WISEBED: An Open Large-Scale Wireless Sensor Network Testbed

Ioannis Chatzigiannakis[1], Stefan Fischer[2], Christos Koninis[1], Georgios Mylonas[1], and Dennis Pfisterer[2]

[1] Computer Technology Institute and University of Patras, N. Kazantzaki, Rio, Patras, Greece
{ichatz,koninis,mylonasg}@cti.gr
[2] University of Lübeck and Institute of Telematics, Ratzeburger Allee 160, Lübeck, Germany
{fischer,pfisterer}@itm.uni-luebeck.de

Abstract. In this paper we present an overview of WISEBED, a large-scale wireless sensor network testbed, which is currently being built for research purposes. This project is led by a number of European Universities and Research Institutes, hoping to provide scientists, researchers and companies with an environment to conduct experiments with, in order to evaluate and validate their sensor network-related work. The initial planning of the project includes a large, heterogeneous testbed, consisting of at least 9 geographically disparate networks that include both sensor and actuator nodes, and scaling in the order of thousands (currently being in total 550 nodes). We present here the overall architecture of WISEBED, focusing on certain aspects of the software ecosystem surrounding the project, such as the Open Federation Alliance, which will enable a view of the whole testbed, or parts of it, as single entities, and the testbed's tight integration with the Shawn network simulator. We also present examples of the actual hardware used currently in the testbed and outline the architecture of two of the testbed's sites.

Keywords: WISEBED, testbed, sensor network, large-scale, experiment, open, federated, portal, web services, heterogeneous, actuators, software library, simulation.

1 Introduction – Motivation

In the last few years, we have begun to witness the effects of turning a vast number of heterogeneous objects into one large and decentralized network, as a result of a growing trend to interconnect the natural and the digital worlds. A variety of methods and technologies is being used to realize the Internet-of-things vision, where myriads of networked devices will allow the provision of ubiquitous computing services and closer inspection of the physical domain. Still, the large-scale effects of the interaction between hardware, software, algorithms and data are just starting to show, and many of the resulting emerging phenomena often come as a surprise, rather than by design.

N. Komninos (Ed.): SENSAPPEAL 2009, LNICST 29, pp. 68–87, 2010.

Until very recently, scientific efforts for studying computing methodologies for decentralized complex systems have been very limited. A particularly promising and active research area, in this context, is *wireless sensor networking* (WSN), which is attracting researchers from very different backgrounds, such as hardware, software, algorithms, etc., as well as researchers from various application areas. So far, the number and size of actual testbeds for sensor networks has been rather limited. All of these efforts have been struggling with a number of different issues:

- **Hardware.** Developing/acquiring small-scale devices for sensor networks is a tedious/expensive task.
- **Software.** Dealing with the limitations of small-scale special-purpose computing devices makes it very challenging to develop appropriate software.
- **Algorithms.** Dealing with the challenges of designing algorithms for well-organized, large-scale distributed systems requires new algorithmic methods.
- **Data.** The large volume of collected sensing information, as well as the communication overhead gives rise to huge amounts of data.

WISEBED [1] is a European research project that will try to overcome some of these impediments by building a network of networks. More specifically, it will expose a network of heterogeneous sensor network testbeds, together with a unified and universal approach to software, algorithms, and data. Connecting sensor networks to the Internet creates endless opportunities for applications and services, new emerging models of operation. Users will be able to get real-time data from the physical world for *everything, everywhere and anytime*. To make such a vision a reality, be effective and produce applicable results, it is important to encourage interaction and bridge the gap between fundamental (theoretical) approaches and technological/practical solutions.

WISEBED, in general, involves the following long-term actions:

- Deploy large numbers of wireless sensor devices of different hardware technologies in different types of terrains to use for evaluating and testing solutions at a large scale.
- Interconnect these wireless networks with the Internet and provide a virtual unifying laboratory to enable testing and benchmarking, in a controlled way, in different "real-life" situations. Researchers will be able to use the facilities remotely, thus reducing the need for a local, private testbed and, more importantly, reducing the cost for conducting all-rounded research.
- Operate the testbeds to collect traces of data from the physical environment and derive models of real-life situations and scenarios. These scenarios will be used to evaluate the performance of algorithms and systems and draw conclusions on their operation and how it can be improved.
- Provide a repository of algorithms, mechanisms and protocols and develop a library, called WISELIB, that can be directly used in experiments with WISEBED.

2 Previous Related Work

In this section we give a brief description of a number of existing wireless sensor network testbeds and related software environments. We chose to include only testbeds of significant scale or based on their unique characteristics (e.g., mobile nodes, open to the public, etc.). The characteristics we chose to highlight include total number of nodes, indoor or outdoor deployment, heterogeneity support, overall architecture and topology, openness to the public, total operational time.

Existing testbeds. The *Trio testbed* [2] was one of the largest wireless sensor testbeds, indoor and outdoor, built yet. The main target was to build and demonstrate a large-scale outdoor sensor testbed to be used in a multi-target tracking application. It consisted of 557 solar-powered motes, seven gateway nodes, and one overall testbed server. It was not open to the public research community, since it was targeted to a specific application. *MoteLab* [3] is an indoor sensor network testbed on the campus of Harvard University and is open to the public. It currently features 190 Tmote Sky sensor nodes. All sensor nodes are wired to programming boards allowing for direct reprogramming and communication. It was designed having in mind that the testbed should be both open and easy to use for other researchers, i.e., users from other research teams should have access for experimentation to real large-scale sensor networks. It provides a web-based interface for programming, debugging, and accessing data from the sensor network. The *TWIST* testbed [4] resides indoor in a building in the campus of the Technical University of Berlin, spanning across several floors. The total number of sensor nodes belonging to the testbed is 200, with heterogeneity supported to some extent. The *TutorNet* testbed [5] uses a 3-tier network topology (similar to TWIST but without any abilities to work in other topology modes) with testbed servers, gateway stations, and sensor nodes, which feature USB connections to the gateway stations. Services provided include remote programming of the nodes. Authorized-only users connect to the testbed servers and use command-line tools to control the testbed nodes. The total number of sensor nodes is approximately 100. *Intel SensorNet* [6] (now discontinued) is an indoor sensor network testbed that featured 100 MicaZ sensor nodes in the Berkeley Intel Research facilities. It allows resource allocation between multiple users submitting their jobs to be scheduled and executed in the sensor testbed. *Kansei* [7] is another sensor testbed targeted towards large indoor sensor networks. It currently features 210 sensor nodes, with 210 gateway stations attached to each one of the sensor nodes. It features a web-based interface for researchers, which allows for submitting jobs to the testbed, visualization of sensor readings, debugging and health monitoring, along with other sophisticated features. Only registered users can use this testbed. *TrueMobile* (Mobile Emulab wireless sensor network testbed, [8]) is an extension to the popular EmuLab wireless ad hoc networks testbed. The mobile testbed currently covers a total area of 60 m^2, and is situated indoor, and includes six mobile robots and 25 fixed Mica2 motes. EmuLab is an open testbed to the public (registered users). CitySense [9] is an urban (both indoor and outdoor) sensor network testbed, consisting of 100

wireless sensors deployed across a city, such as on light poles and buildings. Each node consists of an embedded PC with wifi and various sensors for monitoring weather conditions and air pollutants. CitySense is intended to be an open testbed. SENSEI [10] is another EU Research Project, aiming among other to provide a Pan-European test platform, enabling large scale experimental evaluation and execution of field trials - providing a tool for long term evaluation of the integration of sensor and actuator networks into the Future Internet. We should point out that, apart from TWIST and SENSEI, all the other testbeds are situated in the United States.

Related Software. One of the most important problems in designing applications for WSNs is the heterogeneity in hardware and operating systems, especially when talking about a distributed testbed operated in different domains. It makes sense to add an integration layer somewhere between hardware and application, called *middleware*. Creating middleware for WSNs is a challenge, mainly due to resource restrictions. We point the reader to a number of survey papers on WSN middleware [11,12,13,14,15].

Simulation is also an invaluable tool for evaluating protocol designs for complex systems such as WSNs, that are either comprised of a large number of interacting entities, systems that exhibit highly dynamic behavior or systems where directly performing an experiment is difficult due to cost or time constraints. A number of simulators have been developed and/or extended to allow the modeling and simulation of WSNs. Most notable examples of such simulators are *ns-2*, *OMNeT++*, *OPNET Modeler*, *GTNetS* and *YANS*. Some attempts to bridge the gap between simulation and real-world performance and to make the transition smoother have been proposed in international literature, such as *TOSSIM* [16], which takes advantage of TinyOS's structure to generate discrete-event simulations directly from TinyOS component graphs. The level of detail provided by these simulation tools is resource-demanding and limits simulations to small scenarios with only a few thousand of nodes while future scenarios anticipate networks with millions of nodes.

A very important aspect of developing an application for wireless sensor networks and deploying an operational sensor network is *testbed debugging*. Simulations and lab deployments in controlled environments cannot always reveal weaknesses and errors in the design and implementation of such applications, especially since the physical world can heavily influence the overall operation of these systems. Debugging refers both to software parameters and network parameters (mostly connectivity issues). There must also be an easy way to detect and track down problems during the normal operation of the network, i.e., after the end of the development cycle. The problems in the operation of a wireless sensor network can be classified into four categories: node problems, link problems, path problems, and global problems affecting large portions of the network.

Some examples of debugging software for WSN are: the *Sensor Network Management System* (SNMS, [17]), designed for debugging TinyOS applications; *Sympathy* [18], designed for sensor networks that follow a central gateway model;

Memento [19], an environment that provides, apart from failure detection, symptom alerts, i.e., reports on degrading performance and failures that may happen in the future according to certain symptoms; *Marionette* [20], a software environment that allows easy interaction with applications written in nesC, running in TinyOS-based sensor networks; and *Sensor Network Inspection Framework* (SNIF, [21]), which follows a more passive approach by using a separate backbone network that intercepts all data transmitted over the air inside the wireless sensor network. A software environment for managing sensor testbeds associated with the SENSEI project is SIGNETLAB [22]. Other interesting examples of software environments that focus on "replaying" activity inside the sensor network are LiveNet [23] and [24].

Our contribution: Existing wireless sensor testbeds are mainly deployed in indoor environments, only few have above 100 nodes, most are not publicly open to other researchers to test their own ideas, and there is little support for heterogeneity and mobility. In general, a tightly coupled network and software architecture is followed, thus limiting the extendibility, customize-ability and relocation ability of such testbeds. Our approach aims at overcoming several of these limitations:

- We plan to develop and deploy several connected testbeds, each with *several hundred nodes*.
- We plan to connect these testbeds into a much larger heterogeneous network (with a few thousand nodes), *creating the potentially largest sensor network testbed in the world*.
- This federation of testbeds is based on the concept of *testbed virtualization* and *virtual links* between them.
- The system will provide a *variety of interfaces* for end-users and applications.
- We aim at providing a *unified algorithmic and software environment*, thus overcoming the impediments of customization and allowing for convenient usage of the testbed.

3 Overall Architecture and Considerations

This section introduces WISEBED's general architecture (cf. Figure 1). We consider the WISEBED distributed testbed as a global network where human users, intelligent agents, and powerful computers interact with the wireless sensor network testbeds. In particular, we distinguish the global network into three sub-domains:

- The overlay network where peers are applications executing on powerful computer devices that have the ability to communicate with other peers via Internet or other global networks.
- The sensor devices that form wireless sensor network testbeds and communicate via the wireless medium.

– The portal servers that consist of nodes that control the wireless sensor network testbeds and allow for interaction between the overlay peers and the sensor devices.

Essentially, the architecture of the WISEBED system is based on a hierarchy of layers where each layer is comprised of one or more peers (see Figure 1). Each layer is assigned a particular role in the system. Each peer may be a traditional networked processor or a wireless sensor device.

– The bottom layer contains the wireless sensor nodes that are running iSense, Contiki, TinyOS, or legacy systems. These devices form wireless networks that constitute the WSN testbeds.
– The testbeds of each partner are controlled by *Portal Servers* that provide access and expose interfaces to manage and operate them. Users can connect to a single testbed directly via the Internet accessing the interface provided by the particular portal server. In order to do so, users must be aware of the public IP address of the individual portal servers.
– The portal servers of each testbed partner site are interconnected via an overlay network. Peers connecting to the overlay network may access one or more portal servers in order to use multiple testbeds in a distributed manner. In order to do so, users are not required to know the public IP address of the portal servers.

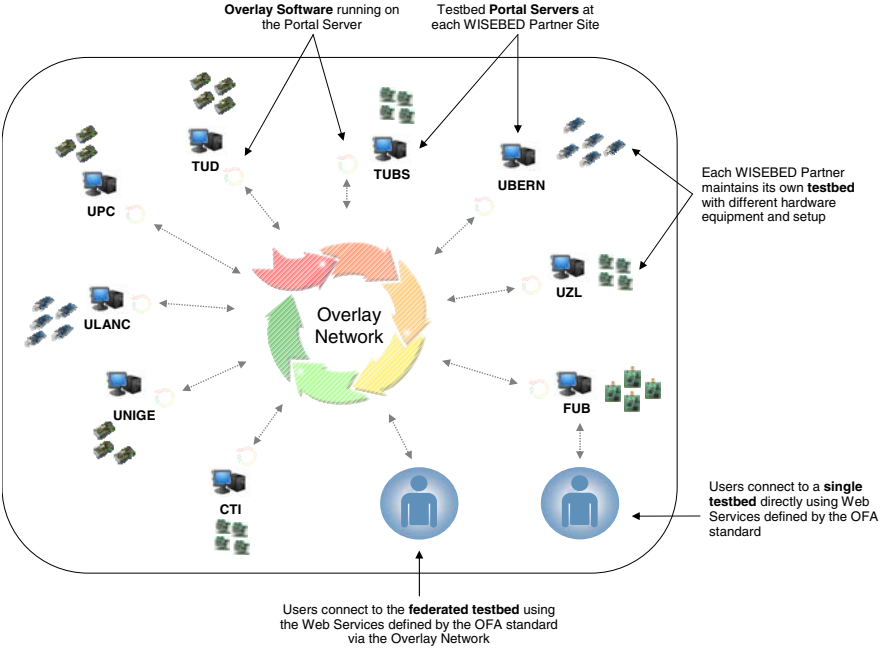

Fig. 1. Overall architecture of the WISEBED testbed federation

The advantageous distributed characteristics of the system are achieved via a series of functions and mechanisms (services) which are activated by the system as a response to various kinds of events, processes, actions, applications that take place on it. Rather than following a multi-tiered architecture, in which a number of tiers are operating in a specific hierarchical way and specific interfaces are used for communication across each layer, we choose to follow a less tightly coupled architecture in order to offer more flexibility to the system. *Interaction among services* is performed over all the levels of the system, in order to exchange necessary information for the successful accomplishment of each of them. Software agents (services) running on a peer are considered as independent modules that may interact with other services on the same peer and/or software agents executed by nearby peers.

This flexibility allows the overlay network to spontaneously federate multiple portal servers into a *Virtual Distributed Testbed* and expose their services as a single unifying virtual testbed. Thus, users connected to the overlay network can access the unifying distributed testbed in the same way they access individual testbeds via the portal servers. Due to the architecture of the overlay nodes and portal servers, the interfaces exposed look identical and therefore there is no need to re-write their code. In addition, the overlay network can partition specific nodes (not necessarily from the same testbed) and expose their services as a single unifying virtual testbed.

Furthermore, due to the heterogeneity of the devices but also to the very nature of such a global system for testbed interconnection, each peer may operate with a different set of software agents, i.e. provide a subset of the available services, and may provide different versions of a particular service, i.e. provide different quality and functionality. In this sense, we point out that:

- Each wireless device may operate a different set of software agents, i.e., provide a subset of services.
- Each wireless device may operate different versions of a particular service.
- Each portal server may operate under different implementation of a particular service that provides a subset of services.

The structure of the peers of our system follows a modular architecture of essentially two layers: the *inner layer* that is comprised of minimum set core functionalities and an *outer layer* that hosts a variety of services.

Portal Server. These peers are responsible for the control and management of the WSN testbed of a single partner. The *inner layer* includes services responsible for accessing the sensor devices via gateways to the wireless networks. User commands are translated to a binary packet format that is generic enough so that it can be implemented by the different hardware/software technologies available in WISEBED. Each portal server is connected to one or more local data stores (e.g., XML files, RDBMS systems, embedded databases etc.) for storing data retrieved from the database, debug traces, access lists etc. The *outer layer* implements a series of services that are used by the users of the testbed. Users can access the services of the outer layer via the public IP interfaces of the Portal Servers.

Overlay Node. These peers are responsible for the interconnection of the WSN testbeds of each partner. The *inner layer* contains client software of the services of the Portal Servers. User commands are directed to the corresponding service interface of the relevant Portal Server. Each overlay node is connected to one or more local data store for storing data retrieved from the Portal Servers (debug traces, access lists etc.) for future reference. The *outer layer* provides interfaces that are similar to those offered by the Portal Servers. Essentially they are proxy (or skeleton) interfaces that translate the incoming requests to one or more Portal Server.

High-level description of Web Services. It is our intension to provide Open Standards for both high-level services and low-level network interfaces. We organize these services in three groups:

- Authentication, Authorization, Accounting (AAA). Each portal server is part of an AAA system spanning across all federated testbeds. We will deploy the well-established, decentralized PKI-based authentication and authorization infrastructure Shibboleth to protect and simplify the inter-organizational access to the sensor network testbeds. Shibboleth is an open-source attribute-based access control system basing on state of the art cryptography and security protocols. By making the portal servers available and joining the federation, each testbed operator offers its testbed resources to all affiliated WISEBED partners.
- Network Control, Debugging and Configuration (MGT). Fully integrated portal server offer services for programming the nodes of the testbed (new binary image) and for debugging the state of the nodes (energy, communication, memory, debug interfaces using out of band methods). They also provide services for configuring the operation of the nodes (channel, transmission power).
- Data Acquisition, Query Processing, Network Operation (OPT). Portal servers (both fully integrated and semi integrated) provide a description of the testbed offered to the community. This is a list of the devices of the testbed and their capabilities. We will use and extend the SensorML standard for describing the available hardware. This group of services also allows the selection of low-level protocols or combination of protocols (from WISELIB) and the programming of the nodes from existing, tested, known binary images. It provides services for accessing the data retrieved from the sensors and issuing queries for data.

4 Software Aspects of WISEBED

In the following we give some further details concerning a number two of the important software aspects of WISEBED, specifically its integration with the Shawn network simulator and the federation of discrete sensor testbeds.

4.1 Integration with the Shawn Network Simulator

Shawn [25,26,27] is a simulation framework for WSNs with unique features for the development of algorithms, protocols and applications, that will play a major part in WISEBED. It does not compete with traditional simulators in the area of network stack simulation; instead, it focuses on an abstract, repeatable and expressive approach to WSN simulation. By replacing low-level effects with abstract and exchangeable models, the simulation can be used for huge networks in reasonable time while keeping the focus on the actual research problem. In the case of a MAC layer, for example, Shawn models the effects of a MAC layer for the application (e.g., packet loss, corruption and delay) instead of performing simulations including radio propagation properties such as attenuation, collision, fading and multi-path propagation. As a result, simulations are more predictable and there is a performance gain since such a model can be implemented very efficiently. Shawn requires orders of magnitudes less resources in terms of memory and CPU-time compared to traditional approaches and scales literally to *millions* of nodes

Shawn allows to reduce the process of porting simulated code to actual sensor nodes by using just a mere recompile. There is already implemented support for the Pacemate [28] and the iSense [29] sensor nodes. This allows using the *same code for the sensor nodes and the simulation tool* thus relieving developers from reimplementing the software for different types of hardware. A direct consequence of this architecture is that Shawn can be used as a virtual testbed, thus simplifying the development of our WISEBED testbed software infrastructure. For developers of testbed applications this simplifies the development before software is actually deployed on real hardware since standard debugging tools can be used.

4.2 Federation of Testbeds and Related APIs

In this section we will discuss further the testbeds' federation concept, presented in Section 3. As mentioned previously, WISEBED will provide abstractions so as to be able to use resources from different testbeds in a transparent manner. WISEBED aims to hide the inherent heterogeneity of the participating sensor networks to the end-user, so that even nodes with different types of hardware situated in discrete geographic areas will be able to communicate with each other.

One way of dealing with the variety of possible scenarios in this case is the use of virtual links between the discrete testbeds. Such links will essentially create "tunnels" between testbeds and sensor nodes. An ambitious goal of the project is to provide to end-users the ability not only to use nodes from different sensor networks, but also to be able to define, to a certain extent, links between nodes belonging to these discrete networks and essentially be able to define network topologies. The introduction of virtual links naturally inserts additional complexity in the operation of the system, as well generates additional issues regarding the realism achieved by the experiments conducted within the scope of the project, and will be further researched through the course of WISEBED.

WISEBED will investigate, among other, the efficiency of using such virtual links in order to implement a distributed testbed over a variety of different kinds of infrastructure, since there is a varying degree in the use of direct wired connections to the testbed nodes in the currently deployed sensor networks. This is due to the fact that these networks on the one hand have to a certain degree different target applications, and on the other hand may be deployed indoors or outdoors. Thus, the testbeds' backbone may range from completely wired, i.e., all nodes are directly connected to testbed gateways, to completely wireless, i.e., all nodes are wirelessly linked between each other and to hybrid approaches. Figure 2 depicts ways of using virtual links to interconnect sensor testbeds in different infrastructure architecture scenarios.

We will now briefly discuss the entities related with the Data Acquisition and Network Operation(OPT) API of WISEBED as a whole and each portal server in particular (see section 3). In short, this API allows programmers and applications to interface with the testbed in order to have access to data describing the setup of the testbed, with regard to the resources used and the underlying networks' operation, acting as a testbed directory. The design goal for these services are:

- To take advantage of the already defined and used open standards such as SensorML, in order to extend the application scope of the project overall.
- To supply a flexible interface that will accommodate the varying needs of several different types of users(protocol designers, application programmers) who choose to use the WISEBED infrastructure.
- Enable the interaction with groups and research projects such as OGC, SANY FP6 project, CONET.

First of all, we need to define a method to describe and uniquely identify all the different entities of the testbed - we use the following entities as a basis:

- *Sensor nodes* - We define *sensor nodes* as the unique nodes comprising each of the partner testbeds in WISEBED. Each sensor node belongs to only one *sensor network*. So, each sensor node belonging to WISEBED is described using the unique ID of the partner site, a sensor network ID which is unique in the partner site namespace, and the sensor node ID which is unique in the specific sensor network scope it belongs to.
- *Node capabilities* - With the term *node capability* we refer to the sensing and acting possibilities offered by each specific node. Node capabilities include sensor types, supported interfaces and other hardware information.
- *Edge attributes* - Edge attributes describe characteristics such as the quality of the link between two sensor nodes of the testbed.
- *Node attributes* - Sensor nodes of each WISEBED sensor network are represented as nodes of the aforementioned graph. We describe such nodes with attributes as whether is a gateway/base station.
- *Testbed portal servers* - Each partner in WISEBED maintains a portal server, offering a defined set of services. For the description of such services refer to the respective deliverables.

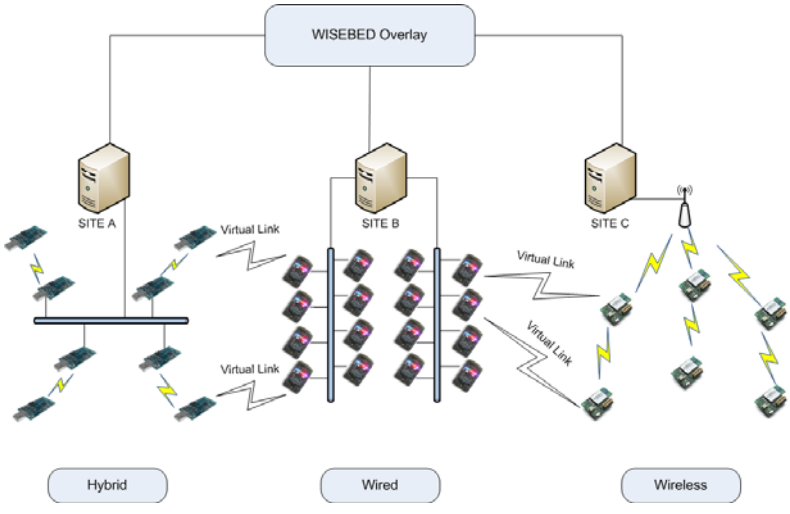

Fig. 2. Different types of testbed infrastructure (wired, wireless, hybrid) and virtual links between them

– *Points of interest* - Points of interest in WISEBED refer to specific areas, covered by the sensor networks belonging to WISEBED.

To give some examples of the API that will be used in WISEBED, Portal Servers, among others, will provide the following interfaces:

string getRecords(): Returns a GraphML document containing the complete list of nodes and the network topology[1]. GraphML is an XML dialect used to "draw" a graph, by defining a set of nodes and the edges connecting these nodes, together with sets of attributes describing the nodes and the edges of the graph. Each such graph represents a discrete sensor network, that is controlled through the use of one (or maybe more) gateway nodes.

string getCapabilities(string urn): Returns all the urn key values, for the specified sensor node, or sensor network.

string describeCapability(string urn): For the specified capability, it returns a full description in SensorML. SensorML is an XML dialect used to describe "Sensor Webs", i.e., entities (sensor nodes or networks of sensor nodes) connected to the Internet. It provides a very descriptive schema of sensors, nodes, and a large number of related attributes.

string getKml(): Returns a KML document containing a description of the networks contained in a Portal Server or the whole WISEBED infrastructure. KML is the language used by Google Earth in order to describe all data related

[1] When referring to "complete list" we imply the "authenticated list of nodes".

to that specific application, including geographic information, visual representation, time-related state, etc. It can be seen as an alternative description of GraphML and SensorML data provided by previously mentioned functions. It is a widely used data format, which is also beginning to be used in other GIS-related applications.

string getRecordsInArea(double x1, double y1, double z1, double x2, double y2, double z2): Returns a GraphML document containing the list of nodes and the respective network topology, that are within a specified area. This area is essentially defined as a cube, by providing longitude, latitude and altitude of two discrete points on Earth's surface.

5 Hardware Aspects of WISEBED – Current Deployment

Existing sensor network testbeds, in terms of hardware, are mainly characterized by small scale, homogeneity and tight integration between hardware and software, i.e., the selection of a certain hardware platform implies the adoption of a specific complementary software platform and vice versa. The so far lacking adoption of software and hardware standards by the sensor network research community, poses further restraints in the interoperability of the existing testbeds. Such limitations have of course a big impact on the applications that can be eventually implemented and tested by researchers and developers.

WISEBED aims to lift such restrictions by:

- federating testbeds of considerable size (in the order of hundreds of nodes) into "virtual" unified testbeds of large scale (in the order of thousands). The initial planning is for approximately 2000 nodes at the end of the project, with the number currently (March 2009) being approximately 550.
- offering heterogeneity, by adopting a number of hardware platforms. Heterogeneity is expressed in the form of using hardware with different computational resources, sensing resources, or the associated software resources.
- using various network topologies, mirroring the various application needs. Such topologies can be flat, hierarchical, having wired or wireless backbone, etc.
- placing the testbed nodes in various "realistic" settings, in order to simulate the conditions for certain applications such as building monitoring or natural disasters (e.g., river floods), indoor/outdoor, etc.
- using mobility to a certain degree, in order to encompass the development of mobility-related applications. A number of mobile robots, such as Roomba, will be used to enable such applications.
- using standardized radio interfaces. The multiplicity of different hardware platforms prevents a collaboration of devices from different vendors. The IEEE 802.15.4 standard is a step towards homogeneous and interoperable WSN hardware platforms. A number of the project partners use compatible 802.15.4 radio interfaces.

The processors used in the testbed's nodes are ranging from quite powerful, such as Intel PXA2xx series processors used in iMotes and GumStix, to much less powerful, e.g. MSP430F1612 used in MSB430. The minimum flash memory is 48KB (Tmote Sky nodes) and the minimum RAM is 2KB (EBS nodes), while the maximum flash and RAM resources is 32MB each (iMote). The wireless interfaces include IEEE 802.11b/g, IEEE 802.15.4 operating at 2.4GHz and other 900MHz radios. There is a variety of other interfaces present, such as Bluetooth, USB, JTAG, etc. A complete range of sensors is available ranging from the most commonly used ones, such as temperature, light, humidity, to other less common sensors like magnetometers and accelerometers. A variety of operating systems enable the users to experiment on the platforms they prefer. An analytic description of the current state of the overall testbed is included in [30]. We will now describe in more detail two specific WISEBED sites.

5.1 Lübeck Testbed Description

Univerzität zü Lübeck (UZL) currently operates two testbeds. Because of their different radio interfaces, the two different testbed nodes can only communicate via the gateway nodes. A gateway node is a normal node with an additional interface to communicate with a PC. The testbed PCs are connected among each other via a TCP/IP network to interact with the other testbed and over the Internet with the users participating in WISEBED.

The first testbed uses the *Pacemate* nodes (developed as part of the *Marathon-Net* project, see [31]). These nodes are wearable and are developed to realize services for athletes during a marathon. The testbed consist of approx. 50 sensor nodes (extensible to 500) and mobile gateways. These nodes are equipped with Philips LPC2136 processors and a Xemnics RF module running at 868 MHz. The Pacemates offer an ergonomic waterproof housing, are very light-weight and are easily attached to the back of the hand. They have a display and offer three buttons. Additionally, they are equipped with the following interfaces:

- A serial extension interface for additional sensors.
- A short range wireless heart rate receiver.
- A long range wireless interface.

The second testbed consists of up to 50 iSense nodes by coalesenses GmbH [29]. The gateway nodes as well as all other nodes have a Gateway Module with a permanent USB connection to a PC. This enables to power all nodes through the USB and guarantee a twenty-four-seven operating mode. The iSense Software is used to implement applications for the Pacemate as well as the iSense nodes. At the moment there is an iSense implementation for the iSense and Pacemate hardware and the Shawn simulation framework. Out of this reason it is possible to write an application with iSense, test it with the Shawn simulation framework and compile the same source code just using a different cross-compiler for the iSense or Pacemate architecture.

In addition to the iSense platform, Coalesenses GmbH provides a software tool called *iShell* to communicate with the network and receive debug information

Table 1. Overview of the pacemate and iSense Hardware

	pacemate	iSense
RAM	64kB	96kB
Serial Flash	256kB	128kB
Current draw operation	47mA	39mA
Current draw sleep mode	60μ A	10μ A
Frequency	868 MHz	2,4GHz
Bandwith	115kbit/s	250 kbit/s
Transmission power	15dB	3dB
Transmission range	100m	600m

from the nodes. iShell is running on a PC, which is connected to one or more gateway nodes via a USB connection. Through the PC and iShell, users can get data output (e.g., sensor readings), network status information and debug messages. Furthermore, the user can directly program the connected gateway nodes and indirectly program the other nodes in the network via over-the-air programming. The testbed PCs are connected amongst each other using cables or WiFi connections via the internet.

UZL is currently planning to enhance its testbed with mobility support. An autonomous mobile sensor network, i.e., a network where the mobility of nodes does not rely on humans, will be introduced. UZL will purchase a small number of mobile robots, which will be able to carry iSense sensor nodes and are controlled by these sensor nodes. Apart from forming their own testbed, these robots can also be used as mobile gateways between fixed sensor nodes, in order to extend their reach and the size of the overall network.

5.2 RACTI Testbed Description

The testbed in Research Academic Computer Technology Institute (RACTI) in Patras currently spans over two locations at the University of Patras' campus. These two locations include the main premises of RACTI and another building used to house several offices of RACTI's Research Unit 1. Currently, these sensor networks are mainly used to monitor condition inside these two buildings, including parameters such as temperature, light, humidity, acceleration, levels of magnetic fields, barometric pressure. In the near future we will also add sensing features such as movement/presence and vehicle detection.

The hardware architecture used for the purposes of our testbed has three hierarchical levels:

- the first level includes the nodes at the sensor network level,
- the second level includes the (stationary or mobile) gateways used to interface the sensor network to the rest of the world,
- the third level includes the servers used to store information and administer the testbed.

The deployment of the devices follows the structure of RACTI's building; each floor of the building is divided in two or three sectors, with each sector separated

with the others communication-wise, due to thick walls and metallic doors. A gateway device is used to interface the devices located in each part of the building, with the selection of the gateway based on the type of sensor nodes used. We chose to use wall plug mounts to power almost all of the sensor nodes inside the testbed, due to the difficulties arising from changing batteries in a large testbed and also to its demand for continuous availability. Currently, the testbed spans across 4 floors, covering almost one third of RACTI's main building.

In general, currently, we use devices provided by Crossbow and Sun on the sensor network level. At a glance, RACTI's testbed consists of the following devices:

- 20 Crossbow Mica2 devices, along with a number of additional sensor boards.
- 20 Crossbow TelosB devices, with embedded temperature, light, and humidity sensors.
- 45 Sun SPOT devices, with embedded light, temperature and acceleration sensors.
- 60 iSense sensor nodes, with a variety of sensor boards (50 environmental sensor boards with temperature and light sensors, 9 security sensor boards carrying a PIR camera and a 3-axial accelerometer and 1 vehicle detection sensor board carrying a magnetometer)
- 2 Crossbow Stargate Netbridge devices, used as stationary gateways.
- 2 Crossbow MIB600 network programmers.
- 5 Alix devices, used as stationary gateways.
- 3 netbook-class laptop computers used as mobile gateways.

The operating system used in the testbed, for the Mica2 and TelosB devices is TinyOS. We have recently started using Octopus (and customized it as well) for the tasking of these two types of devices inside our testbed, so the relevant code is written in NesC using TinyOS 2.x libraries. A few nodes have the Crossbow XMesh firmware installed on them. Sun SPOTs run a customized Java virtual machine, called SquawkVM, that is fully J2ME-capable and also serves as the operating system as well, so all code regarding the Sun SPOT devices is written in Java.

As for the software running on the testbed gateways, we are using Xubuntu on the Alix and the Netbook gateways; a customized Debian distribution is used on the Stargate Netbridge gateways, provided by Crossbow. This customized distribution also has the MoteWorks software (by Crossbow) installed, that is used to collect readings from the nodes using the XMesh firmware.

For the management of RACTI's testbed we mainly use WebDust. WebDust is a software platform for monitoring and controlling a multitude of disparate wireless sensor networks, using a peer-to-peer infrastructure for the communication between the different networks comprising the system, in order to achieve great scalability. The system's overall goals, apart from scalability, are to greatly simplify sensor network deployment, maintenance, and application development by offering a set of implemented services to the user and an extensible architecture to the developer, and also to offer heterogeneity by supporting a number of

Fig. 3. Testbed at RACTI through Google Earth using Webdust

hardware platforms. We currently support Mica2, MicaZ, TelosB, Sun SPOT, and iSense in the near future.

It offers a user interface related to the concepts described above, by using software like Google Earth and Google Maps. Furthermore, we are working on integrating control functionality extensions to our system, thus making actor networks a part of the system as well. In Figure 3 the layout of the deployment of RACTI's testbed through Google Earth can be seen.

6 Use-Case Scenarios – Research Challenges

6.1 Scenarios

We now give two use-case scenarios for WISEBED users, in order to give insight regarding the potential uses of this testbed.

Scenario 1. Suppose that Mr. Doe wants to test whether his new promising routing protocol for sensor networks in mesh topologies has actually a good performance. First, he develops the code for the Shawn network simulator with the aid of WISELIB library for other functionality the algorithm needs, and tests it using simulation experiments. After the initial implementation, Mr. Doe turns to WISEBED and using the user interface reserves a total of 150 nodes for his experiments. Simulated nodes in Shawn can interact with real nodes since they run the same code. Using Shawn the experiments were run with a total number of 20000 nodes; now the total number of nodes has reached 20150. The difference is that these additional 150 nodes add a new realistic dimension to the whole

experiment, by offering the ability to check whether the algorithm works right in real nodes and also scales well to thousands of nodes.

Scenario 2. Suppose that Mrs. Smith, working on an office and building automation application, wishes to test whether her software is functioning properly in practice. She would like input on events like turning on/off lights, motion sensors detecting movement in specific offices, etc. She first connects to the WISEBED infrastructure using the authorization/authentication services of the project, and reserves some testbed resources (e.g., 5 office rooms and the sensors associated with these offices). She then uses the provided services in order to retrieve (live) data about the occurrence of events in the network and its operation. We make the assumption that the testbed nodes are running some "default" software that enables such actions.

6.2 Research Challenges

We outline here some characteristic examples of the research challenges related with the project.

Federation of testbeds - transparency. As described previously, the project revolves around a federation of testbeds that will provide a unified environment. This approach poses many challenges due to the heterogeneity of software and hardware in all of the different parts of the testbed. The notion of reserving resources from parts of disparate project sites (e.g., 50 nodes from site A and another 40 from site B), even in the case of using the exactly same type of nodes and software, injects additional complexity to the project.

Interconnection with the testbed. Existing testbeds do not provide much in the ways of interfacing to the rest of the world, i.e., they usually provide a GUI to reserve some resources, upload your binary code and view results after the execution of the experiment; some environments offer the capability of monitoring the actual situation inside the network (network topology, radio activity, etc.), but all of these is often not provided in a systematic way. WISEBED aims to provide all of these capabilities in such a way, by using web services definitions and standards like SensorML, thus simplifying the whole process of interfacing with the system from an application's or developer's view.

Software and hardware platform independence - Transparency. From a practical point of view, conducting research that combines both theory and practice must deal with considerable difficulties. To name a few, wireless sensor network application developers should acquire skills regarding embedded software engineering, dealing with low-level hardware devices and understanding the peculiarities of the wireless channel (hidden terminal problem, power versus distance model) etc. Still, even if all these skills are acquired, the cost of setting up and maintaining an experimental facility of significant size can be very high. Furthermore, deploying the experimental network into a realistic environment requires iteratively reprogramming dozens of nodes, positioning them

throughout an area large enough to produce an interesting radio topology, and instrumenting them to extract debugging and performance data. WISEBED aims to provide a library of sensor network-related functionality in order to simplify the whole process of developing code for these networks and abstracting many of the details from the users and the applications.

Efficiency. The sheer volume of the nodes expected to be included in the testbed, along with many of the factors mentioned previously, naturally poses efficiency challenges. From the one side, we have the software running on the sensor network level, both as the main application and as a "backbone" service. On the first case efficiency is a necessity, on the second case the software must interfere with the rest of the system as little as possible or find ways to "hide" such activity from the user. In the other side we have the software interfacing the system with the outside world, that has to provide efficient ways to represent the information related to the operation of the testbed (e.g., a system-wide directory) or graphical end-user interfaces (current related implementations leave a lot to be desired).

7 Conclusions – Future Work

As of today, mainly isolated sensor network testbeds exist across Europe and the rest of the world. Their homogeneity, small scale and narrow application scope limit their use, up to a large degree, as a means to answer most of the research challenges related to wireless sensor networks. We have presented in this paper an overview of the WISEBED project, that tries to answer these challenges by the establishment of a large-scale sensor network by a number of federated testbed sites and the provision of a software platform to the public, in order to easily utilize all these resources. WISEBED is currently being developed, already having a number of testbed sites established. Future work on the project involves the expansion of the existing testbed sites, both in scale and heterogeneity, and the implementation of the software ecosystem surrounding the project. There is a large body of work related to the interconnection of the testbed sites, the development of an algorithmic library providing implemented solutions compatible with the existing testbed to application developers, and also the tighter integration between the testbed and virtual (simulated) sensor networks.

Acknowledgments

This work has been supported by the ICT Programme of the European Union under contract number ICT-2008-224460 (WISEBED). We would also like to thank S. Fekete and A. Kröller for providing one of the insightful use-case scenarios mentioned in section 6.

References

1. WISEBED project website, http://www.wisebed.eu/
2. Dutta, P., et al.: Trio: enabling sustainable and scalable outdoor wireless sensor network deployments. In: 5th International conference on Information processing in sensor networks (IPSN), pp. 407–415. ACM Press, New York (2006)
3. Werner-Allen, G., Swieskowski, P., Welsh, M.: Motelab: A wireless sensor network testbed. In: Fourth International Conference on Information Processing in Sensor Networks (IPSN). IEEE, Piscataway (2005); special Track on Platform Tools and Design Methods for Network Embedded Sensors (SPOTS)
4. Handziski, V., Kopke, A., Willig, A., Wolisz, A.: Twist: A scalable and reconfigurable wireless sensor network testbed for indoor deployments. Tech. Rep. TKN-05-008 (November 2005)
5. Tutornet: A tiered wireless sensor network testbed, http://enl.usc.edu/projects/tutornet/
6. Chun, B.N., et al.: Mirage: A microeconomic resource allocation system for sensornet testbeds. In: 2nd IEEE Workshop on Embedded Networked Sensors (2005)
7. Ertin, E., et al.: Kansei: A testbed for sensing at scale. In: 5th International conference on Information processing in sensor networks, IPSN (2006)
8. Johnson, D., et al.: Mobile emulab: A robotic wireless and sensor network. In: 25th IEEE Conference on Computer Communications, INFOCOM (2006)
9. Murty, R., Mainland, G., Rose, I., Chowdhury, A., Gosain, A., Bers, J., Welsh, M.: Citysense: An urban-scale wireless sensor network and testbed. In: 2008 IEEE Conference on Technologies for Homeland Security, May 2008, pp. 583–588 (2008)
10. SENSEI project website, http://ict-sensei.org/
11. Henricksen, K., Robinson, R.: A survey of middleware for sensor networks: State-of-the-art and future directions. In: Proc. of MidSens 2006 (2006)
12. Hadim, S., Mohamed, N.: Middleware challenges and approaches for wireless sensor networks. IEEE Distributed Systems Online 7(3) (2006)
13. Romer, K., Kasten, O., Mattern, F.: Middleware challenges for wireless sensor networks. ACM Mobile Computing and Communications Review 6, 59–61 (2002)
14. Kuorilehto, M., Hannikainen, M., Hamalainen, T.: A survey of Application Distribution in Wireless Sensor Networks. EURASIP Journal on Wireless Communication and Networking 5, 774–788 (2005)
15. R. Project, Survey of middleware for networked embedded systems, deliverable 5.1 (2005)
16. Levis, P., Lee, N., Welsh, M., Culler, D.: Tossim: accurate and scalable simulation of entire tinyos applications. In: SenSys 2003: Proceedings of the 1st international conference on Embedded networked sensor systems, pp. 126–137. ACM Press, New York (2003)
17. Tolle, G., Culler, D.: Design of an application-cooperative management system for wireless sensor networks. In: European Cinference on Wireless Sensor Networks, EWSN 2005 (2005)
18. Ramanathan, R., et al.: Sympathy for the sensor network debugger. In: 3rd International Conference on Embedded Networked Sensor Systems (SenSys), pp. 255–267 (2005)
19. Rost, S., Balakrishnan, H.: Memento: A health monitoring system for wireless sensor networks. In: SECON 2006 (2006)

20. Whitehouse, K., et al.: Marionette: using rpc for interactive development and debugging of wireless embedded networks. In: 5th International conference on Information processing in sensor networks (IPSN), pp. 416–423. ACM Press, New York (2006)
21. Ringwald, M., Römer, K., Vitaletti, A.: Passive inspection of sensor networks. In: Aspnes, J., Scheideler, C., Arora, A., Madden, S. (eds.) DCOSS 2007. LNCS, vol. 4549, pp. 205–222. Springer, Heidelberg (2007)
22. Crepaldi, R., Friso, S., Harris, A., Mastrogiovanni, M., Petrioli, C., Rossi, M., Zanella, A., Zorzi, M.: The design, deployment, and analysis of signetlab: A sensor network testbed and interactive management tool, May 2007, pp. 1–10 (2007)
23. Chen, B.-R., Peterson, G., Mainland, G., Welsh, M.: Livenet: Using passive monitoring to reconstruct sensor network dynamics. In: Nikoletseas, S.E., Chlebus, B.S., Johnson, D.B., Krishnamachari, B. (eds.) DCOSS 2008. LNCS, vol. 5067, pp. 79–98. Springer, Heidelberg (2008)
24. Österlind, F., Dunkels, A., Voigt, T., Tsiftes, N., Eriksson, J., Finne, N.: Sensornet checkpointing: Enabling repeatability in testbeds and realism in simulations. In: Roedig, U., Sreenan, C.J. (eds.) EWSN 2009. LNCS, vol. 5432, pp. 343–357. Springer, Heidelberg (2009), http://dblp.uni-trier.de/db/conf/ewsn/ewsn2009.html#OsterlindDVTEF09
25. Shawn, http://shawn.sf.net
26. Kröller, A., Pfisterer, D., Buschmann, C., Fekete, S.P., Fischer, S.: Shawn: A new approach to simulating wireless sensor networks. In: Design, Analysis, and Simulation of Distributed Systems (DASD 2005), pp. 117–124 (2005)
27. Fekete, S.P., Kröller, A., Fischer, S., Pfisterer, D.: Shawn: The fast, highly customizable sensor network simulator. In: Proceedings of the Fourth International Conference on Networked Sensing Systems (INSS 2007) (June 2007)
28. Lipphardt, M., Hellbrück, H., Pfisterer, D., Ransom, S., Fischer, S.: Practical experiences on mobile inter-body-area-networking. In: Proceedings of the Second International Conference on Body Area Networks, BodyNets 2007 (2007), http://www.bodynets.org/
29. Coalesenses iSense - A modular hardware and software platform for wireless sensor networks, http://www.coalesenses.com/isense
30. Design of the Hardware Infrastructure, Architecture of the Software Infrastructure, and Design of Library of Algorithms, http://www.wisebed.eu/index.php/deliverables
31. Pfisterer, D., Lipphardt, M., Buschmann, C., Hellbrueck, H., Fischer, S., Sauselin, J.H.: MarathonNet: Adding value to large scale sport events - a connectivity analysis. In: Press, A. (ed.) Proceedings of the International Conference on Integrated Internet Ad hoc and Sensor Networks (InterSense 2006), May 2006, p. 12 (2006), http://doi.acm.org/10.1145/1142680.1142696