Chapter 5
Sensor Networks' Integration

Szymon Fedor, Alex Gluhak, and Srdjan Krco

5.1 Introduction – Motivations for SN Integration Frameworks

5.1.1 Picture of Current WSN Deployments, Problems Related

Sensor networks and applications thereof have been intensively researched in the past decade and a variety of systems have been meanwhile deployed in real-world settings. Most of these applications and the corresponding sensor networks they use are designed as vertically integrated systems [1–3]. In such vertical systems, a sensor network or a limited set of mostly homogeneous sensor networks are deployed for a specific application in mind. The application is mostly the sole user of this sensor network and has a priori knowledge of the capabilities that the sensor network(s) provides. An application also typically knows how to address the respective gateways/sinks of the sensor networks, in order to interact with the sensor networks and shares a common interaction protocol with them.

As the number of the sensor networks that may be used by an application grows, it is becoming increasingly cumbersome for applications to manage direct interactions between those. Furthermore, the reuse of the existing sensor network infrastructure for multiple applications could avoid redundant deployment of similar sensor networks at the same location and provide higher returns for the initial investments costs of the deployed sensor network infrastructure. Recent research has therefore focused on overcoming the inflexibility of the tightly coupled vertical system and proposed several sensor network integration frameworks [3–6]. These frameworks aim to break up the vertical systems into horizontal reusable system components and make them available to a larger set of applications. The frameworks typically provide support functionality that significantly reduces

S. Fedor (✉), A. Gluhak, and S. Krco
Ericsson Ireland
e-mail: szymon.fedor@ericsson.com

L. Gavrilovska et al. (eds.), *Application and Multidisciplinary Aspects of Wireless Sensor Networks*, Computer Communications and Networks,
DOI 10.1007/978-1-84996-510-1_5,

the interaction complexity of applications and eases incremental deployment of new sensor networks. Via these frameworks, applications can gain access to a large variety of connected geographically distributed sensor networks.

While representing first stepping stones for a real-world Internet, a variety of different issues remain unaddressed whereas they are essential for realizing an ecosystem for real-world contexts and interactions. Therefore, the development of sensor network integration frameworks is currently being carried by many industrial and academic institutions. In this chapter, an overview of existing sensor network integration frameworks (SNIFs) is presented, highlighting the main concepts and key features. Various examples of these frameworks are provided covering different design approaches from both industrial and academic organizations. Each of these frameworks is briefly analyzed with the description of key features and innovative solutions. Also their potential limitations and shortcomings are highlighted.

5.1.2 Benefits of Integration

The sensor network integration frameworks provide various advantages for different parties: end users, service providers, WSN providers, network operators, and network service providers. This paragraph describes how the numerous features of SNIF can be beneficial for those entities.

First, better visibility of WSNs obtained with SNIF can be advantageous for the users of WSNs. They can avail of the broader range of measurements which are closer to the phenomena of interest and therefore are more precise. Besides, the availability of diverse sensors would allow deployment of a more sophisticated and complex applications. Also, greater number of available measurements will increase competition between WSNs providers and as a result will lead to a lower cost of those services. All these factors will contribute to a so-called community effect where more and more users will benefit from the WSNs. As a consequence, WSN providers would compensate more quickly the investments incurred for the deployment and maintenance of WSNs. Also the service providers and network operators would benefit from the greater visibility of the WSNs which will increase the demand for their services. In addition, the availability of a greater number of diverse sensors would allow the development of more elaborate services requiring measurements of various quantities.

Also the resource naming and discovery provided by SNIF will bring benefits to various entities. Currently, many WSNs are deployed for a specific application as vertical solutions. With SNIF, these systems could be easily discovered when needed by other applications. In consequence, the service providers can offer more complex applications on the basis of already existing infrastructures. Also it would be beneficial to the end users having access to a broader range of applications.

Most of the SNIFs provide the common interface for the WSN measurements. This will reduce significantly the effort required to access the data provided by WSNs. In consequence, the service providers could develop much quicker applications using WSNs. The current situation of the WSNs market with various independent systems makes difficult the possibility of establishing a connection to

a WSN gateway. With SNIF, the access to the WSNs via a standard interface will be much easier and will require a minimal effort from the service providers.

Some of the major advantages of the SNIFs are the mechanisms which enable security of shared resources against misuse, provide estimates of reliability or verifiability of sensed data against malicious intervention or inadvertent errors, and protect the privacy of users who are being sensed or sharing parts of their data. These security issues are always a concern in shared systems and in most currently deployed WSNs they are not considered because the access to the sensor measurements is granted to a known and limited number of users. The security mechanisms are especially beneficial for the end users and WSNs providers who will be keener to share their measurements if the data are handled securely. As a result, more services using WSNs would be developed and offered, which would be advantageous for service providers and network operators.

The security enhancements provided by SNIF would enable the support of accountability, access control, and billing methods. These features would facilitate development of business models based on the WSNs and therefore more investors would be interested in investing in WSN applications.

Another functionality of SNIF which will also contribute to the expansion of the WSN market is a high-level query processing. Current WSN systems handle very basic queries and because they are designed as vertical solutions, they are not able to respond to complex queries which would require interaction with various heterogeneous WSNs. This feature would enable deployment of sophisticated, complex applications and composed services.

A different feature of SNIF which would enable new applications is caching and data history. Many end users will be interested not only in applications using current and future measurements but also in applications which require sensor measurements from the past. Most current WSN systems can respond to requests for measurement of future events. The previously obtained sensor samples can only be reused if the user explicitly requests to store them for a potential future application. With SNIF, the WSN measurements can be shared among various applications. Therefore, it is possible that the sensor measurements used for one service can be requested in the future by another application. As a result SNIFs would allow service providers to offer applications which require sensor data from the past.

5.2 Existing Integration Frameworks

5.2.1 Overview

5.2.1.1 Historic Perspective on Integration Frameworks Deployment

The research and development in WSNs was initially driven by defense applications. Around 1980 the Defense Advanced Research Projects Agency (DARPA) started the Distributed Sensor Networks (DSN) program in order to study whether the agency's

approach for communication could be extended to sensor networks. This research program resulted in many WSN systems like acoustic tracking of low-flying aircraft [7] or Remote Battlefield Sensor System (REMBASS). These solutions were very expensive and could be used only for a dedicated military purpose. The main focus of the WSN research remained in the military area until the end of 1990s when the first motes for environmental monitoring were developed [8].The availability of low-cost sensor nodes has resulted in the emergence of many other potential applications, from industrial sensing to infrastructure security and health care. Since the end of 1990s many companies have created vertical WSN systems which can only be used for a single application. To stimulate WSN market development by increasing the interoperability between these dedicated systems, the companies developed communication standards for WSNs (Zigbee [9], WirelessHART [10] and ISA-SP100 [11]). Although these standards are already mature and many companies sell products complaint with them, WSN market has not expanded significantly since their publication. Therefore currently, most WSN systems are deployed as vertical solutions and the users do not tend to apply them for multiple applications. As a result the price of WSN nodes is still high and the WSN suppliers try to respond to the market demand by developing systems only for dedicated application.

The WSN research community realized the weaknesses of the vertical solutions (described in Section 5.1.1) and one of the first solutions that addresses these problems was IrisNet project from Intel Research started around 2000 [12]. The project aims to develop a scalable software infrastructure that employs data mining to let Internet users query real-time and historical video information produced by Web cams and other sensors. ÌrisNet takes database-centric approach in its design and users can query for measured data using XPATH query language (see Section 5.2.2.1).

Since the emergence of IrisNet system many solutions have been proposed which facilitate deployment of horizontal applications. They share many features of IrisNet, e.g., database-like design approach (Hourglass [13], SenseWeb [14]) or multi-tier architecture (JWebDust [15], Janus [6]). Existing sensor networks integration frameworks differ mainly by the number of provided features and also by the maturity of the system implementation. Some of them have only been implemented as a prototype (e.g., Janus) and others have been widely used for various applications and are constantly being improved by the research community (e.g., GSN). But none of the proposed solutions have been used in a commercial application and only the future can show if the sensor networks integration frameworks would contribute to the expansion of the WSN market.

5.2.1.2 Summary of Features of the Existing Integration Frameworks

Table 5.1 summarizes nonfunctional properties of the surveyed existing SNIFs. The detailed analysis of each presented SNIF is provided below. Most of the proposals enable the interactions of applications with different heterogeneous SANs and support many necessary types of interactions such as query- and event-based interactions or streaming. Some approaches such as IrisNet and JWebDust provide

applications or framework components with a database view of all sensor network systems. Most others follow a centralized broker structure in which a central entity takes care of interactions with different SAN systems.

Table 5.2 provides a summary of the functional features of the surveyed approaches. Most notably, nearly all proposals fail to address **accountability and access control** of service interactions between application and SANs as well as **privacy, trust, and reputability** of offered SAN services and information. Only Urban-sensing considers access control, privacy, and data integrity as fundamental issue that needs to be addressed as part of the architecture, but current results are limited to mere conceptual discussions. Nearly all approaches provide some way of **service discovery** to the applications. **Service composition**, if provided is preliminary static, mostly based on information available to application at service discovery time. Most approaches do not address issues of **mobility and sudden service unavailability** with respect to longer-lasting queries to ensure the continuity of request information and actuation services. Similarly, closely related **mechanisms for ensuring quality of information (QoI) and actuation (QoA)** are not provided by nearly all solutions. Only SWE and SenseWeb enable QoI attributes to be attached as metadata to sensor readings or information which can be used as selection criteria during service discovery. Few proposals allow **intermediate in-flow processing** services to be accommodated between the SANs and applications. This is essential to facilitate high-level composition of context information and semantic adaptation to be performed, which can even involve information coming from different sensor networks. The surveyed approaches provide limited or no support for **resource arbitration**. In particular for actuation services, mechanisms to manage concurrent access of resources are essential, but all proposals fail to explicitly address these requirements with specific solutions. Finally, many of the surveyed approaches fall short in addressing **management support** within the framework. Only CoBis, JWebDust, and USN provide some management tools within their framework, which address some aspects of the overall system.

5.2.2 Prevailing Sensor Networks Integration Frameworks

SNIFs can be classified according to the system architecture into two main groups: server–client SNIFs and peer-to-peer SNIFs. The former type of integration frameworks can be described as a central system which requires data owners to register their data sources to one central server. These sensing resources are updated periodically to let the server know about their availability. When an application submits a query to search for a service, the central server analyzes the query and finds the appropriate sensor network, and then produces a response. The latter class of SNIFs adopts P2P techniques where each WSN with a gateway acts as a peer. The main goal of P2P overlay is to treat the underlying heterogeneous WSNs as a single unified network in which users can send queries without considering the details of the network. User peers communicate with gateway peers in a P2P approach.

Table 5.1 Summary of nonfunctional properties of the different WSN intergration approaches

	IrisNet	Hourglass	Janus	JWebDust	SWE
Basic concept	Database view	SANs as services in a stream processing overlay	Centralized broker	SQL DB backed with framework services	Web-service-based standards framework
Framework structure	Distributed, two-tiered	In principle distributed but not realized	Centralized broker	N-tier (5)	SPS acts as central-ized broker
Data represen-tation	User-defined DB schema per application	Topic and predicate per service	Exported as functions at SAN	Basic sensor types	XML-based descriptions
Heterogen. of SANs	Yes	Yes, support for low capability by proxies	Yes	Yes, limited to TinyOS support	Yes, SOS facilitates unified interface
Flexibility of queries	Limited to database schema/ XPATH selections	Limited to HCDL expressi-veness	Limited to set of functions provided at a SAN	Limited to SQL-like queries over sensor types	SOS allows simplistic queries
Scalability	Medium	Medium, large signalling overhead for circuit management	Low, due to centralized nature	Low-Medium, all data have to go through centralized DB system	Medium-high, direct WSN access; however, centralized broker
Interaction types	Query, streaming, event-based	Streaming	RPC-like interactions, query and event-based	Query, streaming, event-based	Query, streaming via SOS, event-based via SAS/ WNS
Level of Mediation	Low, exact knowledge required (schema)	Medium, service discovery and CM	Medium, provides discovery of APIs	Low, may implement broker in middle tier	Medium, SPS
Implemen-tation available	Yes	Yes, basic functions, single domain	Yes	Yes	Yes, components

(continued)

Table 5.1 (continued)

SenseWeb	GSN	e-SENSE	Urban	USN	Cobis
Centralized broker, web services	Sensors as service, container infrastructure or virtual sensors, stream processing	Service enabler and GW, IMS-based	High level framework	Service enabler and GW, IMS-based	SoA-based enterprise system integration, Distributed middleware
Centralized broker	Distributed peer	Centralized broker, pub-sub	Unkown	Centralized broker	SoA middleware
Unknown	XML-based description, key-value predicates	XML-based description	Unkown	XML-based (SensorML)	XML-based (CoBIL)
Yes, unified webs service interface	Yes, by the help of wrappers	Yes	Yes	Yes	Yes
Medium-high, provides metadata-based selection	Will depend on how services are located	XML-based queries, depending on knowledge base in SE	Unkown	XML-based queries	XML-based queries
Medium, data flows through centralized broker	Medium-high, depending on support infrastructure	Medium, data have to go through SE	Unkown	Medium, centralized SE in data path	High as completely distributed
Query, streaming, event-based	Query, streaming, event-based	Query, streaming, event-based	Unkown	Query, event-based, streaming	Query, event-based, streaming
Medium	Low	Medium-high, broker provides matching	Limited, Mediation in terms of network enforcement points	Medium-high	Depending on the tools
Yes	Medium	Yes, lacks semantic query process	No, currently only concept	Yes	Yes

Table 5.2 Functional properties of the different WSN integration approaches

	IrisNet	Hourglass	Janus	JWebDust	SWE
Access control	Based on senselet	No	No	Unkown	Unkown
Accountability	No	No	No	No	No
Actuation	No	No	No	No	No
Caching & history	Yes	Yes via operators	No	Yes	Yes, data repositories
Fault tolerance	Replication of distributed DB	Disconnection	No	Disconnection	No
Geo-support	Yes	Yes, via predicates	No	Unkown	Yes, geo-tagging
Mobility support	No	No	No	No	No
In-flow processing	Only at sensor source	Yes	No	Possible in middle/ presentation tier	No
Privacy	Privacy filters on senslets	No	No	No	No
QoI, QoA	No	No	No	No	Limited, allows QoI description of sensor information
Traffic Optimization	Within the overlay of framework	Yes, between overlay nodes	No	No	No
Resource arbitration	Yes on SAN level	No	No	Possible in middle tier; however, not implemented	No
Service composition	Static	Semi-static, only SEPs may be replaced	No	Possible in middle tier	Limited and centralized, SPS
Service discovery	No	Yes	Yes	Yes	Yes
Trust and reputability	No	No	No	No	No
Management support	No	No	No	Yes, web-based tools	No

(continued)

Table 5.2 (continued)

SenseWeb	GSN	e-SENSE	Urban	USN	Cobis
Policies at GW	Yes	Yes, but simplistic	Mediator as policy proxy	IMS-based	No
No	No	No	Unkown	No	No
No	No	No	No	Yes	Unknown
Yes	By use of intermediate services	No	No	No	No
Disconnection through mobile proxy	No	No	No	No	Unknown
Yes, geo-tagging	Yes, geo-tagging	Yes	Space–time coordinates	Yes, geo-tagging	Unknown
Mobile proxies for SANs	No	IMS-based for SAN	Limited, change of mediator supported	IMS-based for SAN	Unknown
Data transformers btw broker and application	Yes, limited to SQL-like manipulation of I/O streams sync of data streams	Within SAN or SE	Possible, but not defined	Event filtering and processing in SE	Unknown
Limited through metadata	Encryption between containers	Simple A&A	Mediator as privacy proxy	No	No
Limited, allows expression of QoI metadata	No	No	Quality checks of data?	No	No
Yes, data combination across queries in broker, caching	No	No	No	No	No
No	No	No	Via mediator	No	Unknown
Static	Static	Dynamic in SE, not implemented	No	No	Yes, own language
Yes	No, provides metadata for discovery of virtual sensors	Yes	Yes via registries	Resource discovery	No
No	Data integrity	No	Yes	No	No
No	Life-cycle and resource management of virtual sensor in containers, time service	No	No	Yes via SE	Yes, support tools

5.2.2.1 Server–Client

SenseWeb

SenseWeb [4] provides an infrastructure for sharing information generated by globally distributed sensor networks. Applications can use SenseWeb to create a variety of different applications, otherwise not possible due to the lack of sensor network coverage and diversity of sensor information. Heart of the SenseWeb system architecture is the *coordinator*, which is central point of access into the system by all applications and sensor network contributors. It can be seen as a centralized broker that coordinates the information access of application to relevant sensor networks. The coordinator is decomposed into a *tasking module* and a *senseDB*. The tasking module receives application requests and tries to find matching sensor network information, considering required accuracy, capabilities, and policies of available sensor networks. The senseDB component of the coordinator tries to optimize data access across different application queries with overlapping space–time window by combining requests for common data whenever possible or serving request from cached data of previous queries. The senseDB also indexes sensor network characteristics and other shared resources in the system and enables their discovery by applications. Sensor or sensor networks are connected via *sensor gateways*, which on one side implement sensor-network-specific access methods, but on the other side, expose a standardized WS API to allow other SenseWeb components to access sensor data streams, submit data collection demands, or access sensor characteristics. Sensor gateways typically implement policies defining what sensor information is to be shared. Sensors that do not have a gateway can be connected by a shared gateway referred to as *Datahub*. In addition, *mobile proxies* are special GWs dedicated to one spatial area that allow mobile sensors to opportunistically provide information, while hiding the temporary availability of different sources to applications. Senseweb also provides data transformers that convert data semantics by some processing. *Data transformers* can be shared across multiple applications and link themselves between applications and coordinator.

SenseWeb has a variety of features that are able to deal with heterogeneity and scalability present in the real-world Internet. Heterogeneity in sensor network access is overcome by providing access through a unified WS interface. Heterogeneity in terms of sensor information quality and access policies is addressed by metadata in the sensor descriptions and learning sensor characteristics (e.g., disruptions of availability) at runtime, while allowing application to explicitly specify their requirements. Improved scalability is achieved by minimizing data collection for common data among different application queries and approximating subset of information, e.g., based on cached information. In addition, SenseWeb allows data to be collected only when actually required by applications. While these features certainly contribute to scalability by reducing the amount of generated traffic in the system, the architecture of SenseWeb does not scale well for many applications and many sensor networks due to its centralized broker nature. While SenseWeb provides support for inserting data transformers, between the coordinator

and applications, it does not seem to provide means to insert such mechanisms closer to the source between coordinator and sensor networks, where they often make more sense. Continuous queries are static in a sense that once the coordinator decides on a way to serve a sensor request, they cannot adapt to changing availability of services. Only mobile proxies provide a limited support to deal with such changes at sensor network level. While mentioning incentives, cost-sharing, security, privacy and trust, SenseWeb does not provide explicit support for such functions in its infrastructure.

Janus

Janus [6] is an attempt to break up the tight coupling between sensor networks and their application, by inserting an intermediate broker into the interaction path. Janus makes use of extensible resolution protocol (XRP) and introduces two entities as part of its architecture, namely an *XRP agent* and an *XRP engine*. Instead of directly interacting with a sensor network, applications interact with an intermediate broker realized by the XRP agent, typically located somewhere in the access network. The XRP agent then interacts with *XRP engine*(*s*) located in the gateway of to the sensor network(s), via an RPC-style interface. The XRP agent can discover available services at a sensor network and gain access to the services by receiving a locator bound to local functions calls at the corresponding XRP engine. These locators can be used as *selectors* to identify services in subsequent RPC function calls at the remote sensor network, realizing both query–reply-based as well as event-based interactions. Different applications can be interfaced to the XRP agents via the implementation of application specific proxies.

Janus achieves a decoupling of sensor networks and applications, by introducing itself as level of indirection between the two systems. Thus, applications and sensor networks can evolve independently, while relying on the Janus framework to remain an invariant achieving compatibility via XRP. Janus is able to integrate different heterogeneous networks, as long as they implement an XRP engine that exports the available services via new RPC selectors and that implements a translation of the function RPC calls to the sensor network native mechanisms. Heterogeneous applications can be supported; however, for each application a specific proxy needs to be implemented that interacts with the XRP agent. Clear interfaces to the XRP agent seem not to be specified, which makes it difficult to write application proxies. The reliance on an XRP agent as centralized broker makes Janus not scale well for large number of applications and sensor networks. Janus does not provide any support for composition of context information of different sensor networks or functions to aid the automatic selection of appropriate sensor networks for interaction. It shifts the onus to perform this task to the application. Janus does not provide any mechanisms that are able to optimize the delivery of same sensor information to multiple applications, nor does it address functions for security, privacy and trust, and accounting.

JWebDust

jWebDust [15] provides a software framework that allows web-based applications to query and control multiple potentially heterogeneous wireless sensor networks. WebDust is based on a multitier architecture, splitting the overall system into five tiers, namely sensor, control, data, middle, and presentation tier. All framework components are implemented in Java, apart from the ones contained in the sensor tier. The *sensor tier* is formed of one or more wireless sensor networks consisting of sensor nodes (motes) operating a TinyOS-based jWebDust firmware. The jWebDust firmware enables multi-hop routing within the WSN and provides support functions such as query subsystem, discovery services, monitoring service, and time synchronization. Sensor networks are connected to control centers that form the *control tier*, acting as gateways between the sensor tier and data tier. The control centers are responsible for the gathering of all readings coming from the sensor network and the forwarding of queries from the data tier to the sensor nodes. Control centers periodically poll the data tier for new available queries and store all available sensor readings into the data tier. Control centers are able to handle temporal disconnections of the sensor tier from the data tier by buffering of sensor readings until reconnection subject to local capacity constraints. The *data tier* is based on a relational database system and hosted by SQL servers. Information is stored in tables that can be grouped into three categories. Mote-related tables store information on the hardware characteristics and sensing services of sensor networks. Query-related tables hold information on currently active queries in the jwebDust system. Sensor-reading tables hold sensor readings that have been performed by each particular mote in the system. The *middle tier* provides a set of reusable components that allows the mapping of the information stored in the tables of the data tier, the manipulation of the information, e.g., creation of queries in the data tier and implementation of rule-based actions and notifications. The *presentation tier* implements the user interface components visible to the end user, providing interaction controls and sensor data visualization tools.

jWebDust decouples applications from the sensor networks via the *n*-tier architecture and supports horizontal composition of WSAN applications of different domains. Applications can learn about the availability of different sensor networks and their capabilities by querying/browsing the data tier. Individual motes within a WSN are assumed to have unique IDs. In order to make them globally distinguishable across multiple WSNs, each WSN is assigned a unique sensorNetworkID. The addition of a new sensor network requires some initial configuration, e.g., assignment of sensorNetworkID and inclusion of novel mote and sensor types. Afterwards, discovery of new motes and sensors in a sensor network executes automatically. jWebDust supports interactions with different heterogeneous WSNs, where control centers are able to hide the heterogeneity of WSNs from the data tier. Although the framework claims to support a variety of different query types, applications currently require explicit knowledge of a sensor network and sensor types (learned from the data tier) in order to create various queries for sensor information. Service broker-like components with semantic query support could

theoretically be implemented as part of the middle tier; however, such functionality is currently nonexistent. Although jWebDust claims to provide concurrent access by multiple applications, mechanisms for resource arbitration are currently lacking. JWebDust also requires all sensor nodes to implement the same firmware for correct query processing and service discovery. While most TinyOS-capable platforms are supported, it requires each sensor node in the network to be reprogrammed and configured prior participation in the framework. jWebDust also lacks functions for security, privacy and trust, and accounting and provides no explicit support for controlling the access to different sensor network resources. The reliance on a central relational database in the data tier can become a scalability bottleneck, once the number of participating sensor networks and querying applications grows.

IrisNet

IrisNet [4] is one of the first attempts to develop an architecture that is able to provide integrated access to globally distributed sensor networks over the Internet. The IrisNet's goal is to reuse the infrastructure of deployed sensor networks by enabling the sharing of generated sensor feeds among many applications (sensing services). IrisNet provides sensing services with the view of a distributed database in which data of different sensor networks can be collected and queried. IrisNet is realized as a two-tiered architecture with *organization agents* (*OA*) and *sensing agents* (*SA*) as fundamental components. A developer of a sensing service provides a database schema tailored for its application, which is implemented on a possibly distributed set of OAs. The group of OAs maps to a single sensing service and must collect and organize sensor data to answer a particular set of service-specific queries. OAs form a distributed database in which data are hierarchically organized in self-describing tags based on XML (to naturally reflect the hierarchal organization of existing geographic and political boundaries). Queries on the database are expressed in XPATH and select data from a node set in the hierarchy. Each of the distributed OAs can store a subset (subtree) of the hierarchy in which each node either points to data sources for sensor streams (represented by SAs) directly providing corresponding service data or other OAs, besides pointing to other nodes, that implement parts of a missing subtree. The IrisNet infrastructure enables the distribution of the query to adequate OAs and ultimately to the SAs providing the required data, and the composition of the final response across multiple OAs on the reverse path. OAs register a global name and IP address with DNS, so queries across distributed OAs can be dynamically resolved. Data of queries can be cached at their corresponding OAs, and repeated requests directly served from cache to improve subject to freshness requirements of the sensing service. IrisNet also provides replications of OAs and placement of OAs as additional mechanism to improve system reliability and query performance. SAs provide a generic data acquisition interface toward sensors and sensor networks and typically collect raw sensor readings as required. Besides a database schema for the OAs, developers of sensing services write so-called

senselets that execute in a secure environment of the SAs. These senselets are able to process (e.g., filter) the incoming raw sensor stream and send the processed sensor information to nearby OAs. In addition, SAs mediate the access of senselets to the resources of its attached sensors.

The IrisNet architecture seems to provide several desired features, ranging from sensor network reuse, application-specific in-network processing resource mediation on the SAs, fault tolerance, and geographic information lookup and seems to scale well as it provides a distributed database view for each sensing service. While providing the possibility of sharing computation across senselets, it does little in optimizing the data traffic from the sensor networks to the sensing services. Data is routed via the OA overlay, which may result in suboptimal data paths (could be alleviated by OA placement if physical hosts are arbitrarily available) and it does not allow concurrent applications to share the same sensor data. Creating a distributed database for each sensing service may lead to services often implementing redundant databases that could have been shared among several applications. IrisNet also does not provide discovery mechanisms that allow sensor networks and their capabilities to be automatically discovered by application developers at design time, not to mention runtime. IrisNet also does not address functions for security, privacy and trust, and accounting.

Ubiquitous Sensor Networks

The work presented in [16] represents the first step of an ongoing research activity Telefonica is performing toward the Ubiquitous Sensor Networks concept from the ITU-T [23]. The presented platform is being designed following a horizontally layered approach, so networks and services can evolve independently. The four layers of the platform, following a bottom-up approach, are: the Sensors and actuators networks, the Gateway (that provides independence from the networking technology), the NGN core (IMS), and the Service Layer (where an enabler is provided). The key elements of the platform are:

- The *USN-Gateway*: is a logical entity whose main goal is to provide independence from the sensing or networking technologies used to communicate sensors and actuators. The independence is provided by performing two transformations: from one side it provides homogeneous communication toward and from the sensors and actuators networks, and from the other side it provides homogeneous data representation. It is being defined as an IMS User-equipment which already provides important functionalities like AAA and it allows to be deployed in a wide range of devices.
- The *USN-Enabler*: is defined as an OMA enabler, intended to allow services to be created in a cost-efficient way following a horizontal approach, where multiple services can access the same sensor and actuator networks. The basic functionalities it provides are: resource discovery, publish–subscribe–notify mechanisms, event-filtering and processing, and homogeneous remote management.

More than the functionalities it provides the key issue of the USN-Enabler is the way in which it has been designed, since it follows the OGC Sensor Web Enablement Family of Standards and the OMA Presence Simple and XDM specifications.

- *Standardized homogeneous representation of sensor data and metadata*: It provides homogeneous representation of the sensors and actuators representations and measurements following the OGC® SensorML and Observations & Measurements (O&M) standards.

The more interesting issues brought by the platform, more than the functionalities it provides, are the way in which these functionalities are provided, since this approach, instead of redefining some existing functions, uses the existing standards. Especially important is the use of SensorML as the language that unifies the heterogeneous sensors and actuator definition. Even considering that the USN Platform presents an interesting approach to tackle the problem of integrating sensors and actuators to services, mainly due to the extensive use of standards, it is still a first step and much work still needs to be carried out for it to be considered as a solution for the USN. Issues like billing, trust, accounting, and high-level interaction mechanisms are not still attached. The architecture can be viewed as centralized, but with catalog functionality in order to have some distribution of functionality. It is mentioned that some of the functionalities could be provided by elements like the Gateway, but it is not yet defined.

e-Sense

The framework presented by the e-SENSE approach [17] aims at integrating sensor networks into the IP multimedia subsystem (IMS) of future mobile and converged networks. The framework allows applications in IMS-based service platforms to access sensor and context information from a variety of sensor networks with heterogeneous capabilities. The framework introduces two architectural components, a context service enabler and gateway extensions to sensor networks. The context service enabler provides sensor-based context information as a dedicated service via a unified interface using standardized IMS protocols (such as SIP). Thus, the context service enabler can be used as a service building block for the realization of various different context-aware applications hosted on application servers in the IMS domain. Gateway extensions allow the integration of heterogeneous sensor networks into the IMS domain by implementing sensor-network-specific mechanisms on one side and provide service functions for interaction with the context service enabler on the other. Each sensor network gateway is an IMS user identified by a unique IP multimedia private identity, several public ones for each gateway. Using the proposed gateway extensions, the sensor network systems register their presence with the IMS core platform and available services with the context service enabler. It is expected that applications and services will express their context requirements on a semantically high level of abstraction and that the context service

enabler may have to interact with multiple sensor network systems to derive the required context information components. Based on an incoming context information request of an IMS application context service enabler performs a decomposition of high-level context information requests into low-level service task graphs to be executed by one or more wireless sensor networks. It then requests the low-level services from the identified sensor network gateway and composes the context information response from the required context information components of the service responses. Communication between sensor networks and gateways is enabled by publish/subscribe mechanisms based on the SIP event framework.

The e-SENSE framework offers several desired features such as discovery of available sensor networks and their service capabilities. It enables a decoupling of application from the underlying sensor networks – that is, applications can enquire the contextual information without requiring knowledge of the underlying sensor networks. Application queries can encompass information that may be jointly provided by multiple sensor networks and processing and composition can be achieved either in the sensor networks/gateways or in the service enabler. While making use of scalable well-understood signaling mechanisms, it raises scalability concerns as communication requires always the involvement of the centralized service enabler. Although multiple physical instances of a context service enabler can be deployed, the architecture does not address how coordination between different such instances is achieved. Optimization of data flows across different queries are currently not addressed by the proposed frameworks, as are not adoptions of existing queries to changing conditions in the system. Furthermore, the architecture of the e-SENSE framework falls short in considering security, privacy, trust, and accounting issues.

5.2.2.2 Peer-to-Peer

Global Sensor Networks

Global Sensor Networks [18] is an approach of providing a distributed middleware platform for integrating heterogeneous sensors into a "sensor web" providing internal stream processing capabilities on the exchanged sensor information. The architecture of the GSN framework is based on distributed peer entities called GSN containers. *GSN containers* are typically deployed at normal Internet hosts or servers and communicate with each other via point-to-point connections. Core element in GSN is the so-called *virtual sensor* abstraction. Virtual sensors abstract implementation details to access information from physical sensors and allow a unified way of treating sensor services or composed sensor service by the middleware. The specification of a virtual sensor includes metadata for identification and discovery, the structure of input and output streams, SQL-like internal streaming processing and properties related to life-cycle management and physical deployment. Virtual sensors have one or multiple input streams and produce exactly one output stream. Input streams can come from physical sensors interfaced via implementation-specific wrappers or other virtual sensors. Virtual sensors can manipulate and combine

streams of different characteristics using SQL-like operations and conditionally produce output streams (event-like). Virtual sensors can be dynamically deployed on the GSN containers and the production of its output stream is dynamically triggered by the arrival of input streams. Besides running instances of deployed virtual sensors, GSN containers provide additional functionality supporting the management of the virtual sensor instances and their required resources, function to manage streams and resources required for stream processing, query management (request input data from other virtual sensors and keep track of other virtual sensors requiring their output), and a storage layer for the management of persistent storage of data streams. Access to GSN-container internal functions is provided by an interface layer, which is used to communicate between GSN containers or can be accessed directly via web interfaces. Besides providing access control at different levels of granularity down to the virtual sensor level, the interface layer provides integrity and confidentially functions for the exchange of data streams.

GSN provides many interesting features. The ability to create aggregate virtual sensors from different heterogeneous sensor information sources suits well the context information processing demands. Unlike other surveyed frameworks GSN offers access control to sensor information and integrity protection. In addition it provides a plug-and-play-like feature for integration of new sensors which allows upon detection of a new sensor the dynamic download of a IEEE1451 transducer electronic data sheet and automatic generation of a virtual sensor (given the wrapper code for the WSN technology is known). A current limitation represents the SQL-like stream processing operations, which do not allow complex processing data stream and data fusion to be performed in the system. The decentralized peer-to-peer nature of the system seems to indicate good scalability properties. It is unclear however, how data streams between GSN containers can be optimized, e.g., by selecting the same virtual sensor streams for different independent queries. Point-to-point transmission of streams between GSN containers may be another feature potentially reducing its scalability. Although meta-information is provided for virtual sensors, which can be used for service discovery, it is still unclear what mechanisms would be used to discover virtual sensors. The current descriptions seem to "hard code" required input streams into the virtual sensor descriptions, which does not make dynamic composition or late binding of virtual sensors possible at runtime. GSN does not seem to provide any infrastructure support for accounting nor does it provide a little information on an infrastructure that can be used for dynamically composing or modifying active virtual sensor services.

Sensor Web Enablement

Sensor Web Enablement (SWE) [19] is an initiative by the Open Geospatial Consortium aiming at the development of a set of standards to enable the discovery, exchange, and processing of sensor information and tasking of sensor systems over existing Internet. SWE strives for plug-and-play-like integration of sensor networks and enabling protocols to make those accessible and controllable by web-based

applications. The current standards framework encompasses seven different standards, some of them completed and others in draft stage. Three of the standards are concerned with the XML-based encoding and representation of sensor information/observations and the description of sensor capabilities and related information processing steps. The remaining four standards describe standard web service interfaces for tasking of and interaction with sensors. The *observations and measurement* (*O&M*) schemas provide XML schemas for representing and observations, measurements, procedures, and metadata of sensor systems and efficiently encoding them for transfer and archiving. The *sensor model language* (*SensorML*) supports the description of a functional model of a sensor system by providing models and XML schema for describing processes of measurement and post-measurement processing and their exact chaining. The *transducer markup language* (*TML*) provides models and CNK schema for describing hardware response characteristics of transducers (more complex integrated sensors/actuators) and efficient method for encoding and real-time transport of sensor data. While partially overlapping with SensorML, TML focuses more on support of streamed real-time sensor information flows, preserving their spatial and temporal association for later data fusion. The *sensor observation service* (*SOS*) specifies a web service interface that allows SWE clients to obtain observation and measurements from a collection of sensors. The SOS also allows clients to access metadata information about associated sensors, platforms, procedures, and other metadata associated with observations. The information is exchanged using the three aforementioned XML-based data formats. The *sensor planning service* (*SPS*) acts as a broker service between clients and different SOS. It allows clients to determine the availability of certain sensing services that may be needed to satisfy collection requests and the feasibility of those via a standardized web service interface, potentially spanning multiple sensor systems and management of such collection requests. The *sensor alert service* (*SAS*) provides web service interfaces that allow clients to subscribe to alerts/event notifications of particular sensors. The SAS acts only as a registry that enables clients to determine the nature of available alerts, protocols used, and the options to subscribe to specific alerts. Alerts or event notifications themselves are forwarded my messaging servers. The web notification service (WNS) specifies a web service interface that allows clients to interact with one or more services in an asynchronous way. WNS provides support for both unidirectional and bidirectional asynchronous communication.

The standards framework of SWE addresses many issues including standardized descriptions for sensor/actuation platforms, actual sensor information and processing chains in a sensor web as well as several interfaces for applications to interact with sensor systems, perform asynchronous communication, and manage event notifications. In addition the SWE framework defines interfaces for a service-broker-like component, the SPS, which enables more complex interactions between applications and sensor systems, such as determining suitable sensor observation components across multiple sensor systems and information repositories to satisfy more complex sensing requests and the respective tasking of the sensor systems. While specifying the high-level architectural framework and interfaces, the SWE does not address the realization of the framework services or required interaction protocols.

Some of the standards are at draft stage and still undergoing specification and further harmonization between the different standards brought into the framework from outside, such as SensorML and TML is required. The framework does not explicitly address aspects of security, privacy, trust, accounting, and resource arbitration. In addition, the SWE representation formats allow application only to express simplistic queries and are not suitable for the high-level declarative service interface as well as the complex processing of sensor information inside of the system based on ontological models.

Hourglass

Hourglass [13] aims at creating an Internet-based framework for connecting heterogeneous geographically distributed sensor networks with applications that require sensor information. Hourglass provides an infrastructure for data collection referred to as data collection network, which handles service naming, service discovery, route setup from sensor networks to applications, and provides support for integrating internal services along the data dissemination path to perform aggregation or buffering of sensor information. Hourglass primarily addresses stream-based aggregation and processing of sensor information that is required by applications over a longer period of time. Hourglass treats the sensing and processing capabilities sensor networks offer as services, and extends the service concept to also encompass any intermediate processing service on sensor data. Typically services can act as data consumers, data producers, or both. Services in Hourglass are organized into service providers. Each service provider comprises more Hourglass nodes forming a single administrative domain, entering or leaving the Hourglass system as a unit. Each service provider needs to support minimum functionality in terms of a *circuit manager* and *a registry*. In addition a service provider can provide several generic or application-specific services. A service registry is a (distributed) repository of information about various services and active circuits in the Hourglass system. It is a lookup service that allows the resolution of service endpoints. Each service provider typically maintains an own local registry, with which active services of a service provider register via service announcements. Such service announcements typically contain communication endpoint identifiers, topic name, predicates, and expiration time as entries are kept as soft state. An application that aims to establish a "streaming session" with one or more sensor information sources and intermediate processing first queries the service registry for available services in the Hourglass system. It then specifies its query requirements as so-called circuit descriptions that link one or more data producers and a data consumer with possible intermediary in-network services into a logical data flow. The circuit manager instantiates the described logical flow as network data flow by establishing connections between the different physical nodes offering the respective services. The hourglass service layer manages the invocations to the service interface and the multiplexing of data to and from the connected circuits. Sensor data are routed along the established path and possible processed at intermediate nodes.

Hourglass provides reliability to system dynamics by explicitly supporting a mechanism to deal with temporary disconnections of a circuit, that is, if the connectivity to a service provider that is part of a circuit becomes unavailable. Disconnections are monitored by heart beat mechanism along the circuit, based on explicit control messages or implicitly by data that are exchanged. Once a disconnection is detected appropriate actions such as buffering of data can take place in the circuit. Thus, Hourglass offers the advantage to modify existing circuits to adapt the services to changing conditions for continuous application queries. It allows optimization of the delivery of same sensor information by combining transmission between service endpoints across multiple circuits/applications. While the architecture has the intention to scale well by planning for distributed operation across multiple service providers, it leaves open how respective service registries distributed across multiple service providers interact or are managed and how the connection managers of different service providers cooperate to establish connections across multiple domains. The overhead for establishment of circuits for each data request by application together with the fact that state needs to be maintained at each node that is part of the circuit is a severe scalability concern. While it may be justified for streaming-type queries for longer periods of time, it does not suit well one-shot queries or periodic queries with little data exchange. However, the framework does not address functions for security, privacy and trust, and accounting.

Urban Sensing

In the Urban Sensing project [5] they consider three types of applications: personal, social, and urban. A personal application uses information about the end user for the purpose of the end user. A social application mimics Facebook and other social networking sites, where data are shared among a set of users for free. In urban applications the users share data with the general public, and the importance of identity control, etc., is thus much higher. It is argued that new network architecture is required in order to share data in a controlled way and to assure basic quality checks of data. In this the authors see an evolution from single-domain WSNs to collective/federated WSNs to full integration into the full global infrastructure. The federation of WSNs is referred to as the sensor fabric.

In order to achieve full integration, the global network must know about the abstractions used in forming the sensor fabric, and the sensor fabric must import notions about the future global network into itself. It is hence a two-way process.

Abstractions required to form the sensor fabric could be of the following types:

- Space–time coordinates
- Policy-mediated rendezvous based on data properties and metadata
- Aggregation-based reliability

The authors argue that embedding these abstractions into the global network changes the network from *host-centric* to *data-centric* in nature.

The authors subsequently argue that most important to solve while incorporating sensors into the global network are the issues of verification, privacy, and dissemination.

The authors therefore seek to incorporate embedded basic data protections into the fundamental mechanisms of the network. The proposed architecture incorporates the four entities below, and also existing network services such as trusted Certificate Authority (CA).

- *Sensors*: These are data sources at the edge of the network; these are not simply a pure source of data, but can also provide a control point to the external world. These control points could be for the purpose of configuring the sensor or for providing global contextual information.
- *Subscribers*: These are the users of the data provided by the sensors; individual users of data or applications providing some value-added service.
- *Registries*: These are network services that help subscribers to find and bind with sensor data streams. Sensors register here and subscribers query these in order to find the sensors they want. The type of handle provided by the registry is extremely important.
- *Mediators*: Nodes in the network that provide selected in-network functions on sensor data streams. These services could be to perform verification of data streams or to provide anonymization of sensors to subscribers.

In a typical deployment scenario, a sensor owner registers a sensor in Registry 1 via the mediator Mediator 1. Registration contains sensor type, location, and context + disclosure and verification rules. If sensor is mobile, it may change mediator over time. The sensor then initiates data transmission to Mediator 1 – either on demand or proactively depending on configuration. The role of Mediator 1 would be to act as a privacy proxy and to provide a network testimony of the validity of the context of the sensor.

A subscriber then sends a query to Registry 1 via a mediator Mediator 2. The query has to go through a mediator since the sensor may have privacy rules depending on the context of the subscriber, and Mediator 2 attests this context. The registry then returns a pointer to the data streams. Mediator 2 can then bind to these data streams, which means that in this case it binds to Mediator 1. Mediator 1 can now run its own privacy rules and allow/disallow sensor data access.

The urban sensing architecture takes a very protective stance on participants, and argues that network support is required for verification and dissemination of data. By embedding support for these functions into the network, it is easier to efficiently and securely execute them. Many issues around management of sensors/sensor networks are, however, not touched upon, and the underlying efficiency of processing/context sessions are hinted at. The basic security architecture is only sketched.

CoBIs

The CoBIs project [20] developed a radically new approach to business processes involving physical entities such as goods and tools in enterprise environments. Advances in networked embedded systems were applied to embed business logic in physical entities to create so-called Collaborative Business Items (CoBIs). Such items enable to relate more closely the state of an enterprise as represented in a business

process with what is actually happening in the real world. Thereby, business processes can be extended to the "point of action" rather than via a centralized back-end system.

The central concept of the CoBIs project was to use a common service paradigm throughout all layers, from the enterprise application down to the logic executed on sensor nodes. A middleware was built based on a service-oriented architecture (SOA). The middleware allows the deployment of business logic in the form of services to the edge of the network and onto the sensor nodes themselves. CoBIs were focused on providing the basic SOA framework as well as the tools to monitor and manage the network. Using a service-oriented architecture in the context of distributed embedded devices as well as sensor and actuator networks solves several problems that are usually associated with such systems. Solutions concern especially, the integration of sensors and actuators with enterprise systems as well as the management, monitoring, and administration of a system with highly distributed logic.

In addition to the SOA framework, a set of reusable collaborative services was defined and described in a newly developed service description language called CoBIL. A CoBIL service description includes a definition of the interface, which is based on the Web Service Definition Language (WSDL). Furthermore, it includes a textual description of the service as well as information about the composition of the service and technical constraints for the deployment.

Three different sensor network platforms, namely Particles, μNodes, and Sindrion, were integrated with the middleware through a common abstraction layer to demonstrate the feasibility of connecting heterogeneous hardware to the system [24]. The different platforms have different characteristics: Depending on the application scenario, one can thus choose the most suitable technology. Criteria have been developed that will help end users to make that choice, also comparing it to existing technology like RFID and wired sensors.

While the middleware, service description language, and system support tools developed could be the foundation of a widespread, multi-partner sensor network infrastructure, CoBIs did not address formal semantics and context models or security issues.

5.3 Road Ahead

5.3.1 Introduction

Internet that we know today was designed 40 years ago as a tool that will facilitate easier exchange of information between researchers. From that vision, Internet grew to a ubiquitously available platform people and businesses depend on in all aspects of everyday life: social networking, business applications, health care, learning, information exchange, etc. With proliferation of mobile networks and particularly with the introduction of high-speed mobile technologies (HSPA – High Speed Packet Access), the requirements for Internet access broadened from just home and office environments to any place and at any time, including while on the move in a car, a bus, or a train. In addition to that, it is becoming a norm to have

private data stored online to make it always accessible. The pervasiveness of Internet and its intertwining with everyday life has brought a number of new requirements as well as problems together with the benefits it provides. Security and data privacy present a big problem, with spam email, scams, and identity thefts contributing a huge percent to the overall Internet activities.

In order to efficiently support these new trends, a number of activities have been initiated in the last couple of years. Their intention is to design a new generation of Internet, commonly known under the name of Future Internet. In Europe, these efforts are primarily combined under the FP7 program and it is Future Internet Assembly (www.future-internet.eu). In Japan, the driving force in this domain is the AKARI initiative (http://akari-project.nict.go.jp/eng/index2.htm), in the USA it is the Future Internet Network Design (FIND) project (http://find.isi.edu) and FIF in South Korea (http://mmlab.snu.ac.kr/fif).

One of the main points that Future Internet will bring is the integration of the physical and digital worlds, i.e., embracing the Internet of Things as one of its core components. Numerous sensors, actuators, RFIDs, machines, and in general "things" will become easily accessible to other Internet users and devices, thus forming infrastructure that pervades into all aspects of our lives. This will enable efficient interaction with the physical world, adaptation of Internet applications to the users' contexts as well as influencing and changing the environment based on the applications' settings.

The "things" will range from simple sensors measuring temperature or humidity, to complex intelligent semantic systems capable of providing answers by combining a number of inputs, simple sensors, actuators, and other network services like location, security, and charging.

Before this vision becomes a reality, a number of technical, legal, socioeconomic, and business challenges and issues have to be resolved. Discovery of information and capabilities provided by different "things" in such distributed environment, standardized description of the capabilities, scalability of solutions to support huge number of connected "things," how to trust the information provided by unknown sensors embedded somewhere in the environment, how to protect privacy of "things" providing the information, new business cases on which applications and services will be built and provided, how will people react and adapt to such new Internet, etc. are just some of the challenges ahead.

In the following two sections, two projects dealing with some of these issues are described. CommonSense is an industry-driven project, focusing on integration of sensors and actuators in the mobile networks context. FP7 SENSEI is a large integrated project under the EU FP7 program with a goal to design a framework for integration of the digital and the physical worlds.

5.3.1 CommonSense

The CommonSense system [21] was proposed to enable the vision of ubiquitous sensing where sensor networks provide the missing link between the virtual and

physical world. Today we observe a tremendous increase of mobile subscribers with already existing about three billion users of handheld computers. These mobile devices are powerful communication and computing multipurpose devices that are increasingly being equipped with a number of different sensors: image, sound, light, temperature, acceleration, RFID readers, etc. The ability to interact with sensors in their vicinity via built-in short-range communication interfaces like Bluetooth, in addition to the previously mentioned characteristics, make mobile devices an excellent platform for sensing the physical environment and interacting with it. The authors of CommonSense first analyze the roles of different entities which will potentially be involved in the provisioning of WSN services in the future, and they propose the system architecture that incorporates the conclusions of their study.

The first identified entity is the WSN provider who provides the sensor network its services and because of the equipment ownership, the WSN provider defines sensor network access and utilization policies. Higher-level services are provided to the end-users service providers. They combine and process different sensor networks services and other required input like Google Maps. The third entity which provides the link between two aforementioned parties is called CommonSense provider. It acts as a broker to the service providers and helps them to find a sensor network, enforces access policies set by individual WSN providers, processes the data received from multiple sensor networks before delivering these to the requesting service provider, and provides authentication, accounting, and billing functionality. The role of the CommonSense provider is to provide a unified interface to services provided by heterogeneous sensors and actuators. The CommonSense providers will collaborate with other entities such as location providers, telematics information providers, presence providers, etc. These entities, referred to as the third-party service providers, will process collected information in a specific manner or will be adding own information to the mix, thus providing additional value to the services provided by the CommonSense providers.

The CommonSense system is based on a tiered-service-oriented architecture. The service providers interact with the CommonSense provider, who in turn is the entity directly interacting with the WSNs. The service providers treat the CommonSense provider as an entity providing services, and thus have no direct knowledge or influence over how the CommonSense provider finds the appropriate data to respond to their requests. This constitutes the first level of the service architecture. The CommonSense provider then in turn treats the individual WSNs as entities offering services. This means that the sensor networks have to be able to describe themselves, where they are, and what they can offer.

Traditionally SOAs focus mainly on peer-to-peer workflow-driven processes. In CommonSense architecture the authors instead envision that individual, moving WSNs offer very thin atomic and dynamic services while the CommonSense provider offers more complex services by combining these primitive WSN services to create for example mash-ups.

The authors provide several reasons supporting this tiered architecture. The first is that they wish to create a scalable system where the focus is not on every single individual sensor, but rather collections of them offering a service. Secondly, focusing

on services only the collection of sensor samples becomes independent of specific sensor network implementations as long as these networks are able to describe how the service they provide can be used. Thirdly, the authors recognize the problem for an autonomous device to manage multiple dynamic security associations and the associated authorization decisions and therefore they propose to outsource the authorization task to the CommonSense provider – something which is enabled by the tiered SOA.

The proposed architecture is mapped on three technology planes: Communication services, Application enablers, and Application plane (see Fig. 5.1). Applications are built using common service blocks residing in the Application enablers' plane and all are connected by a number of network solutions residing in the Communication services plane.

Different domains are identified on each plane. The WSN, CommonSense, and third-party service domains comprise the service plane, while the Peripheral, Access, and Core domains comprise the Communication Services plane. The application providers providing end-user applications reside on the application plane. In the Applications plane the authors differentiate between existing applications that do not depend on physical world context (e.g., call setup) and applications that cannot exist without one (e.g., burglar alarm).

The Application enablers' plane is divided into three domains: WSN domain, CommonSense domain, and third-party services domain. The WSN domain comprises all atomic sensor services, i.e., services provided by individual sensors or sensor networks and used as small building blocks of more complex services offered by the entities residing in the CommonSense domain. The CommonSense domain is where the core functionality of the proposed architecture resides. This domain does not host any specific applications, but provides a set of enablers for all types of applications. These enablers include information exchange, sensor network discovery, data processing, aggregation of atomic sensor network services (sensor mash-ups), Authentication, Authorization, and Accounting (AAA) services. In short, the domain creates the possibility of having a dynamic binding between applications and WSNs. It is mainly based on semantic technologies which provide access independency. A single attachment point for sensor networks also facilitates security

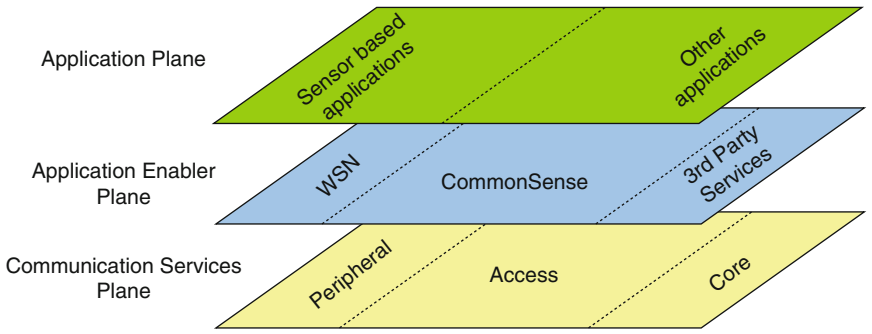

Fig. 5.1 The proposed layering in CommonSense architecture

and privacy support. The authors defined a set of needed core functionalities in the CommonSense domain. The most important function is the

Service Control Function (SCF) which controls the interaction with all external parties. High-level service requests from applications are analyzed by the SCF with the support from the Request Analyzer (RA) and sensors, sensor networks, or existing sensor mash-ups. The Request Analyzer (RA) is a decision engine that can decompose a request from an application to multiple individual information requests, and then recompose an aggregated answer. The output from the RA is used to search the SR (a database containing registration descriptions of all attached WSNs) and find WSNs with matching capabilities. Once suitable WSNs are identified, the SCF issues either standardized low-level service requests or special legacy WSN requests using a mediating function (called Service Gateway). The WSN provider functionality on the Application enablers' plane is represented by Service Gateways (SGW). The SGW represents atomic sensor network services and is responsible for mapping the SCF requests onto the sensor network technology specific commands, which is a core requirement for interoperability. The CommonSense domain entities will use third-party services as an additional tool in creation of application responses. The applications can also use the third-party services directly if they are required by the application logic. Some of the already existing services provided by the network are considered as the third-party services in the network of the future. Examples of the third-party service providers are presence servers, location providers, object identity resolution providers, etc.

The communication services plane provides the underlying secure and reliable communications services to the Application enablers and the Application plane and enables interaction of all their entities across the different domains. The Communications service plane is divided into three domains: Peripheral refers to the local connectivity functionality (e.g., Bluetooth, Zigbee), Access refers to the wireless and wired last hop connectivity functionality (e.g., WCDMA, ADSL), and Core refers to the actual backbone.

5.3.2 SENSEI

FP7 SENSEI (www.sensei-project.eu) is a large integrated project under the EU FP7 program. It has the following objectives: to create a common, global, WS&AN (wireless sensor and actuator networks) framework that will enable making the WS&AN available to services and applications via universal service interfaces. The following are the main planned tangible results of the project:

- *A highly scalable architectural framework* with corresponding protocol solutions that enable easy plug-and-play integration of a large number of globally distributed WS&AN into a global system – providing support for network and information management, security, privacy and trust, and accounting.
- *An open service interface and corresponding semantic specification* to unify the access to context information and actuation services offered by the system for services and applications.

- *Efficient WS&AN island solutions* consisting of a set of cross-optimized and energy-aware protocol stacks including an ultra low-power multimode transceiver targeting 5 nJ/bit.
- *Pan European test platform*, enabling large-scale experimental evaluation of the SENSEI results and execution of field trials – providing a tool for long-term evaluation of WS&AN integration into the Future Internet.

5.3.2.1 Initial SENSEI Architecture

The high-level overview of the SENSEI architecture is given in Fig. 5.2. The central part of the architecture is the SENSEI (Real-World) Resource layer. This layer provides a set of interfaces that applications and services use to interact with the physical world. Within the SENSEI Resource Layer, the main concept is that of a resource. A resource is a conceptual representation of any information source that enables real-world sensing or has the ability to act upon the environment and entities within it. The concept covers not just the actual entities that have direct access to the physical world, but also the entities with indirect access acquired via aggregation, fusion, or inference from other SENSEI Resources. All WS&ANs are presented by their descriptions (Resource Description) detailing the capabilities of the corresponding WS&ANs including location, access policies, available operations, type of information produced, etc.

The SENSEI resource layer interacts with the communication layer to map all requests received from the applications and services to appropriate communication channel. SENSEI community management is responsible for management of all SENSEI entities.

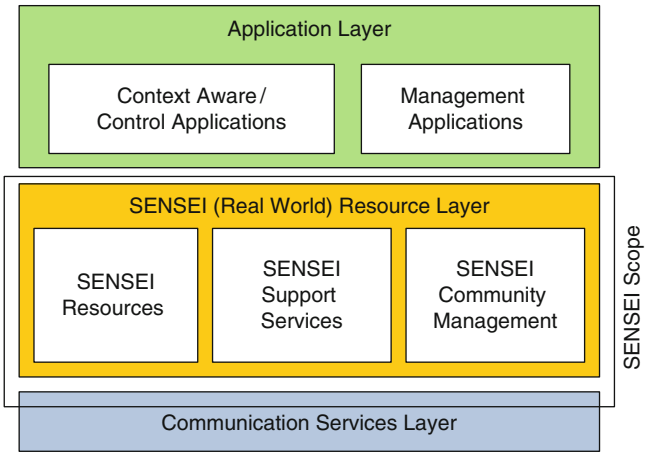

Fig. 5.2 High-level overview of SENSEI architecture

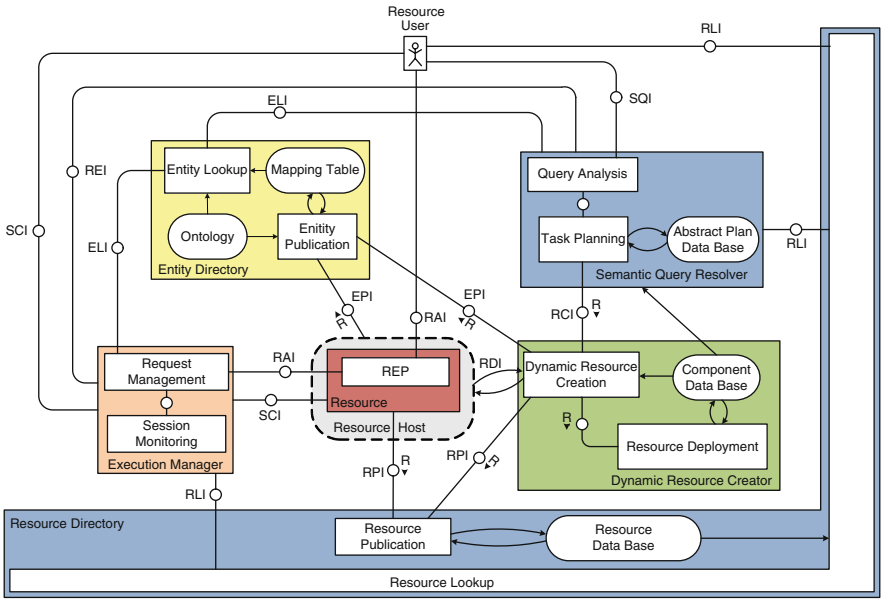

Fig. 5.3 SENSEI resource layer architecture

SENSEI Resource layer architecture is given in refrence to Fig. 5.3. The following components comprise this layer:

- (Real-world) resource provider consists of one or more real-world resources. It provides access to these resources via Resource Access interface (RAI) and is responsible for interaction with the rendezvous component to publish information about its resources.
- Rendezvous makes the glue between resource clients and resources. Its purpose is twofold: to provide mechanisms for resources to publish their capabilities and functionalities (resource publishing interface – RPI), and to provide mechanisms for the resource clients to lookup particular resources (resource lookup interface – RLI). It also stores resources descriptions of all available resources at any given time in the resource directory.
- Semantic Query Resolver (SQR) is an advanced component, responsible for analysis of complex queries and their decomposition to simple queries. These simple queries are then used to search for adequate resources in the resource directory. In case no adequate resources exist, the SQR can trigger Dynamic resource creation component to create a new resource based on the available resources.
- Dynamic resource creation is a component capable of dynamically combining several resources into one when required to meet specific functional requirements.
- Execution management manages long-lasting interactions and handles changes in resource availability at runtime.

The rendezvous component has the central role in the system. It provides a repository for all resource descriptions currently available in the system as well as two interfaces: publishing and lookup. The publishing interface is used by resources to register in the rendezvous and publish own description. The lookup interface is used by resource clients to search for resources capable of fulfilling the clients' requests. This type of architecture allows late binding of resources, i.e., the applications do not have to define which resources shall be used, but only the type of information required by the application. Based on the description of the required information, the Resource layer provides the actual resources that can most efficiently at a given time provide the requested information.

All resources provide access policies as part of their descriptions, outlining who and under which circumstances can access a given resource. The rendezvous component uses these policies to grant or refuse access to the resources based on their credentials.

The proposed architecture remains to be tested and validated in a test bed combining a number of different applications in a real-world setting. Based on this evaluation the architecture shall be improved. A number of interesting points remain to be resolved and proved like the scalability of the system, interaction between different SENSEI system, management of such system and its components, etc. The project is very well embedded in the EU's FIA initiative and to a great extent influences the design of a Future Internet with the specific real-world requirements.

5.4 Conclusions

In recent years many SNIFs have been proposed as a result of a tremendous increase of heterogeneous WSNs deployments. This chapter describes the existing SNIFs from a historical perspective and compares their functional and nonfunctional properties. Also a description of currently developed systems is provided with a perspective view on trends in mobile and future internet.

Currently there is no existing standard technology for the SNIFs and none of the developed systems has gained prevailing attention as a reference model for future research. However, some of the described frameworks are built by a large consortium of industrial or academic partners (e.g., SENSEI) or are promoted by a standardized organization (e.g., Sensor Web Enablement). Others are developed and available as an open source project used by an increasing number of contributors and users (e.g., GSN). It will take some time until the SNIFs gain broad attention as an integral part of the communication system. Before then, WSNs have to be largely deployed and ubiquitous so that the benefits of SNIFs become pronounced and they will emerge as the only solution to handle and manage the amount of data produced by the sensors. This may happen soon because the WSNs market will grow over eleven times within next 10 years [22]. Then the SNIF which is most advanced and provides broad scope of features will have the biggest advantage and potentially will dominate other solutions.

References

1. Mainwaring A, Polastre J, Szewczyk R, Culler D, Anderson J (2002) Wireless sensor networks for habitat monitoring. ACM International Workshop on Wireless Sensor Networks and Applications, Atlanta, GA, 28 Sept 2002
2. Paek J, Chintalapudi K, Govindan R, Caffrey J, Masri S (2005) A wireless sensor network for structural health monitoring: performance and experience. The Second IEEE Workshop on Embedded Networked Sensors EmNetS-II, Syndey Australia, 30–31 May 2005
3. Lombriser C, Bharatula NB, Roggen D, Tröster G (2007) On-Body Activity Recognition in a Dynamic Sensor Network, In Proceedings of 2nd International Conference on Body Area Networks (BodyNets), Florence, Italy, June 2007
4. Gibbons PB, Karp B, Ke Y, Nath S, Seshan S (Oct–Dec 2003) IrisNet: an architecture for a worldwide sensor web. IEEE Pervasive Comput 2(4):22–33
5. Srivastava M, Hansen M, Burke J. Parker A, Reddy S, Saurabh G, Allman M, Paxson V, Estrin D (April 2006) Wireless urban sensing system. CENS Technical Report #65
6. Dunkels A, Gold R, Marti S, Pears A, Uddenfeldt M (2005) Janus: an architecture for flexible access to sensor networks. In: Proceedings of the 1st ACM workshop on dynamic interconnection of networks DIN '05. ACM, Cologne, Germany, pp 48–52, 2 September 2005
7. Lacoss RT (1987) Distributed mixed sensor aircraft tracking, presented at the American Control Conference, Mineapolis, MN
8. Polastre J, Szewczyk R, Culler D (2005) Telos: enabling ultra-low power wireless research. In: Proceedings of the fourth international conference on information processing in sensor networks: special track on platform tools and design methods for network embedded sensors (IPSN/SPOTS), UCLA, Los Angeles, CA, USA, April 25–27, 2005
9. ZigBee Allince (2008) www.zigbee.org
10. HART Communication Foundation (2009) http://www.hartcomm2.org/index.html
11. International Society of Automation (2010) http://www.isa.org/MSTemplate.cfm?MicrositeID=1134&CommitteeID=6891
12. Gibbons P, Karp B, Nath S, Ke Y, and Seshan S (2003) IrisNet: An Architecture for a Worldwide Sensr Web, In IEEE Pervasive Computing, Special Issue on Sensor and Actuator Networks, IEEE Press, October-December, 2003
13. Shneidman J, Pietzuch P, Ledlie J, Roussopoulos M, Seltzer M, Welsh M (2004) Hourglass: an infrastructure for connecting sensor networks and applications, Harvard technical report TR-21-04
14. Aman K, Suman N, Jie L, Feng Z (Oct–Dec 2007) SenseWeb: an infrastructure for shared sensing. IEEE Multimedia 14(4):8–13
15. Chatzigiannakis I, Mylonas G, Nikoletseas S (2007) The Design of an Environment for Monitoring and Controlling Remote Sensor Networks. Int. J. Distrib. Sen. Netw. 5, 3 (July 2009), 262–282
16. Bernat J, Pérez S, González A, Sorribas R, Villarrubia L, Hernández L (June 2008) Ubiquitous sensor networks in IMS: an ambient intelligence telco platform. ICT Mobile Summit
17. Gluhak A, Schott W (2007) A WSN system architecture to capture context information for beyond 3g communication systems. In: In Proceedings of the third international conference on intelligent sensor, sensor networks and information processing (ISSNIP) 2007, Melbourne Australia, 3–6 Dec 2007
18. Aberer K, Hauswirth M, Salehi A (2007) Infrastructure for data processing in large-scale interconnected sensor network, In Proceedings of 8th International Conference on Mobile Data Management (MDM), Mannheim, Germany, May 2007
19. Botts M, Percivall G, Reed C, Davidson J (eds) (2007) OGC sensor web enablement: overview and high level architecture, White Paper Version 3, Open geospatial consortium Inc., 27 Dec 2007
20. CoBIs Final Project Report (Mar 2007) Deliverable D104, Version 2.0. http://www.cobis-online.de/files/Deliverable_D104V2.pdf. Last visited 2 Sept 2008

21. Krco S, Johansson M, Tsiatsis V (2007) A commonsense approach to real-world global sensing. In: Proceedings of the SenseID: convergence of RFID and wirelesssensor networks and their applications workshop. ACMSenSys 2007, Sydney, Australia, Nov 2007
22. Harrop P, Das R (2008) Active RFID and sensor networks 2008–2018. IDTechEx research report, Feb 2008
23. ITU (2007) TSTAG: A preliminary study on the Ubiquitous Sensor Networks. TSAG-C 22-E. Feb 2007
24. Marin-Perianu M, Meratnia N, Havinga P, Moreira Sa de Souza L, Muller J, Spiess P, Haller S, Riedel T, Decker C, Stromberg G (Dec 2007) Decentralized enterprise systems: a multiplatform wireless sensor network approach. IEEE Wireless Commun 14(6):57–66