-tooth (or Bluetooth LE). The smart phone also provides sensors, e.g. accelerometer, gyroscope and GPS. The Shimmer and DTI-2 devices are chosen since they are wearable and configurable, support Bluetooth communication and provide several types of sensor data. The DTI-2 wristband integrates the accelerometer, skin conductance, skin temperature, environment temperature and light sensors.

The Basic Platform, which provides the services for other components and the GUI, can be downloaded and installed as a regular Android app. The GUI allows the users to manage their BASN, such as adding and configuration a new sensor or installing a new application. The sensor driver, service and application can also be installed as regular apps.

Ontology mapper and message

The mapping between the application ontology and the local ontology is performed by the map per component, which also provides interfaces (APIs) for the data communication and control (e.g. sensor configuration). The map per is implemented as an Android service, which can quickly be implemented using a generic development pattern. Figure 8 shows the simplified class diagram of the sensor mapper and the sensor driver. The Sensor Mapper class is extended from the Abstract Service class. The send(Message) and on Receive Message methods implement the message-based communication. The local To Domain and domain To Local methods are used to translate between the local and domain ontology's. The Sensor Mapper calls the methods provided by the Sensor Driver for the purpose of streaming data or configuring the sensor. The mappers of the service and the application are implemented similarly.

The message based communication is implemented based on two message classes provided by the Android platform, including the Intent (android.content.Intent) and the Message (android.os.Message). The Intent object is a passive data structure, which can be used to activate Android activities, services and broadcast receivers. The Intent messaging is a facility for late run time binding between components in the same or different applications. For example, the Basic Platform can use the Intent messaging to launch or to communicate with a sensor driver or a data processing service. The Message object is used for other purposes, such as the advertisement of a data publisher, the configuration of a sensor and the data sharing.

QUALITATIVE EVALUATION

We adopted the ontology-based approach to support the semantic interoperability of the platform. Instead of using the database schema as in [3], the OWL DL ontology language is used to implement the application ontology. The high expressiveness of the ontology language provides a more accurate and precise representation of the domain knowledge, which limits the ambiguity during knowledge sharing. According to a continuum of kinds of ontology's [16], the very lightweight ones (at the extreme) may consist of terms only, with little or no specification of the meaning of terms. As we move along the continuum, the amount of meaning specified and the degree of formality increase (OWL DL is a kind of Description Logic); the support for automated reasoning also increases. Although the SQL engine of the relational database can be used to perform reasoning, it is highly specialized and tuned for answering queries and ensuring data integrity. The fundamental role of a reasoning engine for ontology's is, however, to derive new information via automated inference. For example, we need the Measure model to be the composition of Input quant, Out-put quant and Measure function. An ontology can derive that a new instance, which is identified to have these three parts, can be automatically classified to be a Measure model.

Extensibility

As mentioned, the semantic interoperability indicates that the platform is easily extended to add a new type of sensor or application. A successful addition of a sensor means that it should be properly configured and used by the platform. The platform allows adding a new sensor by installing its driver as a regular app. The ontology mapper of the sensor driver simplifies the development and integration processes. With a new sensor driver, only the mapper needs to be developed to map the local ontology of the driver to the application ontology and vice versa. The mapper pattern can be used to make the development even simpler.

Above Figure shows the three sensor devices (namely ECG, Left Arm and Right Arm) that are connected to the platform. The Bluetooth addresses of these devices are shown on the right column. When a new device is connected, its configuration properties (e.g. available sensors, sampling rate) are advertised to the configuration Manager and stored into the application ontology. Depending on the con-text, an application may request a particular configuration. Figure (right) shows the available sensors of the DTI-2 de-vice, but only the Acceleration

Data sharing

We evaluate the capability of data sharing between sensor drivers and an application. The Vitruvius Graph application, which is able to visualize data received from multiple sensor drivers, is then installed on the platform. Assuming that a user wants to visualize acceleration data, the application then sends a request to the configuration Manager (the broker) to determine which sensor device can provide the acceleration data. The configuration Manager then queries the application ontology and returns a list of available devices. Figure 10 (left) shows the returned devices on the GUI of the application. The devices include the Left Arm (DTI-2 wristband), the Right Arm (Shimmer accelerometer) and the Device Sensor (phone accelerometer). Assuming that the user selects the Right Arm device, the application then sends a data subscription message to the sensor driver. The application may also send a configuration request (e.g. set the sampling rate) to the driver. The received acceleration data are visualized on the GUI as shown in Figure 1 (right). Similarly, the user then selects the device to visualize the skin conductance data. In this case, only the Left Arm device that provides such data is returned. The skin conductance data are visualized at the bottom of the GUI.

CONCLUSION

We showed that adding an ontology based semantic architecture to an existing platform improves the semantic interoperability among sensor networks and applications. By applying principles of separation of concerns and transparency, loose semantic coupling emerges. Moreover, we have discussed the design choices of the data storage and our reference implementation of the application ontology and the message based communication for data sharing.

By evaluating the platform through aspects of semantic expressiveness, extensibility and data sharing, we showed such platform design indeed provide significant improvements regarding semantic interoperability. By using the ontology mapper, the semantic interoperability can be achieved with minimized mutual dependence. Thus, the process of adding new type of sensor or application are simplified. We have further verified the functions of configuring a sensor and sharing data among sensors and applications. In future work, we want to implement the complete application ontology and evaluate the semantic interoperability with a quantitative evaluation.

References

[1] Kyung Sup Kwak, Sana Ullah, Niamat Ullah, “An overview of IEEE 802.15.6 Standard”, 3rd International Symposium on Applied Sciences in Biomedical & Communication Technologies (ISABEL2010) in Rome, Italy

[2] IEEE 802.15 WPAN Task Group 6 (TG6) Body Area Networks : http://ieee802.org/15/pub/TG6.html

[3] S. Saleem, S. Ullah, and K.S. Kwak, A Study of IEEE 802.15.4 Security Framework for Wireless Body Area Networks, Sensors, vol.11, No.2, pp. 1383-1395, 2011.

[4] Rijmen, V.; Daemen, J. The block cipher Rijndael. In Smart Card Research and Applications; LNCS 1820;, Springer-Verlag: New York, NY, USA, 2000; pp. 288-296.

[5] Bellare, M.; Kilian, J.; Rogaway, P. The security of the cipher block chaining message authentication code. J. Comput. Syst. Sci. 2000, 61, 362-399.

[6] Xiao, Y.; Chen, H.H.; Sun, B.; Wang, R.; Sethi, S. MAC security and security overhead analysis in the IEEE 802.15.4 Wireless Sensor Networks. EURASIP J. WCN 2006, doi:10.1155/WCN/2006/93830.

[7] AES CCM Encryption and Decryption : http://www.inno-logic.com/resourcesEncryption.html#1

[8] Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality :

National Institute of Standards and technology (NIST) :
http://csrc.nist.gov/publications/nistpubs/800-38C/SP800-38C_updated-July20_2007.pdf

[9] Maulin Patel and Jianfeng Wang : Applications, challenges, and prospective in emerging body area networking technologies , Journal IEEE Wireless Communications Volume 17, Issue 1, February 2010, pp 80-88.

[10] Ming Li, Wenjing Lou and Kui Ren , Data security and privacy in wireless body area networks, Journal IEEE Wireless Communications Volume 17, Issue 1, February 2010, pp 51-58.

[11] S. Heiler. Semantic interoperability. ACM Comput. Surv., 27(2):271{273, June 1995.

[12] M. Lenzerini, D. Milano, and A. Poggi. Ontology representation & reasoning. Technical Report IST-508011, Universit di Roma La Sapienza, 2004.

[13] T. Mens and M. Wermelinger. Separation of concerns for software evolution. Soft. Maint. Evol.: Res. Pract., 14(5):311{315, 2002.

[15] S. Movassaghi, M. Abolhasan, J. Lipman, D. Smith, and

A. Jamalipour. Wireless body area networks: A survey. Communications Surv. & Tutor., IEEE, PP(99):1{29, 2014.

[16] M. Patel and J. Wang. Applications, challenges, and prospective in emerging body area networking technologies. Wireless Comm., IEEE, 17(1):80{88, 2010.

[17] Shimmer. Shimmer sensors, 2013.

[18] M. Uschold and M. Gruninger. Ontologies and semantics for seamless connectivity. SIGMOD Rec., 33(4):58{64, 2004.