# Using and Operating Wireless Sensor Network Testbeds with WISEBED

Horst Hellbrück*, Max Pagel†, Alexander Kröller†,
Daniel Bimschas‡, Dennis Pfisterer‡, and Stefan Fischer‡
* Lübeck University of Applied Sciences, Germany, Email: {hellbrueck}@fh-luebeck.de
†Braunschweig University of Technology, Germany, Email: {pagel, kroeller}@tubs.de
‡Institute of Telematics, University of Lübeck, Germany, Email: {bimschas, pfisterer, fischer}@itm.uni-luebeck.de

***Abstract*—Current surveys and forecast predict that the number of wireless devices is going to increase tremendously. These wireless devices can be computers of all kinds, notebooks, netbooks, Smartphones and sensor nodes that evolve into real-world scenarios forming a "Real-World-Internet" in the future. In our work we focus on the Future Internet with small battery driven devices forming the "Internet of Things". In recent networking research, testbeds gain more and more attention, especially in the context of *Future Internet* and wireless sensor networks (WSNs). This development stems from the fact that simulations and even emulations are not considered sufficient for the deployment of new technologies as they often lack realism. Experimental research on testbeds is a promising alternative that can help to close the gap.**

**The deployment of testbeds is challenging and user and operator requirements need to be considered carefully. Therefore, the goal is to design an architecture that allows operators of WSN testbeds to offer numerous users access to their testbeds in a standardized flexible way that matches these requirements. In this paper we first identify some of the requirements, then introduce the architecture and general concepts of our WISEBED approach and show how this architecture meets the requirements of both groups. We give an overview of existing WISEBED-compatible WSN testbeds that can be used for experimentation today. Main focus in this paper compared to previous work is to address the perspective of both users and operators on how to experiment or respectively operate a WSN testbed based on WISEBED technology.**

## I. Introduction

In current research of the *Future Internet* small battery driven devices forming the "Internet of Things" are of special focus as the number of wireless devices is going to increase tremendously. Two years ago a survey of the wireless world research forum predicted that in the year 2017 there will be seven trillion wireless devices for seven billion humans which is equivalent to 1000 devices per human being on average [1].

The development in this field is based on a clear trend, where computers of all kinds, notebooks, netbooks, smartphones and sensor nodes evolve into real-world scenarios forming a "Real-World-Internet" in the future. Modeling of real world applications is challenging and simulations are only one step towards evaluation of practical usage of these systems. In order to investigate how protocols and algorithms perform in real-world, experimental research is one of the most promising means. In research institutions, testlabs or testbeds are deployed exclusively for of all kinds of research experiments. Setup in realistic environment is preferable in large scale with many devices forming large networks.

In recent years several global initiatives for experimental network research have been started. One of the largest and well known activities is PlanetLab [2], that is in operation since 2002 and has developed to a world-wide available distributed lab. PlanetLab is the basis of various initiatives like GENI [3], and its European counterpart FIRE (Future Internet Research and Experimentation, [4]). At national level G-Lab [5] provides a German experimental platform for the German Future Internet activities.

While all these large projects include all aspects of the Future Internet some special projects focus on the Internet of Things and the Real-World-Internet. Especially in the Internet of Things with wireless sensor networks (WSNs) there is an upcoming need for large heterogeneous testbeds available 24hs a day that can be used automatically without supervision.

We have identified the following requirements from the user. *Transparent access* to the testbed is important for the user. Transparency in this context means that sensor nodes in a remote testbed can be handled in the same way as local sensor nodes. In order to use the testbed a *robust, fast and simple reservation* is essential, including support for isolated (non interfering) experiments with other users.

*Heterogenity* and *scale* of the testbeds are important to address challenges that occur in real world deployment scenarios. Many algorithms and applications do not scale or behave unstable in large scale. Therefore, the network size should be as large as possible. As deployment cost is still a dominating factor the concept of testbed *federation* to enable experiments at scale can help to reduce overall facility cost. To enable federation *standardization* of APIs is a key concept if not a prerequisite. In the same way heterogeneity can affect the performance of an application. Heterogeneity might include memory (Program and Data), wireless transceivers characteristics, or microcontroller. We can introduce heterogeneity by using different classes of devices from an embedded 8-Bit microcontroller platform to a fully equipped PC. A full support of a testbed for heterogeneity even allows performing interoperability tests.

*Reproducibility* is a property that is very difficult to achieve in a testbed. However, the support to repeat experiments is needed to reach statistical soundness of the experiment. An

978-1-4577-0899-2/11/$26.00 ©2011 IEEE 

additional supporting feature is detailed *logging* capabilities to enable post-failure analysis and debugging support. *User interaction* is very helpful to stimulate events or errors in the testbed during experiment runtime in order to investigate reactions of the system under test.

In the future support for *mobility* in testbeds will become more important. Therefore, testbeds deployed today must be prepared for a further extension by mobile devices in a next step.

The requirements from the operator additionally include *robustness* of the testbed, which is of course desirable for the user too. When nodes or the complete wireless sensor networks crash due to software errors, the testbed must recover from this failure. Therefore, support for remote or even better automatic reset of the experimental facility to a safe and basic state is mandatory.

Additionally, the operator requires access control system to his testbed that monitors the activities and allows for future accounting and even payment per use.

Another success factor of a testbed approach is an *easy installation* and a *fully automated reservation system*. A fully integrated testbed management and maintenance system including health monitoring capabilities is definitely helpful and can be addressed as a long term goal when the testbeds grow in size. The testbed software must be ready for *future extensions*. Therefore, open software is preferable over a closed or proprietary system.

In this article we will introduce into the domain of wireless sensor networks testbeds and discuss solutions from the users or experimenters perspective as well as the operators' perspective. The focus of previous work like [6] was on the technical details of WISEBED whereas in this article we will discuss the WISEBED approach from users and operators perspective. We will demonstrate how a common API based on Internet technologies like URN and Web services is beneficial for both sides (usage and operation). One important aspect of the success of a testbed is its capability to allow seamless transition from simulation to testbed or even allow for mixed operation. The transparent access and convenient planning of topologies allows forming a worldwide large testbed from small single testbeds at different places. These aspects of transparency include additionally the uniform way to receive debugging and measurement data from heterogeneous testbeds.

All of these characteristics are enablers of testbeds but without proper management support for authentication and authorization to allow secure usage, testbed operators could not offer services like non-interfering test runs to experimenters. Consequently there is a need for planning and management tools to support testbed operators. We will introduce our integral approach and provide details of the user scenario for the testbed.

The remainder of this paper is structured as follows. Section II will discuss related work including overview of existing publicly available WSN testbeds. The basic architecture of our approach WISEBED is introduced in Section III followed by a discussion of existing WISEBED testbeds. The main parts Section IV and Section V will discuss the users and operators perspective of WSN testbeds. Section VI concludes this paper.

| | # of nodes | Heterogeneity | Federation | SW-Reuse | Reference(s) |
|---|---|---|---|---|---|
| **DES-Testbed** | 95 | | | | [7] |
| **FRONTS** | 21 | | | | [8] |
| **Kansei** | 260 | ✓ | | | [9] |
| **KanseiGeni** | 576 | ✓ | ✓ | | [10] |
| **MIRAGE/Intel** | 100 | | | | [11] |
| **MoteLab** | 190 | | | ✓ | [12] |
| **NetEye** | 130 | | | | [13] |
| **Senslab** | 1024 | ✓ | ✓ | | [14] |
| **TutorNet** | 104 | | | | [15] |
| **TWIST** | 204 | ✓ | | ✓ | [16] |
| **VineLab** | 48 | | | | [17] |
| **WISEBED** | 750 | ✓ | ✓ | ✓ | [6], [18] |
| **w-iLab.t Testbed** | 200 | | | | [19] |

TABLE I
OVERVIEW OF PUBLICLY AVAILABLE WSN TESTBEDS

## II. RELATED WORK

Recently, a number of testbeds have been described in literature while only some of them are actually active and publicly accessible. This section provides an overview of existing, publicly available testbeds and classifies their features with regard to the testbeds' size and the features of the software used for operating, managing, and using the testbed. Table I summarizes the most well-known testbeds, lists the number of available sensor nodes in a testbed and indicates whether the testbed comprises heterogeneous hardware, if it supports federating multiple testbeds at different sites, and whether the software used for testbed operation has been reused to run third-party testbeds.

Some testbeds (e.g., KanseiGenie, Senslab, and TWIST) support heterogeneous hardware and thus allow testing algorithms on different platforms. Three of the testbeds support the notion of federation. KanseiGenie is federated with the GENI infrastructure, Senslab federates four different sites of the same layout, and WISEBED provides a standardized federation scheme. For most of the testbeds the software that is used for operation and management has been custom-tailored for each specific testbed and has not been reused at other sites. Only MoteLab, TWIST and WISEBED provide software that was reused to operate third-party testbeds. However, except for WISEBED, no other testbed management software has produced a generic infrastructure or standard APIs with different implementations that can be used to create custom testbeds and federate them with testbeds operated by others.

## III. WISEBED ARCHITECTURE AND EXISTING TESTBEDS

WISEBED's architecture was designed with generality in mind. The top-most design goal was to create a set of standardized APIs by which a testbed is accessed from a user's perspective. By standardizing the API, users can access

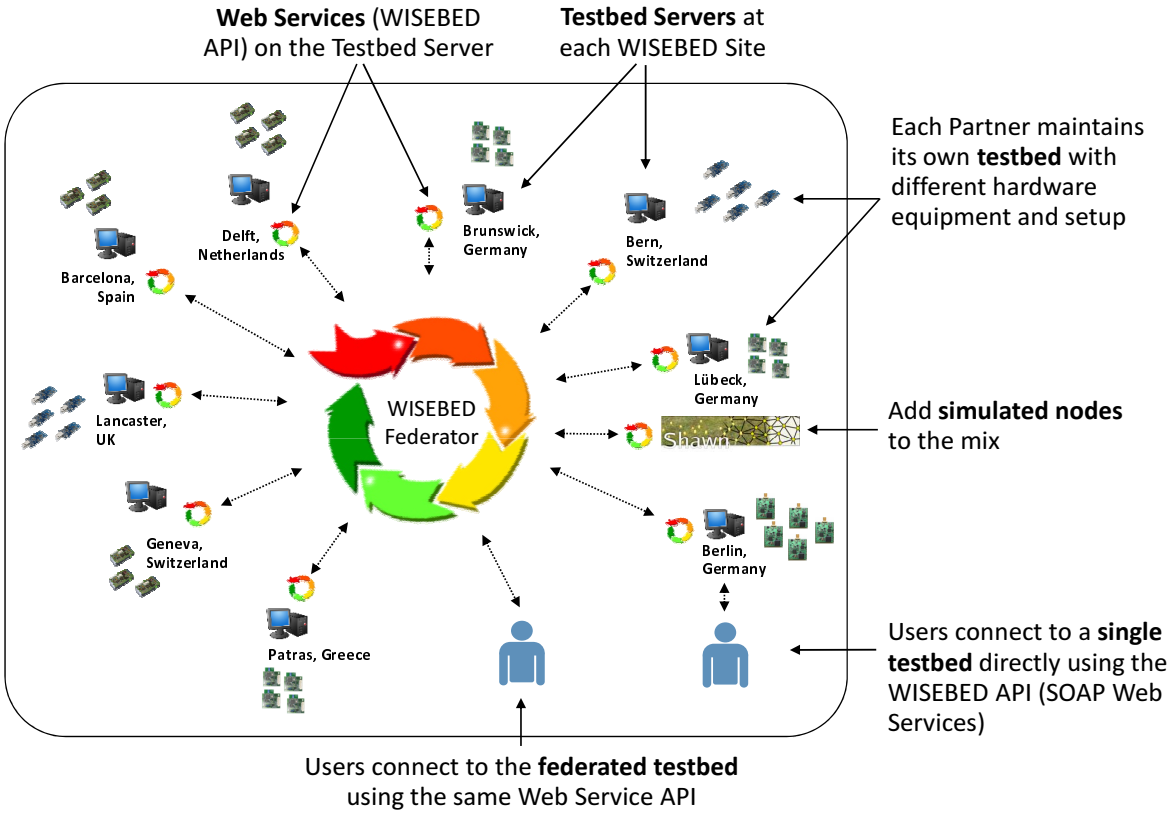


Fig. 1. Testbed Federation Architecture

compatible testbeds using the same clients, no matter if the testbed comprises only a single node connected to a laptop or a full-blown testbed with thousands of nodes. Using the same API and hence the same client software, researchers can deploy the same experiment to a number of testbeds automatically and compare the results.

Implementations of the APIs are completely up to the operators of testbeds. However, a number of backend implementations of the WISEBED APIs are available under open-source licenses and individuals or research organizations can easily download and deploy the software to run WISEBED-compatible testbeds. The result is an ever growing ecosystem of testbed client/backend software including documentation and it is possible to put descriptions of experiments online which can be used by others for repeating experiments and verifying results.

### A. Architecture

Figure 1 depicts the overall WISEBED architecture. Each testbed (e.g., the ones located at Lübeck or Patras) comprise a number of sensor nodes. These nodes are managed and controlled by a so-called *Testbed Server*. A Testbed Server exposes the functionalities of a testbed to users in the Internet by running software that provides WISEBED API-compatible Web services. Using these Web services, users can run experiments on single testbeds.

Testbeds are interconnected by a *Federator* component, which exposes a federation of testbeds using the same WISEBED APIs and thus creating a virtual large-scale unified testbed. From an API point of view, the federation appears to be a single testbed. This allows users to use all sensor nodes of all testbeds at the same time and as part of a single experiment transparently. To interconnect spatially divided testbeds, WISEBED employs the concept of *Virtual Links* [20]. Virtual Links emulate broadcast connectivity between arbitrary nodes by tunneling messages between the communication partners using the WISEBED APIs. Additional nodes can be added to the federation by integrating simulated nodes into experiments. The Shawn [21], [22] network simulator has been extended to support the WISEBED APIs and can hence be part of a federated testbed.

The Testbed Server (TS) runs the aforementioned Web service-based *WISEBED APIs* that users call to interact with the testbed, consisting of the

- *Sensor Network Authentication and Authorization* (SNAA) API, the
- *Reservation System* (RS) API, the
- *Wireless Sensor Network* (WSN) API and the
- *Controller* API.

The first two APIs provide interfaces for authentication and authorization of users as well as resource reservation. The WSN API describes the main entry point for conducting ex-

periments and allows users to interact with the nodes (e.g., reprogram and send messages). Users that conduct experiments start a Web service endpoint implementing the Controller API where the Web service listens for output generated by the nodes (e.g., using the serial interface).

Figure 2 illustrates the architecture of an individual testbed in more detail. We distinguish between two different types of testbed architectures: *wired* and *wireless*. In a *wired* testbed every sensor node is attached to a host system (called *gateway*) via a serial USB connection. In contrast, wireless testbeds have at least one sensor node that is *not* connected to any kind of wired communication backbone, i.e., communication with and reprogramming of such nodes can only be done wirelessly. In wired testbeds, the use of additional *gateway* components may be required if not all sensor nodes can directly be attached to the Testbed Server, e.g., because they are spread over multiple rooms. Typically, such a gateway is an embedded PC or a netbook-class computer.

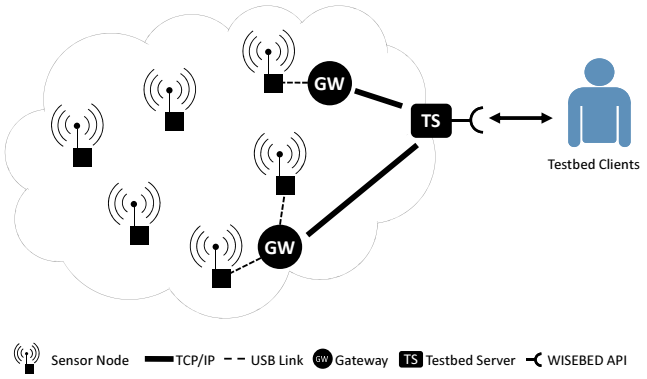


Fig. 2. Testbed Architecture

The Testbed Server and the gateways typically communicate through a wired infrastructure, such as Ethernet to provide a reliable backbone. Additional sensor nodes can be "attached" wirelessly as described above. (Re-)Programming of these nodes is based on an over-the-air programming (OTAP) mechanism which must be present on the sensor node at any time. Our implementation *Testbed Runtime* [23] of the WISEBED APIs (described in more detail in Section V) supports both types of testbed infrastructure architectures, wired and wireless.

### B. Existing WISEBED-compatible Testbeds

Currently nine partners provide testbeds comprising a total of 750+ heterogeneous wireless sensor nodes of various architectures and vendors (cf. Table II). All these testbeds are permanent installations with wired backbones. The wired backbone enables out-of-band interaction with the nodes (e.g., collect traces and debug information or send commands to the nodes). Apart from the permanently installed wired testbeds, several sites offer on demand extensions such as the possibility to introduce mobility into experimentation by using robots that piggyback wireless sensor nodes and move autonomously through the testbeds.

Apart from the WISEBED testbeds provided by the nine partners, a number of WISEBED-compatible testbeds have been deployed recently:

- Wireless solar powered outdoor testbeds are available at the University of Braunschweig and the University of Lübeck, Germany. They can be used to perform experiments under realistic, outdoor conditions.
- The University of Applied Sciences in Lübeck, Germany, extended one of the WISEBED APIs implementations to support their custom-made sensor node platform TriSOS. The TriSOS platform is based on AVR Raven evaluation board. Their testbed is now run using WISEBED software and is accessible using the WISEBED APIs and clients.
- The upcoming testbed of the city-wide deployment of the SmartSantander project [24] is based on WISEBED technology and will contribute actively in the further evolution of APIs, client and backend implementations.
- The projects WSNLAB [25] and MOVEDETECT [26], which are funded by the Federal Office for Information Security (BSI), base their deployment architecture on the WISEBED APIs and the Testbed Runtime implementation.

The above listed activities demonstrate that the WISEBED ecosystem is rapidly growing and there's an ongoing active development in order to support new requirements.

## IV. Using WISEBED-compatible Testbeds

The WISEBED infrastructure offers several valuable properties that enable testbed users to conduct well-defined experiments. One important property is the ability of the client APIs to receive live data from a running experiment. Based on this data the experimenter is able to feed information into running experiments to influence the experiment, e.g., to set different parameters or to run a protocol that interacts with the sensor nodes such as 6LoWPAN (an IPv6 adaptation layer). By using this feature testbed users can create interesting and interactive experiments, or simply try different parameter sets easily, to quickly find reasonable values based on real-world data.

Another important aspect of the WISEBED architecture is the ability to federate several testbeds into a larger testbed. These federated testbeds can be further enlarged by adding virtual, simulated nodes to the federation (e.g., by using the Shawn simulator [21], [22]). This offers a simple and transparent way to conduct experiments on large-scale testbeds, which can be important as some effects only occur in the presence of a large number of devices [27].

Figure 3 provides an overview of the actual API calls that are involved in the whole process of setting up and running an experiment on a WISEBED testbed. The following subsections will walk through this process in detail, explaining the semantics involved and discussing what features and operations are available to clients during experiment runtime.

### A. Authentication and Reservation

In order to be able to conduct experiments on one or more WISEBED testbeds, the experimenter needs an account at

TABLE II
AVAILABLE WISEBED TESTBED SITES

| Partner | Nodes | Sensors | Wireless Interface | Amount |
|---|---|---|---|---|
| Lübeck | Pacemate | Heartrate | Xemics RF (868 MHz) | 60 |
| | iSense | Temperature, light, PIR, accelerometer | IEEE 802.15.4 (2.4GHz) | 60 |
| | TelosB | Temperature, humidity, light | IEEE 802.15.4 CC2420 (2.4GHz) | 60 |
| Lancaster | Tmote/TelosB | Temperature, humidity, light | IEEE 802.15.4 CC2420 (2.4GHz) | 40 |
| Bern | TelosB | Temperature, humidity, light | IEEE 802.15.4 CC2420 (2.4GHz) | 20 |
| Patras | TelosB | Temperature, humidity, light | IEEE 802.15.4 CC2420 (2.4GHz) | 20 |
| | MicaZ | Temperature, light, accelerometer, acoustic, and magnetic fields | IEEE 802.15.4 (2.4GHz) | 20 |
| | iSense | Temperature, humidity, PIR | IEEE 802.15.4 (2.4GHz) | 60 |
| | SunSPOT | Light | IEEE 802.15.4 (2.4GHz) | 60 |
| Barcelona | SunSPOT | Temperature, light, accelerometer | IEEE 802.15.4 (2.4GHz) | 24 |
| | iSense | Temperature, humidity, PIR | IEEE 802.15.4 (2.4GHz) | 5 |
| | iSense | Solar harvesting | IEEE 802.15.4 (2.4GHz) | 5 |
| Geneva | iSense | Temperature, light, PIR, AMR, accelerometer | IEEE 802.15.4 (2.4GHz) | 28 |
| | MicaZ | Temperature, light, accelerometer, acoustic, and magnetic fields | | 2 |
| Delft | TNode | Temperature, humidity | CC1000 (868 MHz) | 24 |
| | Tmote Sky | Temperature, humidity, light | IEEE 802.15.4 CC2420 (2.4GHz) | 8 |
| | G-Node | | IEEE 802.15.4 (2.4GHz) | 108 |
| Berlin | MSB-A2 | Temperature, humidity | CC1100 (864-970MHz) | 100 |
| Brunswick | iSense | Pressure, infrared | None | 34 |
| | | | **Total** | **738** |

the SNAA system, which is available through the WISEBED homepage [18]. As the WISEBED SNAA API is based on the open source single sign-on system Shibboleth, a user account is valid for each WISEBED compatible testbed. Once an account is provided, the user is able to authenticate himself by sending a Web service request to the SNAA endpoint address of the desired testbed he wants to use or to the Federator that represents a federation of several testbeds. This can be done by writing a custom Web service client or by using the BeanShell client or the command line interface that ships with *Testbed Runtime*. If the Authorization succeeds the SNAA API returns a secret authentication key, which is needed to reserve the nodes needed for an experiment. WISEBED enables performing several experiments in parallel on the same testbed which is used to run big testbeds more efficiently. However if a researcher needs to completely eliminate all side effects from other experiments he needs to reserve the complete testbeds at every participating site. To choose sensor nodes at a testbed site a user can download an XML-description of the testbed, containing all meta information about the available sensor nodes like hardware platform, sensors, position etc. Each senor node is identified by a unique Universal Resource Name (URN) which shares a common prefix for all nodes inside a single testbed. The urn ”urn:wisebed:tubs:0x191” for example represents a node with the MAC-address 0x191 inside the testbed of the University of Braunschweig (”tubs”).

After selecting nodes for the experiment, the user uses the Reservation System to reserve these nodes for a certain time period. This is done by calling the *makeReservation* method of the Reservation API, and pass the secret authentication key received from the SNAA along with a list of nodes which are to be reserved. The Reservation System then checks with the SNAA if the user is authorized to perform the desired action and reserves the nodes if they are available for the desired time period. As return value the user receives a secret reservation key which he uses to access his experiment through the WSN API. This secret key is called the reservation key in the following.

### B. Running experiments

With a valid reservation key a user is able to interact with the WSN API. Interaction with the sensor node can be twofold. On the one hand he can reprogram and reset nodes. That enables full access to the nodes for experiments on all layers of the networking stack ranging from physical layer, MAC layer, routing or transport issues to application layer and applications on system level. On the other hand he can send and receive messages to and from the nodes. The second interaction can be used to either collect debugging information / measurement results or to send controlling instructions to the nodes to manage running experiments. For both interactions the user keeps full control over the complete experimental setup. The testbed provides remote access to individual nodes program memory as well as access to a stream oriented debug interface.

The user needs to call the session management interface which provides him with a dedicated instance of the WSN API for his reservation key. This instance is provided as a Web service endpoint URL that is the address for all communication with the experiment. To enable bidirectional communication between the experiment and the experimenter, the caller needs to provide a controller URL as a parameter to the *getInstance* call to the session management interface. There has to be a

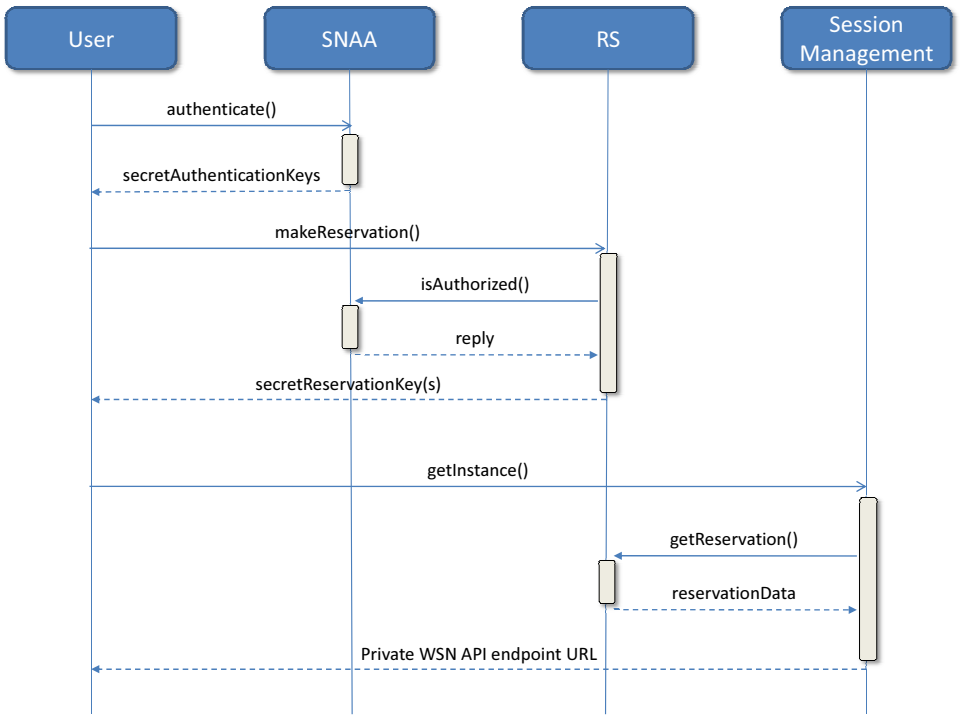


Fig. 3. Interaction of the different WISEBED APIs to obtain a private testbed instance for experimentation

valid implementation of the WISEBED controller API available at this given URL. This controller API is used for callback mechanisms. The methods of the WISEBED controller API are called by the WSN API implementation to send experiment output or asynchronous status updates on ongoing operations, like (re-)programming nodes, to the owner of the experiment. There are already several possible implementations of the controller API that ship with *Testbed Runtime* like a bean shell command-line script and a pure java Web service client. Apart from those implementations the user is free to implement a Web service that conforms to the controller API WSDL in any technology that suits his needs. These generic and well defined interfaces with sample implementations enable the usage the WISEBED testbed infrastructure in a wide range of experimentally driven research activities.

If a special topology in the sensor network is needed for an experiment, the WSN API offers control over link availability between nodes and the formation of virtual links between nodes which are not in communication range. Using these mechanisms a user is able to create any network topology he needs even if the utilized nodes are not in the same testbed. The link control is achieved on the one hand by a software component on the sensor nodes, that suppresses messages between two nodes if their physical link has been disabled and on the other hand by a module inside the gateway software, e.g. *Testbed Runtime*, which controls virtual link message flows between two separate testbeds through the backbone Network. Virtual links are advantageous when experiments are not time critical as the virtual links introduce additional delays especially when creating links across testbed sites. The main advantage is that the user keeps control of the topology without adjusting sensor node positions.

## V. Operating WISEBED-based testbeds

A valuable option is to use the WISEBED software components to deploy a private testbed that is not part of the WISEBED federation but that is API compatible. The WISEBED APIs are designed in a way that they are technology independent. This allows deploying different backend implementations ranging from small-scale deployments on a single PC to a full-blown testbed federation. In the simple case, no authentication or reservation may be required and dummy implementations for both APIs can be used. In the case of a large federation, more complex implementations are required. In the following, we discuss how users can deploy a private testbed running on a single PC with only one or more nodes attached and how one can deploy a large-scale WISEBED-compatible testbed, which is not part of the federation.

All software parts, both infrastructure and client applications are freely available under an Open Source license from the project's homepage [23] containing RS, SNAA and iWSN implementations. They are ready-to-run and only require some configuration. In the remainder of this section we discuss how users can run small-scale desktop testbeds on their workstation machines (e.g., to debug their applications) and how to operate a large-scale testbed that may or may not be or become part of the WISEBED federation. Both variants can be realized using the *Testbed Runtime* [23] implementation of the WISEBED APIs.

### A. Small-Scale Desktop Testbed

In the simplest case one can run the WISEBED software solutions on desktop systems that have one or more nodes attached to it (cf. Figure 4). In this setup, a user can run his sensor network applications with the same WISEBED-compatible client in his personal test setup before using a larger testbed. This way, the application on the node as well as a custom testbed client application and their interaction can be debugged beforehand, thereby saving valuable experimentation time on a larger testbed. Setting up a personal desktop testbed is also very helpful for users to learn about the environment and the software solutions first. In this deployment, dummy implementations of the SNAA and RS API are used to simplify the setup and interaction. In addition, an instance of the WSN API is started and attached to a local Controller that controls the experiment.

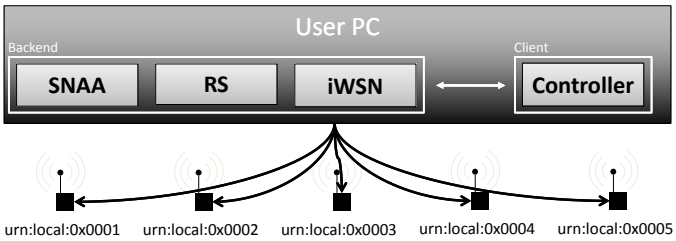


Fig. 4. Architecture of a small-scale desktop-only testbed

To run Testbed Runtime on the local machine a user simply has to customize a configuration file according to the devices attached and start the system using a shell script. In the next step he can run several client applications (for e.g., flashing, resetting, and logging) that are also part of the package, write his own client applications from scratch or use the packaged ones as templates for custom ones.

### B. Large-Scale Private Testbed

A more flexible setup that allows for resource reuse is the WISEBED APIs and software components to deploy, operate and use a private, potentially large-scale testbed. Setting up such a testbed is easy and in most cases involves only hardware setup and configuring ready-to-run software components.

For the sake of simplicity let us assume that we want to deploy a *wired* testbed as described in Section III. The biggest advantage of a wired testbed is that it needs less maintenance as the nodes are always attached to a power source and can be reprogrammed even if the application on the node crashes, overwrites its own program memory, etc. Guaranteeing stable operation in a wireless testbed is much harder as reprogramming of nodes can only be done trough (multi) hop over-the-air programming (MOTAP) algorithms that need to be present and operational on the node and eventual intermediate nodes on the multi hop path at any time. Currently, Testbed Runtime offers some support for wireless testbeds comprised of iSense hardware.

To run a testbed, the actual implementations of the different APIs must be configured. Testbed Runtime's reservation system implementation ships with several persistence layers:

- *Java Persistence API (JPA):* allows the use of virtually all kinds of databases supported by Java's object-relational mapping standard JPA.
- *Google Calendar:* uses a dedicated Google Calendar as persistence layer.
- *In-Memory:* provides an in-memory implementation of the persistence-layer which can be used, e.g., for desktop or small-scale private testbeds or unit-testing.

Also, Testbed Runtime ships with several implementations of the SNAA API so that it can easily be integrated into existing environments:

- *Java Authentication and Authorization Service:* allows pluggable and extendable components to authenticate and authorize users as defined by the well-known JAAS standard. Currently used, e.g., for LDAP servers and *htpasswd* files.
- *Shibboleth:* provides support for the well-known distributed single sign-on system Shibboleth [28].

In a standard setup the implementations of the iWSN API are deployed on several physical machines: the testbed server is connected to further *gateways* via a local area network. The sensor nodes themselves are attached to these gateways. When configured and running, Testbed Runtime employs a simple and high-performance overlay network between testbed server and the gateway machines. On every gateway an instance of Testbed Runtime's gateway module connects itself to the attached sensor nodes in order to listen, reset, reprogram, etc. the nodes. If a user invokes a method on the testbed server that involves interaction with a sensor node, this call is forwarded to the appropriate gateway(s) which then execute the actual logic. In addition, sensor node outputs are forwarded to the testbed server which delivers these to the controller APIs implementation provided by the client (cf. Section III).

For more details on how to install and configure a small- or large-scale private testbed, how to use client software and how to contribute please take a look at the Testbed Runtime Wiki [23]. As an open-source project, contributions and extensions from interested individuals and institutions are welcome. The project's source code is hosted on github [29].

## VI. Conclusion and Outlook

In this paper we have motivated the need for large wireless sensor network testbeds. We addressed important requirements from the users and operators perspective and presented how they can be fulfilled with the WISEBED approach. WISEBED provides solutions for small and large-scale experiments. The main benefit is that these experiments can be performed across testbed platforms at different sites spread all over the world even with heterogeneous sensor node hardware. All testbed sites implement the same interfaces and therefore provide a consistent API to users. Thereby, testbed users gain complete remote access to the sensor nodes including

program memory and a stream oriented debugging interface achieving the same flexibility like doing experiments in a local environment. WISEBED operators on the other hand keep control of their testbed site by means of authentication and reservation mechanisms that provide security features and allow only registered with appropriate user rights to perform experiments.

Although accessing and conducting experiments for heterogeneous wireless sensor networks in a unified manner is a large step forward there are other challenges to tackle in the future. If users want to run their application on a different hardware platforms or perform a mixed operation in a heterogeneous environment we need further support for seamless writing of programs for these different hardware platforms. A further step will be the extension of the APIs to support the setup of mobility scenarios.

## ACKNOWLEDGEMENTS

This work has been partially supported by the European Union under contract number IST-2008-224460 (WISEBED) and by the Federal Ministry of Education and Research of Germany: Förderkennzeichen 01BK0905, 01BK0906, 01BK0907, Real-World G-Lab and 17PNT017, DataCast.

## REFERENCES

[1] J. F. Gantz, C. Chute, A. Manfrediz, S. Minton, D. Reinsel, W. Schlichting, and A. Toncheva, "The diverse and exploding digital universe," *An IDC White Paper*, vol. 2, 2008.

[2] L. Paterson and T. Roscoe, "The Design Principles of PlanetLab," *Operating Systems Review*, vol. 40, no. 1, pp. 11–16, January 2006.

[3] C. Elliott and A. Falk, "An update on the geni project," *SIGCOMM Comput. Commun. Rev.*, vol. 39, pp. 28–34, June 2009. [Online]. Available: http://doi.acm.org/10.1145/1568613.1568620

[4] A. Gavras, A. Karila, S. Fdida, M. May, and M. Potts, "Future internet research and experimentation: the fire initiative," *SIGCOMM Comput. Commun. Rev.*, vol. 37, pp. 89–92, July 2007. [Online]. Available: http://doi.acm.org/10.1145/1273445.1273460

[5] D. Schwerdel, D. Günther, R. Henjes, B. Reuther, and P. Müller, "German-lab experimental facility," in *Proceedings of the Third future internet conference on Future internet*, ser. FIS'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 1–10. [Online]. Available: http://portal.acm.org/citation.cfm?id=1929268.1929269

[6] I. Chatzigiannakis, S. Fischer, C. Koninis, G. Mylonas, and D. Pfisterer, "Wisebed: An open large-scale wireless sensor network testbed," in *Sensor Applications, Experimentation, and Logistics*, ser. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, O. Akan, P. Bellavista, J. Cao, F. Dressler, D. Ferrari, M. Gerla, H. Kobayashi, S. Palazzo, S. Sahni, X. S. Shen, M. Stan, J. Xiaohua, A. Zomaya, G. Coulson, and N. Komninos, Eds. Springer Berlin Heidelberg, 2010, vol. 29, pp. 68–87. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-11870-8_6

[7] *DES-Testbed A Wireless Multi-Hop Network Testbed for future mobile networks*, Stuttgart, Germany, 06/2010 2010. [Online]. Available: http://www.future-internet.org/files/2010/Folien/Folien_Hahm.pdf

[8] T. Baumgartner, I. Chatzigiannakis, S. P. Fekete, S. Fischer, C. Koninis, A. Kröller, D. Krüger, G. Mylonas, and D. Pisterer, "Distributed algorithm engineering for networks of tiny artifacts," *Computer Science Review*, vol. 5, no. 1, 2011. [Online]. Available: http://dx.doi.org/10.1016/j.cosrev.2010.09.006

[9] A. Arora, E. Ertin, R. Ramnath, M. Nesterenko, and W. Leal, "Kansei: A high-fidelity sensing testbed," *IEEE Internet Computing*, vol. 10, pp. 35–47, 2006.

[10] M. Sridharan, W. Zeng, W. Leal, X. Ju, R. Ramnath, H. Zhang, and A. Arora, "Kanseigenie: Software infrastructure for resource management and programmability of wireless sensor network fabrics," in *Next Generation Internet Architectures and Protocols*, K. M. S. et al., Ed. Springer, New York, 2010.

[11] B. N. Chun, P. Buonadonna, A. AuYoung, C. Ng, D. C. Parkes, J. Shneidman, A. C. Snoeren, and A. Vahdat, "Mirage: a microeconomic resource allocation system for sensornet testbeds," in *Proceedings of the 2nd IEEE workshop on Embedded Networked Sensors*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 19–28. [Online]. Available: http://portal.acm.org/citation.cfm?id=1251990.1253396

[12] G. Werner-Allen, P. Swieskowski, and M. Welsh, "Motelab: a wireless sensor network testbed," in *Proceedings of the 4th international symposium on Information processing in sensor networks*, ser. IPSN '05. Piscataway, NJ, USA: IEEE Press, 2005. [Online]. Available: http://portal.acm.org/citation.cfm?id=1147685.1147769

[13] D. Sakamuri and H. Zhang, "Elements of sensornet testbed design," in *Handbook of Sensor Networks*, H. C. Yang Xiao and F. H. Li, Eds. World Scientific Publishing Co, 2009, ch. 35, pp. 1–36.

[14] sensLAB, "Very Large Scale Open Wireless Sensor Network Testbed." 2010, http://www.senslab.info/.

[15] "Tutornet: A tiered wireless sensor network testbed," pp. 19–28, 2009, http://enl.usc.edu/projects/tutornet/.

[16] V. Handziski, A. Köpke, A. Willig, and A. Wolisz, "Twist: a scalable and reconfigurable testbed for wireless indoor experiments with sensor networks," in *Proceedings of the 2nd international workshop on Multi-hop ad hoc networks: from theory to reality*, ser. REALMAN '06. New York, NY, USA: ACM, 2006, pp. 63–70. [Online]. Available: http://doi.acm.org/10.1145/1132983.1132995

[17] The University of Virginia, "VineLab wireless testbed," 2009, http://www.cs.virginia.edu/~whitehouse/research/testbed/.

[18] Seventh Framework Programme FP7 - Information and Communication Technologies, "Wireless Sensor Networks Testbed Project (WISEBED)," ongoing project since June 2008, http://www.wisebed.eu.

[19] S. Bouckaert, W. Vandenberghe, B. Jooris, I. Moerman, and P. Demeester, "The w-ilab.t testbed," in *TRIDENTCOM*, ser. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, T. Magedanz, A. Gavras, H.-T. Nguyen, and J. S. Chase, Eds., vol. 46. Springer, 2010, pp. 145–154.

[20] T. Baumgartner, I. Chatzigiannakis, M. Danckwardt, C. Koninis, A. Kröller, G. Mylonas, D. Pfisterer, and B. Porter, *Virtualising Testbeds to Support Large-Scale Reconfigurable Experimental Facilities*. Springer, Heidelberg, 2010, pp. 210–223.

[21] S. P. Fekete, A. Kröller, S. Fischer, and D. Pfisterer, "Shawn: The fast, highly customizable sensor network simulator," in *Proceedings of the Fourth International Conference on Networked Sensing Systems (INSS 2007)*, Jun. 2007.

[22] A. Kröller, D. Pfisterer, C. Buschmann, S. P. Fekete, and S. Fischer, "Shawn: A new approach to simulating wireless sensor networks," in *Design, Analysis, and Simulation of Distributed Systems 2005 (DASD' 05)*, Apr. 2005, pp. 117–124.

[23] The WISEBED consortium, "Testbed Runtime wiki and bug tracker," 2010, https://www.itm.uni-luebeck.de/projects/testbed-runtime/.

[24] The SmartSantander consortium, "Smartsantander," 2010, http://www.smartsantander.eu/.

[25] "Projekt 668f, Teilprojekt: Wireless Sensor Networks-Labor (WSNLAB, BSI Projekt)," 2010.

[26] "Sichere Detektion und Lokalisation in kritischen Arealen durch klassifizierende drahtlose Sensornetze (MOVEDETECT, BSI Projekt)," 2010.

[27] Exscal Research Group, Ohio State University, "Extreme scale wireless sensor networking," 2010, http://ceti.cse.ohio-state.edu/exscal/.

[28] Internet2, "Shibboleth," 2010, http://shibboleth.internet2.edu/documentation.html.

[29] The WISEBED consortium, "Testbed Runtime source code hosting at github," 2010, http://github.com/itm/testbed-runtime/.