Decentralized Enterprise Systems: A Multi-Platform Wireless Sensor Networks Approach

Mihai Marin-Perianu, Nirvana Meratnia, Paul Havinga, University of Twente
Luciana Moreira Sá de Souza, Jens Müller, Patrik Spieß, Stephan Haller, SAP
Till Riedel, Christian Decker, University of Karlsruhe
Guido Stromberg, Infineon Technologies

***Abstract*— Massively deployed wireless sensor and actuator networks, co-existing with RFID technology, can bring clear benefits to large-scale enterprise systems, by delegating parts of the business functionality closer to the point of action. However, a major impediment in the integration process is represented by the variety of customized platforms and proprietary technologies. In this paper, we present a three-layer service-oriented architecture that accommodates different sensor platforms and exposes their functionality in a uniform way to the business application. Our work is motivated by real business cases from the oil&gas industry. In our implementation, we use three sensor platforms (Particle, μNode and Sindrion) integrated through the Universal Plug and Play (UPnP) standard and incorporated into an enterprise software system. The practical tests and application trials confirm the feasibility of our solution, but also reveal a number of challenges to be taken into account when deploying wireless sensor and actuator networks on industrial sites.**

***Index Terms*— Wireless sensor and actuator networks, RFID, enterprise systems, industrial and business processes**

I. Introduction

Sensing and actuating represent nowadays major functionality when describing the vision of pervasive computing. Made possible by the proliferation of wireless technologies and the advances in manufacturing low-cost, low-power devices, massively deployed wireless sensor and actuator networks (WSN and WSAN) [5] target a large number of applications, ranging from smart spaces [4] to industrial processes [2], and even planetary sensing or space exploration [10]. The enthusiasm generated by these countless possibilities has led in recent years to an outbreak of diverse sensor network platforms, covering both the hardware and the software. Without claiming to give a complete taxonomy, we can identify the following three broad classes of sensor nodes currently in development:

Class 1 – Tiny, cheap, energy-constrained, numerous devices, illustrating the vision of Smart Dust [12]. Application domains include environmental monitoring, battlefield and logistic processes.

Class 2 – Multi-functional, user-centric, rechargeable devices, covering health care, games and sports, as well as various mobile applications [9].

Class 3 – Powerful, reliable devices, approaching the capabilities of an embedded computer [13] and targeting applications with strict requirements such as industry and military.

Today's users are easily overwhelmed by the number of options available when they have to choose the right technology for their application. Due to the unique challenges of WSN [8], the platforms are typically specialized for specific purposes (e.g. data collection, target tracking), so it is often the case that complex applications require the combination of multiple proprietary technologies and customized platforms. As a result, the users are confronted with a considerable amount of low-level programming, tuning and tedious testing. Furthermore, the management, monitoring and administration of a system with highly distributed logic is a very complex task. Without the right tools and architecture, it can increase the total cost of ownership to a point where the deployment of this technology becomes commercially uninteresting.

A service-oriented architecture (SOA) is helpful in solving these issues. The integration efforts are minimized by hiding much of the implementation details and exposing only the functionality of the WSN in use. The management is also simplified because the logic is encapsulated in services with a manageable granularity. The services can be deployed, removed or upgraded from a central location in order to adapt the system to the business needs.

In this paper, we focus on the integration of various WSN platforms in large-scale enterprise environments. Service-oriented architectures based on Web Services technology have recently become popular for building complex yet flexible enterprise systems [17]. However, taking the SOA concept to the level of distributed embedded devices represents an intricate problem [7]. Even if sensor nodes of Class 3 may have the necessary resources, the ones belonging to Classes 1 and 2 are clearly too limited for such a complex task. Consequently, the efforts of the WSN community in this direction are still incipient and are focused on small-scale settings.

In what follows we propose a three-layer SOA designed for large scale, heterogeneous WSN. Motivated by real business cases identified at BP premises in UK, we utilize three sensor platforms (see Sec. III) specialized on specific tasks: dynamic networking under mobility conditions, large-scale infrastructure and co-existence with RFID. In order to leverage the effort of the resource-constrained sensor nodes (i.e. Classes 1 and 2), the platform gateways expose the WSN functionality to the business applications in a uniform way, i.e. by using the Universal Plug and Play (UPnP) standard. The high-

Fig. 1. Application trial in Hull, UK: Chemical containers stored in a warehouse (left); Detail with a sensor node attached to a container (right).

level business logic and management are implemented using the SAP Web Application Server (WebAS) and incorporated within the SAP enterprise software.

II. Application Scenario

The application scenario that motivated this work comes from the oil&gas industry. Every year, the U.S. Department of Labor registers numerous occupational accidents in this field (e.g. only in 2004, there have been recorded 52,830 nonfatal injuries and 29 fatalities due to the exposure to harmful substances or environments). WSN represent a viable solution to this problem. Sensor nodes can collaboratively determine potential hazardous situations, and alert or take an action at the point of interest [16]. In addition, the combination of WSN and RFID technology can prevent errors in the manipulation and storage of chemical containers, leading thus to increased safety and reduced logistic costs. In the EU-funded project CoBIs (Collaborative Business Items) [2], we extend these ideas by designing and implementing a distributed, service-oriented enterprise system, which incorporates the latest advances in WSN technology. The field tests were carried out at a chemical plant of BP in Hull, UK (see Fig. 1), where the following use cases had been identified:

Storage and manipulation of hazardous substances. Chemical containers storing reactive substances must be handled according to a strict list of safety regulations. The following situations are to be avoided: *(i)* storing incompatible substances in close proximity of each other, *(ii)* exceeding the maximum storage volume threshold for any hazardous substance and *(iii)* storing hazardous substances in temporary, unprotected areas longer than a specific maximum time period.

Continuous monitoring of environmental conditions. At the site of a chemical plant, environmental parameters, such as temperature, humidity and light, should be continuously monitored and abnormal conditions should trigger immediate alarms and local actions.

Smart shelves in warehouses with chemical containers. RFID readers placed on the shelves of the warehouses can improve significantly the tracking and identification of various objects (chemical containers, tools, etc.), already outfitted with RFID tags. To overcome the major problem of RFID reader interference, sensor nodes attached to the readers can agree on a non-overlapping subset of readers that are switched on.

These three use cases led us to an integrated approach, since there was no single WSN platform that could optimally fulfill all the tasks. Consequently, we opted for using three sensor platforms, namely *Particle* (produced by Particle Computer), *μNode* (produced by Ambient Systems) and *Sindrion* (produced by Infineon Technologies). The prototype sensor nodes are shown in Fig. 2. In the following section, we give a brief technical overview of the three platforms, indicating the specific services that each of them delivers within the given scenario.

III. Sensor Network Platforms

The Particle node [3] comprises a communication board with the PIC18f6720 microcontroller and TR1001 transceiver. Various types of sensors can be attached to the communication board. The wireless communication uses the AwareCon protocol [6], which is designed to handle high mobility and density of nodes. This makes the Particle platform well suited for equipping chemical containers handled by human operators and checking potential dangerous situations, as described in use case 1 of our scenario.

The μNode platform [1] represents a low-power, general purpose sensor node, built around the MSP430 microcontroller and a single-chip radio transceiver for the 433/868/915 MHz ISM band. After deployment, the μNodes self-organize into a multihop network, through which data can be routed back and forth to a designated sink node. This platform is ideal for building large scale sensing infrastructures, which can function unattended for long periods of time. Since many chemicals must be stored under specific ambient conditions, we use the μNode sensors for continuously monitoring environmental conditions, as described in use case 2 of our scenario.

The Sindrion platform [11] comes with native support of standard networking protocols (e.g. DHCP, IP, UDP, TCP, HTTP) on the sensor nodes. The nodes comprise an Infineon TDA 5250 868MHz transceiver and a 16-bit Infineon microcontroller. The Sindrion nodes integrate natively into the IT infrastructure by a network adapter attached as a USB dongle. In contrast to the other platforms, the Sindrion nodes feature standard UPnP discovery and description. In addition, the Sindrion nodes are designed to be tightly coupled to RFID readers. The combination represents a smart ISO15693 RFID reader, which negotiates the access to the wireless medium in order to prevent interferences and exposes itself in the network as a full-blown UPnP device providing service-based functionality. Consequently, this system implements the "smart shelves" concept, as described in use case 3 of our scenario.

Fig. 2. The three sensor platforms: Particle, μNode and Sindrion.

IV. Architecture Overview

In this section, we define a three-layer SOA, which is intended to provide a bridge between the business applications and the underlying sensor and actuator networks. The three layers – the back-end, the gateway and the front-end layers – are illustrated in Fig.3 and discussed in detail throughout the following subsections.

A. The Back-end (or Application) Layer

The business applications should be able to access the services offered by the WSN at the level of Web Services. In order to achieve this goal, the back-end layer benefits from the uniform interfaces offered by the gateway layer. The components of the back-end layer are therefore implemented completely independent from the underlying platforms. As shown in Fig.3, we distinguish the following high-level components:

Service Repository – contains a database of the available services, including their description and implementation. A single service can have several implementations (or executables) on different platforms. The service descriptions are XML documents formatted in CoBIL (an acronym for CoBIs Language, see [2] for the XML schema). A CoBIL description starts with the service interface, i.e. the set of operations that all the service executables offer and the events they generate. The interface definition is derived from WSDL (Web Service Description Language). For each executable, the CoBIL description contains specific information, such as the reference point, the target platform, etc. In the end, the technical requirements to be used at deployment time are specified (e.g. which sensor and actuators the target nodes have to feature, the minimum remaining energy, the necessary network bandwidth, etc.).

System State Manager – stores the operational state of the nodes, such as the services running, resources available, current position information, etc.

Service Mapper – performs the systematic mapping of the services to the nodes, based on the service descriptions (technical requirements, composition constraints) and the system state. The Service Mapper is used during service deployment (see Sec. V-A).

Service Invocation Manager – processes the service invocations issued by the business application running on the back-end. More specifically, the Service Invocation Manager contacts the node(s) executing the specified service and retrieves the results of the service invocation back to application.

Notification Manager – implements a web service interface for distribution of event messages. Interested client applications can register with the Notification Manager and will receive notifications for all relevant event messages.

B. The Gateway (or Platform Abstraction) Layer

The gateway layer has an essential role in harmonizing different sensor platforms. We opted for the UPnP standard as the uniform interface between the application layer and the underlying WSN. In recent years, UPnP has been widely accepted as a simple and robust standard for ad-hoc and unmanaged networks. Being designed to support zero-configuration and automatic discovery for a breadth of devices from different vendors, UPnP facilitates the integration of new platforms via simple standardized mechanisms. In our case, the Sindrion platform already implements a basic UPnP interface, which makes it capable of connecting directly to the back-end layer. However, the Particle and μNode platforms are too resource constrained to run natively UPnP. It is therefore the responsibility of the gateway layer to handle the proprietary WSN mechanisms and expose the service-oriented functionality through a standard UPnP interface.

With respect to our reference architecture, the gateway layer provides the following functionality:

Message Transformation – handles the packet-level translation between the proprietary WSN messages and UPnP arguments.

Service Lifecycle Manager – assists the deployment of new services in the network. Deployment requests are issued to the gateway by specifying the service executable and the XML-based deployment description.

The key feature of the gateway is the dynamic instantiation of service proxies. Service proxies can be accessed like native UPnP devices, providing detailed service descriptions for the implemented functionality. However, the service proxies only exist as virtual representations of the service interfaces. The gateway transforms the requests issued to the service proxies into WSN messages and vice-versa. In addition, UPnP handles service discovery natively once a proxy is initiated. This means that service proxies have to be instantiated whenever a new service is provided by the WSN and destructed when the service becomes unavailable.

C. The Front-end (or Device) Layer

The device layer encompasses the multitude of WSN and RFID technologies. The design guidelines are driven by the

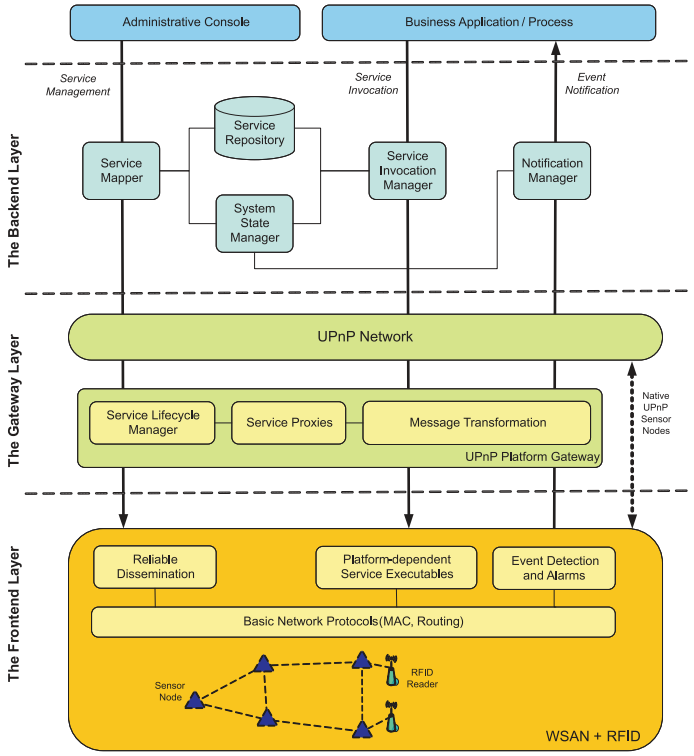

Fig. 3. Three-layer service-oriented architecture.

requirements of the industrial applications and the typical constraints in terms of energy, bandwidth, computational power and storage. As a result, the components developed within the device layer have to meet the following objectives: (1) energy efficiency (both at the node level and the network level), (2) responsiveness, (3) reliability, (4) scalability, (5) reconfigurability and (6) usability. In Fig. 3, we highlight the most relevant components with respect to the service integration:

Reliable Dissemination – enables the service deployment and updating. It works on top of the unreliable WSN communication protocols and guarantees that the new service executables are transferred correctly to the target nodes. Due to the heterogeneity of the deployed WSN, the reliable dissemination provides multicast support for addressing selectively groups of nodes, being thus more scalable than unicast or flood-based solutions (see Sec. VI-B). In addition, it works in conjunction with the MAC layer in order to reduce the communication and idle listening to a minimum. From the design perspective, the reliable dissemination component fulfills therefore objectives 1, 3, 4 and 5.

Platform-dependent Service Executables – represent running instances of the services, which can be invoked by the business application in order to execute specific tasks or retrieve a certain information. The platform-dependent services that we developed include synchronization, proximity, localization, data aggregation and querying. With respect to the design guidelines, these services relate mainly to objectives 1, 4 and 6.

Event Detection and Alarms – target timely and reliable detection of special situations that should be reported to the central system. In addition, the sensor nodes may signal and handle locally these situations, for increasing the overall responsiveness. Compact rule engines (see Sec. VI-A) are shown to be an effective solution to objectives 2 and 6, as they involve minor overhead at runtime while facilitating significantly the task of the business application developer.

Basic Network Protocols – form the platform-specific communication stack. Considerable importance is given to the MAC protocols, which contribute directly to the fulfillment of objectives 1, 2, 3 and 4.

For the detailed design of all the components within the device layer, as well as the performance evaluation and implementation results, the reader is referred to the public deliverables of the CoBIs project [2].

V. Functional Overview

This section gives a functional overview of the entire system. More specifically, the three main operations – service lifecycle management, service invocation and event notification – are explained with respect to the reference architecture (see the thick arrow flows in Fig. 3).

A. Service Lifecycle Management

Service Lifecycle Management is responsible for three major administrative tasks: the deployment, update and removal of services. For the sake of brevity, we describe only the service deployment; the other two operations bear clear similarities in their work flows. The deployment of a new service is initiated by the administrator command in the Administrative Console running on the back-end. Firstly, the Service Mapper creates a service mapping, based on the current state of the

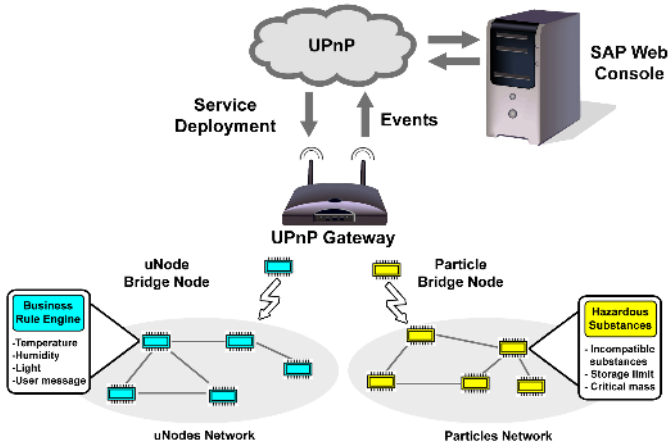

Fig. 4. The complete system using a hybrid setting with both Particle nodes and μNodes.

system (retrieved from the System State Manager) and the technical requirements specified in the CoBIL description of the service (from the Service Repository). As a result, the nodes (identified by individual IDs, group ID or just a geographical area) where to instantiate the service are chosen. Secondly, the deployment request is sent to the Service Life-cycle Manager counterpart on the UPnP gateway. The gateway uses the URI (Uniform Resource Identifier) of the service executable and the executable-specific information from the service description in order to launch the platform-dependent deployment. Thirdly, the new executable is transferred to the WSN, in our case using a reliable multicast dissemination protocol (see Sec. VI-B). Finally, the result of the deployment (success/error) is sent back to the Administrative Console. The log of the dissemination session in the WSN is stored on the gateway for post-analysis purposes.

B. Service Invocation

Service invocation is the key operation through which the business applications running on the back-end use the functionality of the WSN. To initiate this process, the business application issues a service invocation request to the Service Invocation Manager, which in turn retrieves the service description from the Service Repository, in order to validate the parameters and check the service requirements. After identifying the nodes that offer the service by consulting the System State Manager, the service invocation request is delivered to the gateway layer. On the gateway, the UPnP - WSN transformation takes place. For this purpose, a transformation description is associated with each UPnP service description. The descriptions are automatically parsed by the UPnP stack, which already provides the RPC (Remote Procedure Call) dispatching and eventing facilities. For specifying the transformations we use a simple template-based transformation mechanism that works bi-directionally. When a RPC call is received, each outgoing argument is serialized via the template-based transformation to the UPnP state variable and the assembled message is then sent to the WSN. The addressed nodes execute the service and issue a response message, or a timeout/error message. The response messages from the WSN are converted to the according arguments by inverting the template-based transformation process. In the Sindrion platform, however, application-specific proxies are optionally made available directly from the sensor nodes. In this case, the message transformation is not required, since the Sindrion programming environment supports the native development of UPnP device representations.

C. Event Notification

An important functionality of the WSN is the ability to detect and signal events. In this case, the business applications represent event consumers, which subscribe to the Notification Manager component and provide a filter of the relevant events. The sensor nodes detecting the event send notification messages toward the gateway. After being routed in the WSN, the messages arrive at the gateway, which performs the WSN – UPnP transformation that leads to a UPnP state variable change. UPnP's General Event Notification Architecture (GENA) handles the events and transmits them to the Notification Manager. The Notification Manager selects the consumers that have subscribed for the specific events, and distributes them accordingly. In addition, it publishes the notifications to the System State Manager for updating the system operational state.

VI. Implementation and Testing

The previous sections provided a general overview of the integrated SOA, abstracting from the underlying hardware and software platforms. In this section, we discuss the most relevant implementation details and present the operation of our system with a hybrid setting using both Particle and μNode sensor nodes (see Fig. 4). For brevity, we focus on these two platforms because they illustrate best the abstraction of the platform gateway, in contrast with the Sindrion platform, which is directly accessible through the standard UPnP mechanisms.

As gateway device, we utilize an off-the-shelf WLAN router (Asus WL-550g) running OpenWRT (Linux derivative for embedded devices), on which we implement the Cyberlink UPnP stack. The connection to the WSN communication

protocols is achieved by connecting both Particle and μNode bridge nodes to the USB ports of the router. In this way, we can build a cost-effective hybrid gateway that can communicate with both sensor network platforms.

The components of the back-end layer are implemented as Web Services and deployed on a SAP Web Application Server. They are developed in Java for portability, using SAP NetWeaver Developer Studio.

In the following, we present the experiments and application trials that we performed according to the scenario described in Sec. II.

A. Hazardous Substances and Environmental Monitoring

The first setting illustrates the use cases 1 and 2 of our scenario, namely handling hazardous substances and monitoring the environmental conditions. The first of these tasks is implemented on the Particle nodes and the latter on μNodes. Both platforms use the UPnP hybrid gateway to receive requests from the back-end and report the events generated within the network.

The Particle nodes execute the "hazardous substances" service in order to alert about situations that do not comply with the safety regulations given by the oil&gas company (BP in our case). The alarms are triggered locally by the nodes placed on the chemical containers and are reported to the back-end application (along with the location information). The overall responsiveness of the service has to be less than 2 seconds.

In our setting, the Particle nodes use the AwareCon [6] communication protocol, which is a TDMA-based protocol. The timeframe is divided in 70 slots, each of 13ms. The duty cycle is set to 35% (25 slots). Therefore, it can be guaranteed that the delay for detecting a hazardous situation is less than 2 seconds. In order to detect the situations where incompatible substances come in close proximity of each other, the Particle nodes are equipped with TSOP IR receivers and implement a diffuse infrared-based location method. The main advantages of this solution are the low signal processing overhead and the low power operation. The disadvantages concern the disturbances from occluding objects and direct sun light. Depending on these conditions, the accuracy lies between 30cm-1m. The "hazardous substances" service is loosely coupled to the location service and takes its input to compute storage incompatibilities and limits. Limits and incompatibility classes can be configured via the service update interface from the back-end system. The nodes update the storage status within the surrounding space by broadcasting their information during the wake phase of the AwareCon protocol. The alarms are signalled collectively by flashing visible LEDs, and reported to the back-end system. The resources needed on the Particle nodes are 746 bytes of FLASH memory and 10 bytes of RAM for the location service, and 8.5 kB of FLASH memory and 242 bytes of RAM for the "hazardous substances" service, respectively.

Monitoring the environmental conditions for potential dangerous situations is implemented through the business rules [15] support of the μNodes. The business rules for sensor nodes express simple business logic in a compact and efficient way, by following the execution chain *Observe – Check rules – Take action*. A simple example is the following rule: "Measure humidity level H at rate r; if it is outside the interval $[H_{min}; H_{max}]$, launch alarm service S_{alarm}". This approach is well suited to our scenario, as it minimizes the network traffic and prolongs the network lifetime by reacting only to the situations that do not comply with the rule set. Complex conditions can be expressed by forming chains of rules, logically linked through a *next rule* field. In addition, the lifetime of a rule can be set by specifying a certain *running time* value. Our implementation results show that the business rules are very compact (20 bytes each) and easy to write even by inexperienced users.

In our setting, the μNodes are equipped with the following: internal voltage and temperature sensors, a light dependent resistor (LDR), a combined temperature and humidity sensor (SHT), and a push-button. A typical rule set for this setting amounts to approximately 10 rules (200 bytes), which means little network overhead in the reconfiguration phase. The rule engine is also very lightweight, with less than 1 kB code memory footprint. The average sensing driver size is $\approx$ 300 bytes, whereas the average action service module size is $\approx$ 130 bytes. In total, a typical business rule-based program requires only $\approx$ 3.5 kB, which represents 7.3% of the available FLASH memory.

As depicted in Fig. 5a, the integration of the Particle nodes and μNodes is successful. Events (hazardous substances and business rules alarms) from both platforms reach the back-end (a Web-based console in this case) through the UPnP gateway. Sensor readings, e.g. the light level, can also be obtained, by invoking the corresponding service. The integration process is completely transparent to the back-end layer.

B. Service Deployment

Dynamic reconfiguration of the sensor network at runtime directly supports the service deployment in our reference architecture (see Sec. V-A). Due to the critical reliability and scalability requirements, a specialized protocol must be used within the WSN, which guarantees data delivery with minimal energy expenditure. We use RMD [14], a multicast-based dissemination protocol that supports reconfiguration of groups of sensor nodes at scale. RMD is a cross-layer solution, utilizing MAC layer information about neighborhood and packet losses. Moreover, RMD controls the MAC protocol in order to reduce idle listening to a minimum. Compared to other transport protocols for WSN, RMD ensures data delivery to all recipients even under high error rates, while consuming less energy and maintaining a comparable delay. Due to the cross-layer design, the resources demanded by RMD are very low: 2.7 kB of FLASH memory for the code and $\approx$ 190 bytes RAM for internal data.

In RMD, the gateway acts as the source of the reconfiguration message (the new service executable in the case of service deployment). After the initial phase, when the reconfiguration is announced in the network and the multicast tree is reinforced, the source starts sending the message divided in

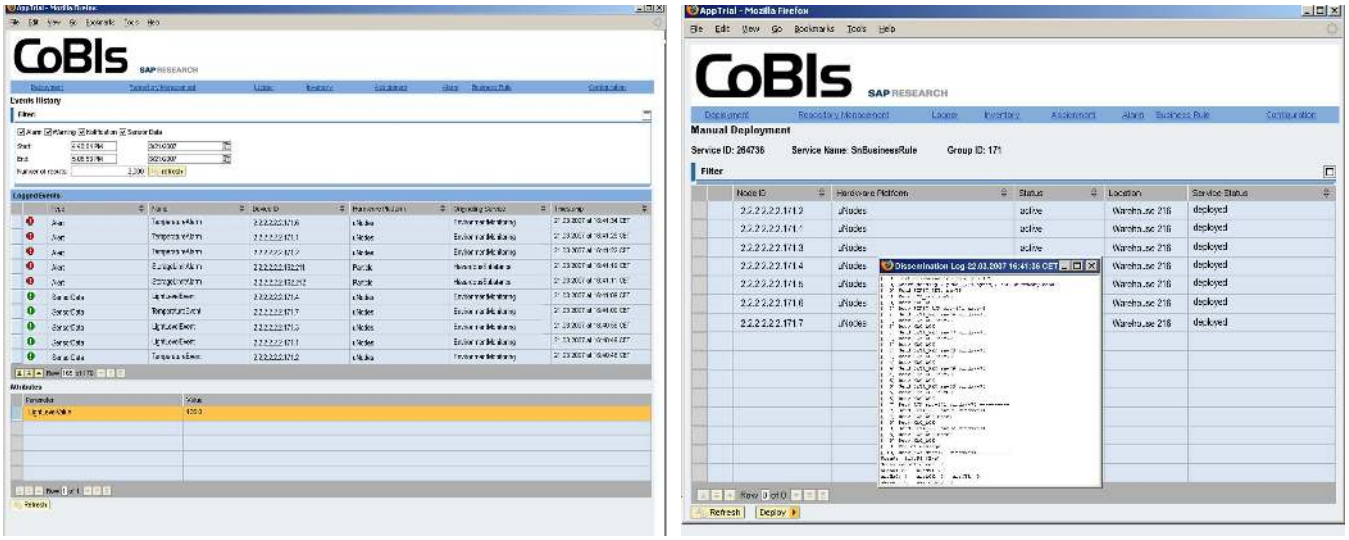

Fig. 5. Web console: Events from both Particle nodes and μNodes (left); A new service is being deployed in the WSN (right).

windows of packets. The packets are further pipelined down the tree by the intermediate nodes, which are responsible for the local error detection and retransmissions.

Figure 5b shows the successful deployment of a service based on business rules. The target nodes are identified by their group ID. The detailed log of the dissemination session (including the average speed, number and types of errors, etc.) is stored on the gateway and can be retrieved in the web console for analysis purposes.

C. Application Trials

To test our system in a realistic environment, we conducted two application trials at a BP petrochemical plant in UK. In both trials we equipped 20 chemical containers with wireless sensor nodes and distributed them in three different locations. A gateway was installed at each location, in order to connect the WSN to the WLAN and further to the back-end server. Each trial lasted about four weeks and consisted of two phases: the supervised phase (installation and manual testing) and the unattended phase (normal operation on-site). The goals of the first trial were to check if the WSN and the back-end system react correctly to real situations, and to measure the scalability of the overall solution. The second trial focused on improving the stability and scalability of the back-end system, and on prolonging the battery lifetime of the sensor nodes.

Both application trials confirmed the feasibility of our solution. During the second trial, for example, the system handled successfully a total number of 162294 messages from the sensor nodes, with an average message load of ≈ 7 messages/minute. The last day of the trial was allocated for testing under stress conditions, by placing all the 20 containers in the same location and generating continuously alarms. As a result, the average message load increased to 212 messages/minute, with a peak rate of 225 messages/minute. Even under these conditions, the WSN continued to operate reliably. However, the stress test revealed a scalability limitation of the UPnP eventing. The UPnP GENA uses TCP connections for eventing on subscriptions, which leads to a complete TCP connection setup on each event. Unless the client implements HTTP/1.1 pipelining and reuses callback sockets for multiple services, there is no way of reducing the overhead of the UPnP eventing. In the following, we briefly list other problems encountered during the application trials. Firstly, the UPnP implementation that we used (Cyberlink) is intended to serve a few services and therefore scales poorly for productive usage, by overwhelming the gateway with excessive multi-threading. Secondly, the real deployment conditions, such as the weather conditions, can adversely affect the performance of the system. For example, during the application trials, the 802.11 network exhibited high packet loss at times due to rather high humidity. This affected the UDP traffic of the UPnP discovery operation and the DHCP-based dynamic addressing for the routers. Finally, when dealing with safety-critical scenarios, the costs of intrinsically safe equipment adds a high factor concerning packaging and quality control. Even if the WSN technology is ubiquitous and cheap, these costs remain rather constant, since strict guidelines have to be fulfilled, such as the directive 94/9/EC (Equipment intended for use in potentially explosive atmospheres – ATEX).

VII. Conclusions

In this article we address the integration of ubiquitous technologies into decentralized enterprise environments. The ultimate goal is to delegate well-defined parts of the business logic to the low-cost embedded devices, and thus reduce the process execution costs and improve the response time in safety-critical situations. We present a layered service-oriented solution that accommodates three different sensor platforms and exposes their functionality seamlessly through the UPnP standard to the business process based on SAP enterprise software. Such a service-oriented approach proves to be beneficial both at design and implementation. The system can be designed top-down, with business services of coarse granularity broken down into lower-level system services. Thereby, the communication between the business process

experts and the technical experts is simplified and allows for a separation of concerns.

The practical tests and application trials confirm the feasibility of our solution. The multi-platform integration provides versatile functionality to the user, in a uniform way. The local, collaborative execution of tasks within the WSN reduces the load on the back-end and improves the overall responsiveness. Energy-efficiency remains a major concern for the battery-powered sensor nodes. This is why low duty cycle operation, lightweight execution models (such as business rules) and cross-layer solutions (such as RMD) should be favored. The scalability of the gateway layer and the integration with the back-end also represent important challenges. Consequently, we consider as future work to evaluate DPWS (Devices Profile for Web Services) as an alternative to UPnP.

VIII. Acknowledgments

This work has been partially sponsored by the European Commission as part of the CoBIs project (IST-004270).

References

[1] Ambient Systems. http://www.ambient-systems.net.
[2] Collaborative Business Items (CoBIs). http://www.cobis-online.de.
[3] Particle Computer. http://www.particle-computer.de.
[4] Smart Surroundings. http://www.smart-surroundings.nl.
[5] I. Akyildiz and I. Kasimoglu. Wireless sensor and actor networks: Research challenges. *Ad Hoc Networks Journal*, 2(4):351–367, 2004.
[6] M. Beigl, A. Krohn, T. Zimmer, C. Decker, and P. Robinson. Awarecon: Situation aware context communication. In *UbiComp*, pages 132–139, 2003.
[7] C. Bornhövd, T. Lin, S. Haller, and J. Schaper. Integrating smart items with business processes: An experience report. In *International Conference on System Sciences (HICCS)*, 2005.
[8] C.-Y. Chong and S. P. Kumar. Sensor networks: evolution, opportunities, and challenges. *Proceedings of the IEEE*, 91(8):1247–1256, 2003.
[9] S. Eisenman, N. Lane, E. Miluzzo, R. Peterson, G. Ahn, and A. Campbell. Metrosense project: People-centric sensing at scale. In *ACM SenSys Workshop on World-Sensor-Web (WSW)*, 2006.
[10] E. Gaura and R. Newman. Wireless sensor networks: The quest for planetary field sensing. In *SenseApp*, 2006.
[11] Y. Gsottberger, X. Shi, G. Stromberg, T. Sturm, and W. Weber. Embedding low-cost wireless sensors into universal plug and play environments. In *1st European Workshop on Wireless Sensor Networks (EWSN)*, pages 291–306, 2004.
[12] J. M. Kahn, R. H. Katz, and K. S. J. Pister. Next century challenges: mobile networking for smart dust. In *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 271–278, 1999.
[13] R.M. Kling. Intel motes: advanced sensor network platforms and applications. In *IEEE MTT-S International Microwave Symposium*, 2005.
[14] M. Marin-Perianu and P.J.M. Havinga. RMD: Reliable multicast data dissemination within groups of collaborating objects. In *Local Computer Networks (LCN)*, pages 656–663, 2006.
[15] M. Marin-Perianu, T.J. Hofmeijer, and P. J. M.Havinga. Implementing business rules on sensor nodes. In *11th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 292–299, 2006.
[16] M. Strohbach, H.-W. Gellersen, G. Kortuem, and C. Kray. Cooperative artefacts: Assessing real world situations with embedded technology. In *Ubicomp*, pages 250–267, 2004.
[17] D. Woods and T. Mattern. *Enterprise SOA Designing IT for Business Innovation*. O'Reilly, 2006.