Sensei-UU: A Relocatable Sensor Network Testbed

Olof Rensfelt
IT Department
Uppsala University
olof.rensfelt@it.uu.se

Frederik Hermans
IT Department
Uppsala University
frederik.hermans@it.uu.se

Lars-Åke Larzon
IT Department
Uppsala University
perg@it.uu.se

Per Gunningberg
IT Department
Uppsala University
lln@it.uu.se

ABSTRACT

A testbed is a powerful complement to simulation and emulation for evaluation of wireless sensor network (WSN) applications. However, testbeds tend to be limited to lab environments and tightly coupled to specific hardware and sensor OS configurations. These limitations, in addition to dependency on local infrastructure make it hard to evaluate applications on actual hardware in the intended target environment.

We introduce Sensei-UU, a WSN testbed designed to be easily relocatable between different physical environments and not tightly dependent on specific sensor hardware or OS. The ability to relocate the testbed enables users to evaluate WSN applications in their intended target environments. The wide range of supported sensor node platforms allows users to evaluate heterogeneous applications. Sensei-UU achieves its flexibility by following a distributed design in which control functionality is put on control machines close to the sensor nodes, and by using a wireless control channel.

We have run experiments to ensure that our wireless control channel does not interfere with the WSN application under evaluation. We show that Sensei-UU can be relocated between environments and that seemingly similar physical locations can have a large difference in radio environment. These differences between locations motivate the need for relocatable testbeds like Sensei-UU.

1. INTRODUCTION

A testbed is a powerful complement to simulation and emulation for evaluation of wireless sensor network (WSN) applications. While testbeds offer the possibility to run code on real hardware with a real sensor OS in a real environment exposed to interference, they are often limited in varying any of these parameters. As a result of these limitations, there may be significant discrepancies between results obtained in a testbed and results from an actual deployment.

We introduce the Sensei-UU testbed, which aims to reduce these limitations of a testbed environment. Rather than being restricted to a lab setting, Sensei-UU can be easily relocated to different environments, enabling WSN applications to be evaluated in different locations. In contrast to many existing testbeds, it supports many different types of sensor hardware and sensor OSs. As Sensei-UU is easily expandable and based on inexpensive hardware and open-source software, it is an attractive alternative to testbeds that are tightly coupled to specific hardware or application scenarios.

The ability to easily relocate a WSN testbed makes it possible to evaluate application behavior in different environments. This is useful as each location has its own characteristics in terms of radio propagation (interfering radio sources, fading effects, signal dampening, etc.) and sensor stimuli. Both radio effects and sensor stimuli may have a significant impact on the application. The ability to run an application *in-situ*, while keeping the benefits of doing it in a testbed setting, makes it easier to evaluate the impact from the surrounding environment in a controlled manner. The alternative would be to simulate or emulate the target environment in a lab, or to simply deploy the application and rely on post-mortem analysis when something breaks.

Deploying the testbed into the target environment also enables using the testbed as a harness for an actual deployment during the verification and debugging phase – when everything is working properly, the testbed components can be removed and one is left with a deployment that has been verified to be operating correctly.

By design, the Sensei-UU testbed makes few assumptions about hardware and sensor OS used. While it certainly supports the commonly used combination of motes and TinyOS[10], it is far from limited to that. We have successfully used a variety of sensor hardware ranging from motes to smartphones, running different sensor OSs like TinyOS, Contiki[6], Linux and Symbian. Experiments can use arbitrary combinations of sensor node configurations, as long as it is possible to control and monitor them remotely. In fact, Sensei-UU does not even assume sensor nodes to be stationary, but rather mobile by default with a stationary location as a special case of mobility. This means that the testbed supports scenarios that include mobility.

The main contribution of the work presented in this paper is a WSN testbed that is not tightly coupled to location, sensor hardware or sensor OS. This simplifies evaluation of WSN applications in different locations using different hardware and OSs. Not being limited to a particular application scenario and also easily expandable, Sensei-UU is an attractive alternative for anyone who wants to evaluate their WSN application in a testbed without being limited to fixed testbed settings or having to develop a new testbed of their own.

The key to an easily relocatable testbed is to have a wireless control channel. This wireless control channel makes Sensei-UU independent of existing infrastructure for communication, but it raises concerns about interference between traffic generated by the control and management of experiments, and the experiments themselves. We have investigated the combination of using IEEE 802.11 as a control channel and standard IEEE 802.15.4 communication in the WSN being evaluated. Our findings are that with care taken when choosing radio channels and with some separation between the different antennas, usage of a wireless control channel is not having a significant impact on the WSN traffic.

Relocating the testbed to a different physical environment will give the testbed user different characteristics that might affect the application running on the sensors. We have run the same experiment at two different locations to ensure that we get similar, yet not identical results. The experiments preserve trends in the results caused by topology and mobility while getting variation caused by the surrounding environment.

In this paper we first describe selected related work on WSN testbeds, followed by an outline of our testbed design. We will then describe our own implementation of the testbed and use it to evaluate interference between the control channel and the WSN, and show how results are affected when relocating the testbed.

2. RELATED WORK

Testbeds have successfully been used to evaluate many aspects of wireless sensor networks. A large number of WSN testbeds follow an indoor setup, in which sensor nodes are attached to a fixed control infrastructure. The control infrastructure consists of low-power computers, such as laptops or WLAN access points, to which one or more sensor nodes are connected, usually over USB. This connection is used for programming, logging data, or injecting sensory data into a sensor node. The control computers communicate with a server over an Ethernet or WLAN connection. Experiment management, such as reprogramming sensor nodes or collecting log data, are performed over the control infrastructure to limit the testbed's influence on the WSN application. Testbeds which follow this general approach include Motelab [23], NetEye [18], Re-Mote [2], Tutornet [22], TWIST [8], and w-iLab.t [4]. These testbeds differ in the hardware components used, their software architecture, user management, and individual features. Deployments of these testbeds comprise up to 204 sensor nodes (in the case of TWIST). TelosB [1] sensor nodes stand out as being the most widely used sensor nodes for such testbeds.

With the exception of Re-Mote, the above mentioned testbeds use the TinyOS serial forwarder to extract log data from sensor nodes. Since the serial forwarder makes assumptions about the format of log data dictated by TinyOS, it makes it harder to use these testbeds with other sensor OSs, such as Contiki. The NetEye testbed also supports the use of a WLAN control channel like Sensei-UU, but has not been designed with relocatability as an explicit goal. Similarly, the other mentioned testbeds were also not designed to be easily relocatable. Furthermore, they often support only a fixed set of sensor node hardware and do not support experiments that include mobile phones.

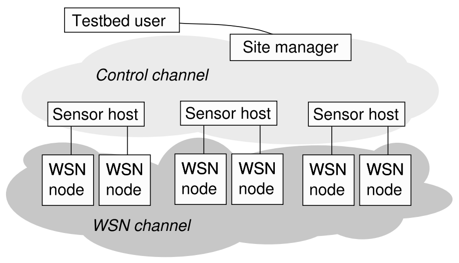


Figure 1: High level design of Sensei-UU.

3. DESIGN OF SENSEI-UU

Sensei-UU is designed to enable users to repeat experiments with mobile, heterogeneous nodes in diverse environments – in contrast to using a static indoor testbed with predefined hardware and sensor OSs. These features makes Sensei-UU a powerful tool for evaluation of WSN applications in their target environment.

To achieve this, Sensei-UU needs to be relocatable with a reasonable overhead to set up the testbed at a new location. This means that the testbed cannot rely on infrastructure (such as an Ethernet network) which may not be available in the environment where the testbed is to be deployed. A wired control channel is not only time consuming to build, but it also limits where sensor nodes, in the WSN to be observed, can be positioned. Therefore, Sensei-UU supports a wireless control channel that offers flexibility when placing nodes and makes deployment and configuration of the testbed faster and easier.

The overall design of Sensei-UU is depicted in Figure 1. Sensor nodes are connected to distributed machines called *sensor hosts*. All sensor-specific software and configuration data is stored only on the sensor host to which the sensor is connected. This approach fits our distributed design and allows for easy relocation of parts of the testbed. Sensor hosts are connected to each other for synchronization and coordination purposes, and connect to a *site manager* over a wireless control channel for control and management. Sensor hosts and the site manager communicate via the control channel, which is separated from the WSN under observation. The site manager provides users with access to the testbed.

3.1 Sensor Hosts

A sensor host is a machine in the testbed that provides means to manage one or more sensor nodes attached to it. Management tasks include reprogramming nodes, controlling power, collecting log data, etc. To have sensor node management at the sensor hosts has two main advantages. First, it simplifies the support of heterogeneous sensor nodes as only the sensor host where a sensor node is attached needs to be aware of the sensor node's hardware specifics. Second, the placement of logging and control functionality on the sensor hosts reduces the demands on the control channel in terms of bandwidth, delay, connectivity etc. As log messages can be filtered at the sensor hosts, log traffic and hence the bandwidth used can be reduced and, because log events are stored locally, continuous connectivity between sensor hosts and the site manager becomes less important.

Since Sensei-UU supports mobile nodes, a sensor host may offer one or more way of localizing itself. The sensor host is responsible for converting the positions obtained with an arbitrary localization method into one common format used by all sensor hosts in the testbed. The sensor management software also provides a common interface to power nodes on and off. To turn off a TelosB node, the power to the corresponding USB port is cut and thus the program is stopped. When the WSN node on a Symbian mobile phone is turned off, the program collecting sensor data instead is stopped.

If a sensor node has a unique identifier that can be easily read by the sensor host, it can be moved between sensor hosts while keeping its identity in the testbed. It will get an estimate of its new position from the sensor host it connects to. This hotplug functionality decreases the time to configure a deployment since there is no need to keep track of which sensor node need to connect to which sensor host.

3.1.1 Mobile Sensor Hosts

One aspect of heterogeneity in WSNs is whether nodes are mobile or static. Mobile nodes introduce new challenges for WSN testbeds. A typical scenario is that a person with a cellular phone moves through a WSN, while tapping the sensor nodes' data. We refer to this type of mobility, where the sensor nodes move distances several times their size, as macro mobility. A mobile sensor node may also do relatively small micro movements as well as macro movements. Vibrations or rotations, where the movement is very small compared to the size of the sensor node, we consider to be micro movements that may need other approaches than what Sensei-UU currently provides. However, that discussion is out of scope of this paper as the focus of mobility in Sensei-UU is on macro mobility.

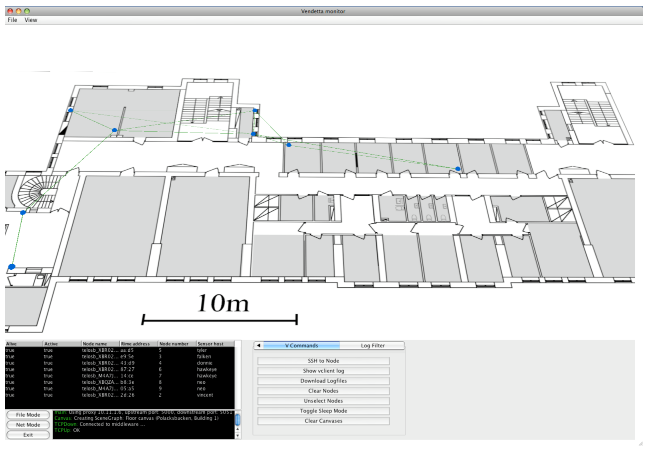
Supporting macro mobility implies that sensor hosts may need to be mobile and follow a mobility scenario. A challenge in such scenarios is to control movements with sufficient precision to support repeatable experiments. The repeatability requirement requires that sensor nodes both move to the same position from one experiment to another and also that the same speed and timing are used. Central to this problem is the ability to localize these nodes in real time to ensure the movement pattern.

Sensei-UU currently supports two types of repeatable movements. First, it supports movements carried out by humans who follow real-time mobility scripts presented on their laptop displays. The scripts describe a mobility scenario with predetermined and known node positions. Different persons may get different instructions to create a choreographed mobility pattern. This method was previously used in the APE testbed [11]. The accuracy of the positioning depends on the ability of the persons moving the nodes to follow the scripts. Second, Sensei-UU supports movements carried out by robots that carry a mobile sensor host with attached sensors. With robots it is easier to move at similar speeds between experiments, but positioning and localization with sufficient precision becomes a challenge – especially if we do not want to be too dependent on a certain location or infrastructure for such purposes.

3.2 Control Channel and Site Manager

As we have a distributed design where sensor nodes may be connected to different sensor hosts, we need a control channel over which sensor hosts communicate between themselves and with the site manager. Instructions to start, stop and reprogram sensor nodes are sent over the control channel, as are events and sensor data detected and collected at sensor hosts.

The site manager is the gateway to Sensei-UU. It provides the user with direct access to the sensor hosts and their attached sensor nodes via the control channel. The site manager is designed to allow a testbed user to control, observe, and manage experiments using different applications. Examples of such applications include a program that forwards data to a database for post-mortem analysis, the monitor GUI described in section 4.3, and a program to coordinate movements of mobile nodes. The site manager can also serve as a time synchronization server for the sensor hosts to ensure that timestamps can be used to synchronize data.



Alive	Active	Node name	Rime address	Node number	Sensor host
true	true	telosb_XBR02...	aa:d5	5	tyler
true	true	telosb_XBR02...	e9:5e	3	falken
true	true	telosb_XBR02...	43:d9	4	donnie
true	true	telosb_XBR02...	87:27	6	hawkeye
true	true	telosb_M4A7J...	14:ce	7	hawkeye
true	true	telosb_XBQZA...	b8:3e	8	neo
true	true	telosb_M4A7J...	05:a5	9	neo
true	true	telosb_XBR02...	2d:26	2	vincent

```
main: Using proxy 10.11.1.6, upstream port: 5000, downstream port: 5051
Canvas: Creating SceneGraph: Floor canvas (Polacksbacken, Building 1)
TCPDown: Connected to middleware ...
TCPUp: OK
```

Figure 2: A screen shot of the monitor, which allows a user fine grained control of the testbed.

4. IMPLEMENTATION AND HARDWARE

We have implemented and evaluated the testbed design outlined in the previous section and used it for different types of experiments. In this section we describe the actual implementation, based on inexpensive commercial off-the-shelf hardware and open-source software.

4.1 Sensor Hosts

We have identified commercially available hardware that suits different applications, and implemented the sensor host management software. Examples of sensor host hardware include a Linux-based broadband router, a Symbian-based smartphone, an Android-based smartphone, and a Linux-based smartphone. In general, any type of machine can be used as a sensor host, as long as it has a network interface to connect to the control channel and an environment where the sensor host management software can be implemented.

The management software for sensor hosts runs on ordinary Linux machines, but some hardware is better suited as sensor hosts. When Sensei-UU is used to evaluate WSN applications built on static TelosB type sensor nodes, a suitable sensor host platform is the Asus WL-500GP wireless access point, which runs a minimalistic distribution of Linux called OpenWrt [12]. The choice to use access points as sensor hosts is based on the good price/performance, the appealing form factor, and the fact that they have IEEE 802.11g interfaces. They also have USB ports, which serve as connection points for sensor nodes.

4.1.1 Mobile Sensor Hosts

If sensor nodes need to be moved while they are managed by Sensei-UU, Openmoko FreeRunner [7] devices are used as sensor hosts. The FreeRunner is an open mobile phone running Linux. It has a GPS receiver, Bluetooth interface, and a IEEE 802.11b/g interface.

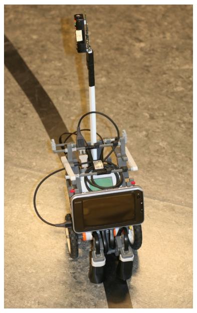


Figure 3: A mobile node in Sensei-UU

The most important feature of the FreeRunner is its ability to act as a USB master which makes it possible to attach external sensor hardware.

Sensei-UU supports experiments that include smartphones. On smartphones, the sensor node and the sensor host are the same physical unit. The management software runs on both Symbian s60 [21] smartphones and Android phones [3]. The Symbian s60 sensor host management software is implemented in Pys60 [15], which is available for the Nokia N95. Pys60 allows access to the sensors on the phones, such as accelerometer, GPS, and cameras. Another type of cell phones included in Sensei-UU is Android phones, such as the HTC Hero. The Android phones run a C-version of the management software with a Java configuration interface. We attach USB sensor nodes to the HTC Hero when we want the mobile phone to be part of the system we evaluate, because Android phones have good user interface capabilities. A limitation is that the HTC Hero cannot power the USB-bus, and therefore we need to supply power with an extra battery pack.

For experiments with mobile nodes, Sensei-UU uses robots to enable repeatable node mobility. The robots are built with off-the-shelf hardware to make them reproducible and affordable for other users. A mobile node in our testbed consists of a Lego NXT robot, a sensor node, and a smartphone. The robot supplies mobility and carries the sensor node and the smartphone. In Fig. 3, a Lego robot carries a smart phone and a TelosB sensor node. Although a custom hardware solution to mobility might offer higher precision control, we argue that the Lego robots offer a better price/performance trade-off with sufficient precision.

Since Sensei-UU is a relocatable testbed, mobile nodes need to be able to easily move and navigate in a new environment. To this end, the robot navigates on a track system that is defined by tape on the floor. The robot follows the track defined by the tape and can be started and stopped arbitrarily by the testbed user. The track system also contains specially marked positions on the track called waypoints, which aid the robot in navigation. We have previously described and evaluated the mobility approach [16, 13].

4.2 Control Channel and Site Manager

The site manager software is implemented in Python and typically runs on a Laptop placed in the center of the deployment. All sensor nodes are attached to the control channel via their sensor hosts, and can be addressed and controlled individually. Currently IEEE 802.11g is used as a wireless control channel, but the design is not limited to IEEE 802.11g; IEEE 802.11a can also be used as well as a wired Ethernet. A requirement is that the reach of the wireless control channel is greater than the range of the sensor nodes, and that the control channel does not interfere with the sensor node radio characteristics.

The range of IEEE 802.11g is often large enough to cover a small deployment. This is the case since sensor hosts do not need to communicate directly to each other; they only need to reach the machine running the site manager. The site manager is therefore preferably positioned in the center of the sensor hosts. In case not all sensor hosts can reach the site manager, the sensor hosts can be connected in a mesh network fashion with the site manager as the sink/source. For longer distances, sensor hosts can be used as relays or WAN technologies can be used to connect to the site manager. Sensei-UU can for example use OLSR [5] or AODV [14] for multi-hop routing between sensor hosts as those protocols are included in the OpenWrt distribution.

Of great importance is the interference on sensor node radios caused by the stronger IEEE 802.11 radios of the sensor hosts. This interference depends on the distances between the sensor nodes and the sensor hosts, as well as on overlap in frequencies. It is of paramount importance to understand and control this interference. We present an evaluation of the effect of a IEEE 802.11g control channel on a IEEE 802.15.4-based sensor network in Sec. 5.1.

4.3 User Interface

A testbed user can connect to a Sensei-UU deployment with different tools. Such tools include a program that can connect the testbed to a database to store experiment data, scripts to control repeatable mobility, and a graphical interface which we will describe here.

The Vendetta software [17] is used for testbed management, to visualize testbed events and to control experiments. Vendetta is a framework for managing distributed testbeds. With Vendetta one can e.g. push new code to sensor nodes, start measurement data collection, power sensor nodes up or down, and control the movement of mobile nodes. These tasks can either be scripted for a batch type of experiments or performed interactively, for example during code development and debugging. The monitor is a Java application that controls and visualizes the progress of an experiment (Figure 2). Control and monitoring tasks are defined in configurations files and implemented as buttons in the user interface.

Measured data, such as packets moving between nodes and real-time energy levels, are visualized in the monitor. The visualization is done in a Java3D canvas. The 3D environment is useful when experiments are spanning multiple floors. The graphical representation of the testbed is also useful when deploying new nodes, because the position of a sensor node can be set by moving the node in the graphical interface. The updated position is pushed to the sensor host and incorporated in the logging of the sensor node.

5. EVALUATION

A concern when using a wireless control channel is that it may interfere with sensor node radios and electronics. This is especially of relevance when overlapping frequencies are used, e.g., when an IEEE 802.11b/g control channel and an IEEE 802.15.4-based sensor network are co-located. We will in this section present the impact of such interference on the WSN communication as a function of distance between sensors nodes and sensor hosts.

When the testbed is relocated to a new location, we expect to get different radio characteristics due to differences in fading, interference, etc. However, we do expect to get similar characteristics related to the layout of the experiment, e.g., topology. We will present results from running the same experimental setup at two different locations.

5.1 Control Channel Interference

Previous research [19, 20] suggests that IEEE 802.11b and IEEE 802.15.4 can coexist if non-overlapping frequencies within the 2.4 GHz ISM band are used. The interference caused by IEEE 802.11b on IEEE 802.15.4 has been studied analytically and in simulation [19] with regard to Packet Error Rate (PER), transmission delay and throughput as metrics. The conclusion is that both technologies can coexist if channels are assigned carefully. Other empirical work also showed similar results [20].

Our specific concern in Sensei-UU is that some sensor nodes are attached to the sensor hosts via USB, meaning that nodes are located nearby their sensor hosts. We ran experiments to investigate how this proximity affects interference. In the experiments we vary the distance between a transmitting sensor host and a sensor node for different IEEE 802.15.4 channels. We use an instance of the Sensei-UU testbed to run the experiments.

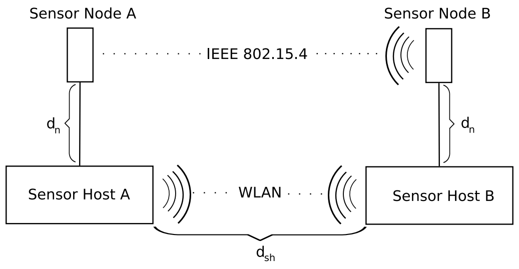


Figure 4: Experimental setup of interference evaluations.

5.1.1 Experimental Setup

The general setup of our experiment is shown in Figure 4. Two sensor nodes were connected to two different sensor hosts. Sensor node B sent packets to sensor node A while the sensor hosts created artificial traffic on the IEEE 802.11g to potentially disrupt the communication between the sensor nodes.

We used a Contiki program to generate the sensor node communication. In order not to hide any collisions, we disabled the IEEE 802.15.4 MAC layer on the sensor nodes so that no acknowledgments or retransmissions are used. The results can thus be seen as a worst-case study and it is most likely that the IEEE 802.15.4 performance will be improved when enabling the MAC layer.

The IEEE 802.11g was set to use channel 1, and the artificial traffic on the control channel was constant bitrate UDP traffic generated by Iperf [9] saturating the link. The transmission power of the IEEE 802.11g was set to 19 dBm, and the transmission power of the IEEE 802.15.4 radio transmitter was fixed to 0 dBm. In each run, sensor node B sent 200 packets over IEEE 802.15.4 to sensor node A. Sensor node A logged every packet it successfully received. We used Tmote Invents from Moteiv with a CC2420 radio transceiver as sensor nodes and ASUS WL-500GP access points with a Broadcom 4318 IEEE 802.11g card as sensor hosts. The presented results were all gathered during night time in our university building. The reason to perform the experiments during the night is to limit the influence of other wireless communications within the building to increase the probability that measured interference is caused by control channel traffic, rather than other IEEE 802.11g networks.

Each run of the experiment was repeated 10 times. Every run was also repeated without IEEE 802.11g traffic to serve as a base and reference case to estimate the background noise.

The following parameters were varied between different runs:

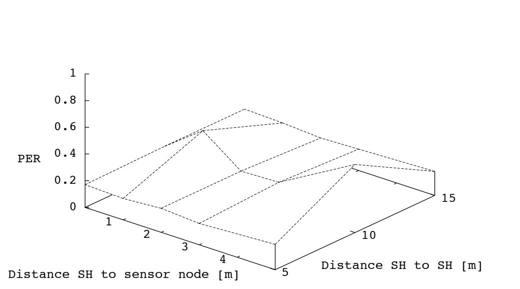
- The IEEE 802.15.4 channel was set to either 12 or 16. Channel 12 overlaps with IEEE 802.11g channel 1, whereas channel 16 is outside of IEEE 802.11g channel 1. We have previously tested all IEEE 802.15.4 channels and channel 16 showed the most interference of the non-overlapping channels. Thus, in our set-up, channel 12 has the worst overlap and channel 16 represents the worst non-overlap case.
- The sensor host generating the IEEE 802.11g traffic was alternated, so that IEEE 802.11g and IEEE 802.15.4 traffic either flowed in parallel or in opposite directions. The reason to alter the direction of the interfering traffic was to investigate the effects of close electronic interference of both the IEEE 802.15.4 receiver and sender.
- The distance d_n between the sensor nodes and their sensor hosts was set to 0 m, 1 m, 2 m, 3 m and 5 m. The distance between the sensor hosts d_sh was set to 5 m, 10 m and 15 m.

5.1.2 Interference Results

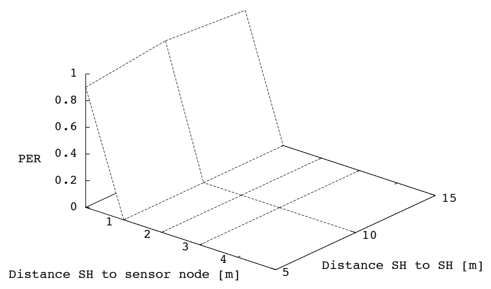
We measured the Packet Error Rate (PER) for channel 12 and 16 when the senders are co-located. PER is the ratio of the number of packets not received to the total number of packets sent. The results are presented in Figure 5. In the case when the two traffic flows went in opposite directions, we observed little interference. Due to space constraints, we decided to omit this data here, and instead focus on the case were traffic flows in the same direction (i.e., from sensor node B to sensor node A, and from sensor host B to sensor host A).

Figure 5a shows that the PER is not linearly correlated with the distance between sensor hosts, as the PER does not increase when the distance between sensor hosts increase. The two points with a higher PER (when d_sh is 10 m while d_n is 1 m and 5 m) also have a very high standard deviation (Table 1), i.e., during some of the runs with these configurations a lot of packets were lost, whereas during other runs with the same configuration packet loss was low. From manual inspection of our log files we could confirm that the results from those two configurations are divided into two clusters. The first cluster of measurements has PER values similar to other measurements, while the second cluster has substantially higher PER values. The runs with high PER values are in sequence, which leads us to suspect that an interference source outside of our control has been active during these runs. The additional interference during those two measurements can also be seen in Table 1. It shows as a non-zero value when there is no IEEE 802.11g traffic between the sensor hosts. We have not further investigated the origin of this interference, nor have we made extra measurements, as we do not plan to use overlapping channels for WSN traffic and control channel.

Figure 5b shows PER values when non-overlapping channels are used. The results are less intuitive. The PER values when the sensor nodes are positioned on top of the sensor hosts show a collapse of IEEE 802.15.4 communication with PER values close to



(a) Overlapping channels (IEEE 802.15.4 channel 12 and IEEE 802.11g channel 1). The senders are co-located at the same sensor host.



(b) Non-overlapping channels (IEEE 802.15.4 channel 16 and IEEE 802.11g channel 1). The senders are co-located at the same sensor host.

Figure 5: Packet Error Rate versus the distance between sensor nodes and sensor host (SH), and the distance between the sensor hosts.

1. When the sensor hosts are 10 meters apart 97 % of the IEEE 802.15.4 packets are lost. We have seen similar trends at other channels, but IEEE 802.15.4 on channel 16 had the highest PER values. We suspect that circuits on the Tmote Invent are suffering from interference, creating a high PER, rather than it being radio collisions. Further investigation is needed to understand the phenomenon in detail, but for our purpose of investigating the feasibility of a 802.11g control channel, this evaluation suffices.

In summary, we draw the conclusion that a IEEE 802.15.4-based sensor network does not suffer from interference of the testbed's wireless control channel if non-overlapping channels are used and sensor hosts and sensor nodes are always separated by at least 1 m. It is worth noting that other sources of interference, such as a university WLAN network or microwave ovens, may still potentially interfere with the sensor network communication. However, we believe that such a degree of realism (interference from another network) is desirable when one wants to evaluate a sensor network in a testbed. Furthermore, complete shielding from uncontrolled interference sources is outside of the scope of a testbed and can only be achieved in a shielded environment such as an anechoic chamber. What we have shown in this section is that the influence of the testbed's wireless control channel on the sensor network communication is low under the stated precautions.

5.2 Relocatability

As Sensei-UU is relocatable, we have used it in different locations. Here, we show how measurements in two seemingly similar corridors differ. The experiments were run in two different buildings. First in our lab called the Angstrom building and then in another building called Polacksbacken.

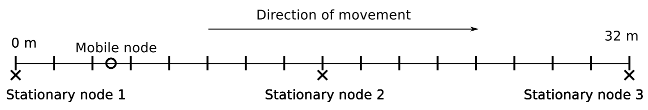


Figure 6: Experimental setup for evaluation of link characteristics in a corridor environment

The overall setup for the two experiments is shown in Figure 6. A robot carries a sensor node along a 32 m long, straight track. Three stationary sensor nodes are placed 0.5 m next to the track at

d_{sh} [m]	d_{n} [m]	Traffic direction	channel	PER	stddev
10.0	0.0	none	12	0.0	0.0
10.0	0.0	none	16	0.01	0.01
10.0	0.0	same direction	12	0.18	0.04
10.0	0.0	same direction	16	0.97	0.01
10.0	0.0	opposite direction	12	0.15	0.02
10.0	0.0	opposite direction	16	0.82	0.03
10.0	1.0	none	12	0.01	0.0
10.0	1.0	none	16	0.0	0.0
10.0	1.0	same direction	12	0.39	0.21
10.0	1.0	same direction	16	0.0	0.0
10.0	1.0	opposite direction	12	0.2	0.03
10.0	1.0	opposite direction	16	0.0	0.0
10.0	2.0	none	12	0.0	0.0
10.0	2.0	none	16	0.0	0.0
10.0	2.0	same direction	12	0.18	0.05
10.0	2.0	same direction	16	0.0	0.0
10.0	2.0	opposite direction	12	0.0	0.0
10.0	2.0	opposite direction	16	0.0	0.0
10.0	3.0	none	12	0.01	0.01
10.0	3.0	none	16	0.0	0.0
10.0	3.0	same direction	12	0.19	0.03
10.0	3.0	same direction	16	0.0	0.0
10.0	3.0	opposite direction	12	0.01	0.01
10.0	3.0	opposite direction	16	0.0	0.0
10.0	5.0	none	12	0.02	0.01
10.0	5.0	none	16	0.0	0.0
10.0	5.0	same direction	12	0.51	0.36
10.0	5.0	same direction	16	0.0	0.0
10.0	5.0	opposite direction	12	0.04	0.03
10.0	5.0	opposite direction	16	0.0	0.0

Table 1: Packet Error Rates when the distance between sensor nodes and sensor hosts and 802.15.4 channel is varied. The sensor hosts are sending on IEEE 802.11g channel 1 which overlaps with IEEE 802.15.4 channel 12 but not with channel IEEE 802.15.4 channel 16.

0 m, 16 m, and 32 m. TelosB sensor nodes are used for both the stationary sensor nodes and the mobile sensor node.

We use a Contiki application to measure the RSSI of packets received by the mobile node. The stationary nodes send PING packets to the mobile node and the mobile node responds with a PONG packet that contains the RSSI value with which the PING packet was received. The three stationary nodes send PING packets in a strict round-robin fashion to avoid packet collisions.

Figure 7b shows RSSI from the Polacksbacken building and Figure 7b from the Angstrom building. Comparing Figure 7a with Figure 7b, the three nodes have the highest RSSI readings at similar positions. However, the three curves have different local peaks at the two sites. We believe that the differences are due to the particular corridor structures in the two buildings such as different materials in the walls, etc.

If a WSN application would be deployed in these two different environments, the performance could differ. For example, if a node picks its routing neighbors by the highest RSSI, the mobile node would change from node 1 to node 2 at 5 m in the Angstrom building while the corresponding change would at earliest take place at 10 m in the Polacksbacken building. This type of differences between environments is what Sensei-UU is designed to evaluate.

The ability to set up Sensei-UU in different environments allows performance experiments not only in the lab, but also in potential target environments. Figure 7 shows that the environment might have a significant impact on a WSN application, depending on how sensitive the application and its underlying protocols are to RSSI changes.

From our measurements, we conclude that Sensei-UU is relocatable between different environments and that seemingly similar locations can have large variation in radio characteristics.

6. FUTURE WORK

An important aspect when running experiments is to understand the environmental conditions – especially when the same experiment is moved between different target environments. We plan to extend the testbed with functionality for fingerprinting, i.e., characterizing conditions and properties of experiments to make it easier to compare experiments that run under different conditions to each other.

In our current implementation of the testbed, we have made the implicit assumption that sensor hosts are always reachable from the site manager. There may be situations when this is not always the case, e.g., if the control channel provides intermittent connectivity or is partitioned, or if there are large delays in the communication over the control channel. These problems can be addressed with a delay-tolerant control channel where sensor hosts continue to operate autonomously in the absence of constant communication with the site manager.

7. CONCLUSIONS

We have presented Sensei-UU, a testbed for wireless sensor network applications that is designed to be easily moved between different environments, simple to setup and work with while supporting a variety of sensor hardware and sensor OSs. The sensor host software has been ported to different platforms, enabling us to build a testbed including wireless routers, Symbian smartphones, Android phones and Openmoko FreeRunners as sensor hosts to which sensor nodes are attached. As we use a wireless control channel in the testbed, we have investigated the interference between the control channel and the sensor network to ensure that experiments are not affected by control channel traffic. Our measurements show that, if care is taken when channels are selected and nodes are placed, the control channel will not interfere with the experiments.

Our experience with Sensei-UU is so far very positive. During the design and development, we have configured and reconfigured Sensei-UU deployments more than twenty times at different locations. While working with Sensei-UU, we have found the design highly flexible and extensible. For example, a deployment with four sensor hosts and seven sensor nodes will fit in a small backpack and can be configured in well under half an hour. We have also found that the distributed design, where much functionality is put at the sensor hosts, makes it easy to incorporate different types of sensor hosts within one deployment. An example of such deployments is when mobile nodes are included. The design then makes it easy to change which type of localization is used. Therefore we think that Sensei-UU meets the requirements of being a relocatable testbed with support for heterogeneous mobile nodes.

From our interference measurements we conclude that the interference between IEEE 802.11g and IEEE 802.15.4 can be controlled, provided that not overlapping channels are used and that the sensor nodes are enough separated from the sensor hosts. Still we advocate that all Sensei-UU experiments should be proceeded by interference measurements since the interference is highly depending on sensor hardware and possibly also on the actual environment.

We also show that our testbed can be moved between environments and that seemingly similar environments can show different radio characteristics.

Our ambition is to make the testbed design and software available for other researchers in the field. Sensei-UU is licensed under the Gnu General Public License (GPL) and will be publicly released together with configuration instructions for sensor hosts.

Acknowledgments

This work was carried out within the Uppsala VINN Excellence Center for Wireless Sensor Networks WISENET, partly supported by VINNOVA. We would also like to acknowledge Joel Samuelsson, Fredrik Hildorsson, and Hjalmar Wennerström for their work to incorporate the mobile sensor host in Sensei-UU.

8. REFERENCES

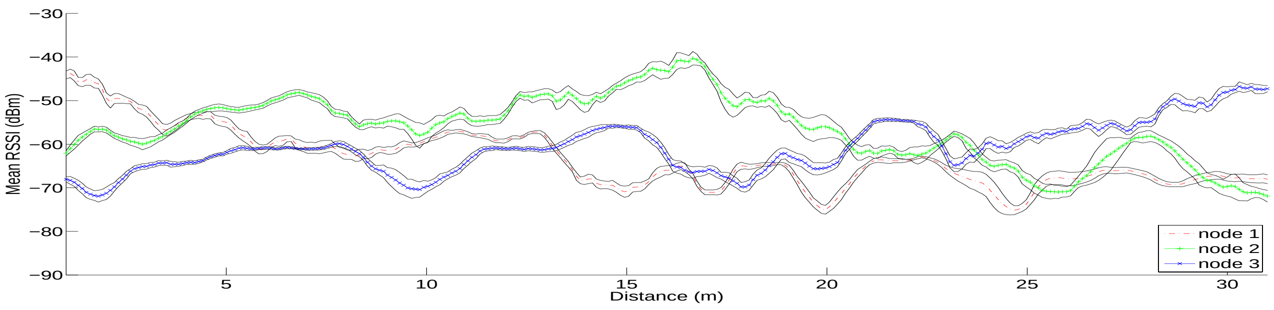
[1] Crossbow Technology, Inc.: *TelosB datasheet*. `http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/TelosB_Datasheet.pdf`.

[2] Re-Mote, Testbed Framework. `http://code.google.com/p/remote-testbed/`.

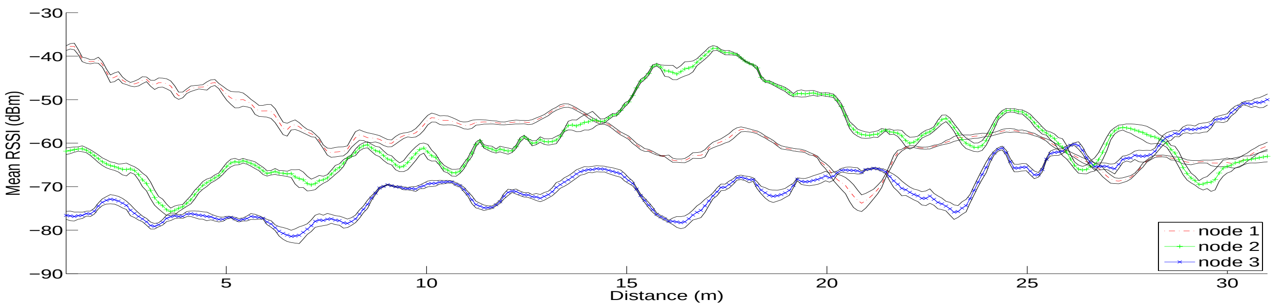
[3] Android. `http://www.android.com/`.

[4] S. Bouckaert, W. Vandenberghe, B. Jooris, I. Moerman, and P. Demeester. The w-ilab.t testbed. In *TridentCom 2010: Proceedings of the 6th International ICST Conference on Testbeds and Research Infrastructure for the Development of Networks & Communities*, 2010.

[5] T. Clausen, P. Jacquet (editors), C. Adjih, A. Laouiti, P. Minet, P. Muhlethaler, A. Qayyum, and L.Viennot. Optimized link state routing protocol (OLSR). RFC 3626,



(a) Mean RSSI and standard deviation in Angstrom building from 10 runs smoothed over a 1 m window.



(b) Mean RSSI and standard deviation from the Polacksbacken building from 10 runs smoothed over a 1 m window.

Figure 7: RSSI measurements from two experiments in different locations.

October 2003. Network Working Group.

[6] A. Dunkels, B. Grönvall, and T. Voigt. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *LCN '04: Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks*, pages 455–462, Washington, DC, USA, 2004. IEEE Computer Society.

[7] Openmoko freerunner. `http://www.openmoko.org`.

[8] V. Handziski, A. Köpke, A. Willig, and A. Wolisz. Twist: a scalable and reconfigurable testbed for wireless indoor experiments with sensor networks. In *REALMAN '06: Proceedings of the 2nd international workshop on Multi-hop ad hoc networks: from theory to reality*, pages 63–70, New York, NY, USA, 2006. ACM.

[9] Iperf, a tool for measuring Internet bandwidth performance. `http://sourceforge.net/projects/iperf`.

[10] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler. TinyOS: An operating system for wireless sensor networks. In *Ambient Intelligence*. Springer-Verlag, 2004.

[11] E. Nordstrom, P. Gunningberg, and H. Lundgren. A testbed and methodology for experimental evaluation of wireless mobile ad hoc networks. In *Proceedings of the Tridentcom*, volume 2005, 2005.

[12] OpenWrt. `http://www.openwrt.org/`.

[13] O.Rensfelt, F. Hermans, C. Ferm, P. Gunningberg, and L.-Å. Larzon. Sensei-UU: A Nomadic Sensor Network Testbed Supporting Mobile Nodes. Technical Report 2009-025, Department of Information Technology, Uppsala University, October 2009.

[14] C. Perkins, E.M. Royer, and S.R. Das. Ad hoc On-Demand Distance Vector (AODV) Routing, 2003.

[15] Python for Symbian s60. `http://sourceforge.net/projects/pys60/`.

[16] O. Rensfelt, F. Hermans, P. Gunningberg, and L. Larzon. Repeatable experiments with mobile nodes in a relocatable wsn testbed. In *MobiSensor '10: 1st International Workshop on Mobility in Wireless Sensor Networks*, 2010.

[17] O. Rensfelt, L.-Å. Larzon, and S. Westergren. Vendetta - a tool for flexible monitoring and management of distributed testbeds. In *Proc. of third International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities*, may 2007.

[18] D. Sakamuri. NetEye: A wireless sensor network testbed. Master's thesis, Wayne State University, Detroit, Michigan, 2008.

[19] S. Y. Shin, H. S. Park, and W. H. Kwon. Mutual interference analysis of IEEE 802.15.4 and IEEE 802.11b. *Computer Networks*, Volume 51:pp. 3338–3353, August 2007.

[20] A. Sikora. Compatiblity of IEEE 802.15.4 (ZigBee) with IEEE 802.11 (WLAN), Bluetooth and Microwave Ovens in 2.4 GHz ISM-Band. Technical report, University of Cooperative Education Loerrach, 2004.

[21] Symbian. `http://www.symbian.org`.

[22] Tutornet: A Tiered Wireless Sensor Network Testbed. `http://enl.usc.edu/projects/tutornet/`.

[23] G. Werner-Allen, P. Swieskowski, and M. Welsh. Motelab: a wireless sensor network testbed. In *IPSN '05: Proceedings of the 4th international symposium on Information processing in sensor networks*, page 68, Piscataway, NJ, USA, 2005. IEEE Press.