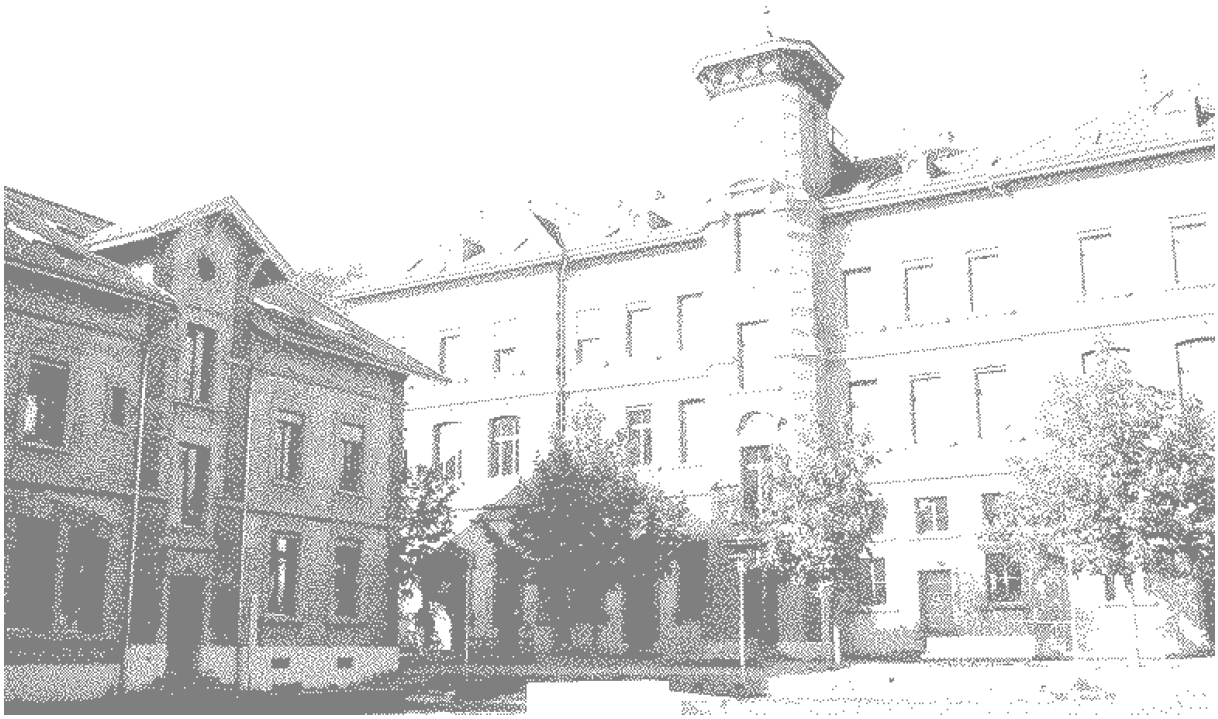
UNIVERSITÄT
BERN

Adding Virtual Mobility to a Federated Testbed for Wireless Sensor Networks: a Proposal

G. Coulson, T. Braun, T. Staub

Technischer Bericht IAM-10-004 vom 30. August 2010

Institut für Informatik und angewandte Mathematik, www.iam.unibe.ch



Adding Virtual Mobility to a Federated Testbed for Wireless Sensor Networks: a Proposal

Geoff Coulson, Torsten Braun, Thomas Staub

Technischer Bericht IAM-10-004 vom 30. August 2010

CR Categories and Subject Descriptors:
C.2.1 [Computer-Communication Networks]: Network Architecture and Design; C.2.2 [Computer-Communication Networks]: Network Protocols; C.2.3 [Computer-Communication Networks]: Network Operations; C.2.4 [Computer-Communication Networks]: Distributed Systems

General Terms:
Design, Management, Measurement, Experimentation

Additional Key Words:
wireless sensor networks, testbed, mobility, emulation, simulation

Institut für Informatik und angewandte Mathematik, Universität Bern

Abstract

In this document we present an initial design that accommodates ‘virtual mobility’ into testbeds for wireless sensor networks. The virtual mobility of physical, simulated or emulated nodes is treated in a uniform manner by embedding the nodes in a virtual space. The virtual space is formed by a simulation model and handles all the traffic from the nodes. The traffic of physical nodes is therefore intercepted and redirected to this model. We discuss the aspects of ‘virtual mobility’ and provide an initial design that supports ‘virtual mobility’ across a federated testbed for wireless sensor networks.

Contents

1 Introduction

Research on wireless sensor networks (WSNs) has grown rapidly in recent years, and large-scale experimental deployments of WSNs are now becoming widespread [1]. This rapid growth has led to a strong emerging requirement for *flexible experimental facilities* to support not only the design and evaluation of new protocols and mechanisms for WSNs, but also the planning of large-scale, real-world, WSN deployments. To this end, a number of testbeds for WSNs have been built, among them the WISEBED testbed [2] (see www.wisebed.eu) that forms the background of the present proposal.

WISEBED is an extremely flexible testbed that supports experimentation at all levels from low-level communications to applications. It addresses the need for flexibility through the concept of *virtual testbeds*, as discussed in Section 3.2, in which physical, simulated and emulated testbed elements can be freely mixed. So far, WISEBED has explored the virtualisation of a number of key aspects of the testbed environment, but the area of *virtual mobility* has been identified as an area that needs further development. This document is therefore an attempt to further explore virtual mobility in the WISEBED context and to set out a roadmap for its integration into WISEBED.

The remainder of this proposal is structured as follows. In Section 2, we survey relevant related work and then, in Section 3, we provide background on the key WISEBED-derived concepts needed on which to build virtual mobility. Then, in Section 4 we discuss the concept of virtual mobility in detail and consider general issues for the design of a support infrastructure to support virtual mobility. Then, in Section 5 we put froward an 'straw man' design for a virtual mobility infrastructure, raising further design issues as we proceed. Finally, in Section 6 we offer our conclusions and discuss the way forward.

2 Related work

The research community has so far relied on three main approaches to the design and construction of WSN testbeds: *physical testbeds*, *simulation* and (to a much lesser extent) *emulation*:

- *Physical testbeds* (e.g. [3, 4, 5, 6, 7, 8]) are extremely useful for the high-fidelity evaluation of mature WSN designs, and for the detailed planning of real-world deployments. However, physical testbeds for WSN systems tend to be small in scale, expensive to maintain, and time-consuming to set up. They are also typically *lacking in flexibility*, often offering only a single, fixed, connectivity topology or being limited in their support for heterogeneity (e.g., offering only a single type of sensor node, radio, operating system or programming language). They also tend to be limited in their programmability at lower levels of the system (e.g., many use fixed OSs and networking stacks). Furthermore, they are often unsuited to experimentation scenarios that require *repeatability* of experiments, as many relevant operating parameters are outside the user's control (e.g. local radio interference from infrastructure or other experiments).

- *Simulation* (e.g. [9, 10, 11]) offers complementary characteristics: it is useful for quickly trying out new ideas, and for investigating the behaviour of new protocols and mechanisms in varied topologies, at a large scale and in a repeatable manner. However, the big drawback of simulation is its *lack of fidelity* – e.g., it is often not realistic to simulate fully at the instruction execution level, or with high fidelity radio or power consumption characteristics. While such limitations are not necessarily problematic in traditional network environments, where simulators such $ns2$ [11] are prominent, they are serious drawbacks in WSN environments where resource scarcity and incidental physical characteristics are of the essence. Therefore, simulation is of limited use in planning the practicalities of real-world WSN systems and deployments.

- *Emulation* (e.g. [12, 13]) is situated between the end points of physical reality and simulation. Whereas simulation *abstractly models* a target system, emulation *duplicates* the functionality of one system in terms of another. It is therefore capable of much greater fidelity than simulation, while potentially also offering greater flexibility than a purely physical testbed. Emulation is a much less exploited approach in the WSN testbed context, but one for which there is much

potential. For example, emulation in the form of network overlay technology could be used to support different inter-node connectivity patterns in a physical testbed; or, a battery-based power supply on a physical node could be emulated by interposing a mains electricity powered hardware module that degrades power over time.

In the following sections we examine related work—i.e. work related to virtual mobility—in each of these areas in turn.

2.1 Physical wireless sensor network testbeds

Physical testbeds allow us to perform experiments in real-world environments and have been succesful for developing applications and protocols in the Internet. PlanetLab [14] is a global research network that supports the development of new network services. Since the beginning of 2003, more than 1,000 researchers at top academic institutions and industrial research labs have used PlanetLab to develop new technologies for distributed storage, network mapping, peer-to-peer systems, distributed hash tables, and query processing. PlanetLab currently consists of more than 1000 nodes at more than 500 sites. PlanetLab is rather suited for experiments on application level and for overlay networks. PlanetLab provides only little support for wireless network applications and protocols. In [15] the authors describe a mobile node running the PlanetLab software enabling its inclusion into the PlanetLab network but this does not support mobility per-se, just the inclusion of mobile communication technologies (e.g. UMTS).
A number of testbed activities on European level aim to build pan-European testbed infrastructures for future Internet research [16]. These projects are running under the FIRE (Future Internet Research and Experimentation) umbrella. Most of them such as OneLab [17], PII [18], and Federica [19], aim at fixed networks. PII implements an infrastructure for federating (fixed network) testbeds. It aims to develop a common control framework to interconnect and federate different testbeds. Federica is developing an European testbed infrastructure based on optical transmission facilities and nodes, which are capable of virtualization to host these experimental activities on future Internet architectures and protocols. There is no support for wireless or mobile networks. Onelab is based on PlanetLab technology and covers European PlanetLab sites. An activity within Onelab addresses wireless network testbeds and is based on the ORBIT testbed [20]. ORBIT provides a configurable indoor radio grid for controlled experimentation and an outdoor wireless network for testing under

real-world conditions. The indoor radio grid offers a controlled environment as an isolated network, in which background interferences can be injected. Although the 20x20 grid of nodes offers a large variety of different topologies, it is probably too restrictive for many scenarios, and mobility tests are even more limited. Furthermore, the scarce ORBIT resources may be not available for all experiments. ORBIT does not comprehensively address mobility and is rather focused on wireless mesh network like testbeds.
Experimentation in wireless mesh networks (WMNs) has also been quite popular during recent years. Reference [21] gives a good overview of earlier wireless mesh network testbeds. Several testbeds and experiments have been established to explore and evaluate wireless mesh networks in campus and city networks [22]. Experiments with real-world deployments have proven the usability of directional antennas for wireless radio networks to connect nodes over long distances [23]. Heraklion MESH [24], WildNet [25], and Quail Ridge Reserve WMN [26] successfully interconnect nodes by directional antennas, providing cheap, stable and robust broadband network access using low-cost radio technology. DOME (Diverse Outdoor Mobile Testbed) has been built for large-scale outdoor experimentation and monitoring in a vehicular network [27] based on wireless mesh technology. In [28] throughput experiments using trains in a network of wireless access points have been performed. These data rather served as input for more comprehensive simulation models. In the case of APE (Ad hoc Protocol Evaluation) [29] students carry laptops in a coordinated way through buildings and measurements are performed during that movement. Results are processed by central entities. Reproducibility is somewhat difficult, since situations of wireless channels may change and mobility patterns are not exactly the same for each experiment. The Illinois Wireless Wind Tunnel (iWWT) [30] is a reduced-scale testing environment for wireless networks implemented in an electromagnetic anechoic chamber. Its main goal is to create a realistic scaled version of the wireless environment maintaining full control over all relevant parameters that affect the performance of the wireless network like obstructions, interferers, etc. Mobility is supported by placing the wireless hosts (laptops, PDAs, sensor nodes) on top of remotely controlled cars. Despite these efforts for complete control of the RF environment, repeatability of small-scale experimental results remains elusive due to intrinsic randomness in the evaluated protocols, and object positioning errors. Several wireless mesh networks have been built in university campuses, e.g., the DES-Testbed (Distributed Embedded Systems) [21] at Berlin. Those wireless mesh network testbeds can be easily extended to wireless sensor network testbeds by directly connecting sensors to the mesh nodes. This approach has also

been followed in the WISEBED project, see below [31].
Physical wireless sensor network testbeds [3, 4, 5, 6, 7, 8, 32] have been developed to support high-fidelity evaluation of mature wireless sensor network designs, and detailed planning of real-world deployments. However, physical testbeds for WSN systems tend to be small in scale, expensive to maintain, and time-consuming to set up. Mobility is rather difficult to maintain since the sensor nodes must be controlled and therefore remain inside a certain region of controller nodes. Moreover, while planning mobility of nodes, accidents must be avoided. Reproducibility of experiments is challenging, since network conditions may vary significantly in mobile scenarios. Trio [3] is one of the largest wireless sensor testbeds yet built, consisting of 557 solar-powered motes. However, Trio was targeted at a specific application and not designed for multiple or external users.
MoteLab [4] is a smaller indoor camus-based sensor network testbed (featuring 190 Tmote Sky sensor nodes) that was specifically designed for ease of programming by external users (e.g. it provides a web-based interface and incorporates a hardware backbone for direct node access). However, like most physical testbeds, it lacks flexibility, providing only homogenous nodes and a fixed connectivity topology. TWIST [5] has around 200 nodes, with a degree of heterogeneity and a 3-tier network topology that is configurable to a limited extent. TutorNet [6] (with around 100 nodes) also uses a three-tier topology but without any configurability. However, it has good user support with command-line tools enabling the control of individual nodes, and a system for user authorisation.
Intel SensorNet [7] (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.
CitySense [33] 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. There are also a few recent initiatives that apply the concept of federation to WSN testbeds. For example, [34] combines the infrastructure and software of the Kansei testbed with the GENI facility [35] to provide a unified solution. Kansei [8] currently provides around 200 sensor nodes, with gateway stations attached to each one of the sensor nodes. Kansei features a web-based interface and supports the visualisation of sensor readings and remote debugging. The Senslab project [36] aims to unify four discrete heterogeneous testbeds into a single one of 1000 nodes. SENSEI [37] aims 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. While all these wireless sensor network testbeds allow experiments in real-world scenarios, mobility support is not available. One approach to the support of mobile wireless sensor network testbeds is to carry nodes by robots within a controlled testbed environment. TrueMobile (Mobile Emulab wireless sensor network testbed, [38]) is an extension to the popular EmuLab wireless ad-hoc networks testbed. Robots carry motes and single board computers through a fixed indoor field of sensor-equipped motes, all running the user's selected software. MiNT-m [39] is a testbed designed to be deployed in small environments. MiNT-m consists of nodes comprising a wireless computing device and a mobile robot for physical movement. The nodes are controlled by a control server using a different frequency range than the nodes are using for their inter-node communication. The mobile nodes are each confined to a sector in which they can move, lest the wires connecting to the PCs entangle. MiNT-m does not provide a localization system, robots are tracked by cameras reporting to tracking server. Reference [40] describes a wireless sensor network testbed for mobile data communication. The testbed is implemented with one base station node, one 'data mule' and five sensor nodes, where data mules move in a random manner through a community of fixed sensor nodes. When the mule comes within range of each sensor node, it collects the available data. After collecting data from all the sensor nodes, the mule returns to a central node, where it uploads all the data collected. The motes are connected to the central node through a LAN. The motes' behavior when communicating with the mule can be monitored in real time.

2.2 Simulation environments for wireless sensor networks

A number of simulators have been developed and/or extended to allow modeling and simulation of WSNs. Several simulators for wireless sensor networks [41, 42, 9, 10, 43, 11] have been implemented and are widely used in the research community. Some of them simulate rather application level behaviour and do not sufficiently simulate wireless sensor network characteristics. Another shortcoming is lack of support of application and system level processing. To address this problem, MSPSim/Cooja [44] and AvroraZ [45] support the simulation of instruction level processing on sensor nodes for different hardware platforms such as MSP and MicaZ

nodes.
There are a few approaches that have attempted to combine physical elements with simulation and emulation in the area of wireless sensor networka. The *TOSSIM* system [10, 46] has attempted to bridge the gap between simulation and real-world performance by exploiting the TinyOS programming model to generate discrete-event simulations directly from TinyOS component graphs. The resulting platform is primarily a simulator with a small number of attached physical nodes as opposed to a true integration in which the simulated and the physical are equal partners. Other examples of systems in which part of an experiment is conducted in simulation and part on physical hardware are [47, 48, 49, 50]. In [48] and [49] only the wireless communication channel of the real devices is utilised, with the rest of the software being executed inside a simulator. In [47] and [50] the software is executed iteratively on real and simulated devices with certain arbitration and timing schedules applied. Mobility issues are not addressed in those systems.
In general, mobility support is rather limited in wireless sensor network simulation environments, except for the simulators also supporting other network scenarios such as mobile ad-hoc networks. While many mobility models have been developed for fixed and wireless network evaluations, little work has been done to extend wireless sensor network simulation tools by mobility models. BonnMotion [51] has developed several (standard) mobility models such as Random-waypoint or Reference Point Group Mobility to be integrated into simulation tools such as ns-2, OMNet++ and Cooja.

2.3 Emulation for wireless (sensor) networks

As mentioned, emulation is capable of much higher fidelity than simulation, while potentially also offering greater flexibility than a purely physical testbed. In the case of network emulation, the characteristics of the underlying network is reproduced, e.g., by use of simulation or system-level implementation. Emulation can approximate the real environment more accurately than pure simulation. In particular, processing delays introduced by applications, operating system and hardware are ignored in most simulation models. This is, however, important to perform evaluations of complete systems in real-world environments.
The combination of node virtualization and network emulation has been proposed by various researchers [52, 53, 54]. The approach presented in [52] tries to integrate the behaviour of the real network stack and the

operating system by using virtualized hosts connected through an emulation framework. The virtual hosts are running an L4 microkernel on top of a real-time kernel. To integrate the wireless network behaviour, the hosts are connected by the 802.11b network emulator MobiEmu [55]. The wireless interface driver has been modified to communicate with the emulator instead of the physical interface, but keeping the interface to the applications unaltered. A drawback of the approach is inherited by the use of MobiEmu: The communication is either possible without errors or not at all. MobiEmu does not model any communication errors. JiST/MobNet [53] provides a comprehensible Java framework for the simulation, emulation and real-world testing of wireless ad hoc networks. MobNet is a wireless extension on top of the Java in Simulation Time (JiST) simulator. The drawback of this approach is that most communication software and network protocol stacks are not written in Java and therefore a further transition to a real-world system may be necessary afterwards. The network testbed Emulab [56] provides various experimentation facilities with advanced experiment management controls. For experiments with wired networks, network nodes run standard operating systems (FreeBSD, Linux and Windows XP) and communicate over an emulated network using virtual local area networks (VLANs) and the emulator Dummynet [57]. Emulab has been extended to wireless networks [58] by an IEEE 802.11a/b/g testbed. Several nodes with real wireless interfaces are deployed on the floors of an office building and can be integrated in an Emulab experiment scenario. Besides the lack of mobility support, the Emulab wireless testbed suffers from limited repeatability due to the shared location in an office building with interferences from productive networks. The wireless network emulator QOMET converts the wireless scenario into a time-series of network states [59]. This state description is then delivered to Dummynet [57] in order to emulate the wireless link between end points. As the normal operating system tools cannot change the wireless parameters of the network, QOMET is not suitable for testing software that influences the wireless interface of a node. The newly developed network simulator ns-3 [60] allows the integration of virtualized nodes running native applications and protocol stacks under the Linux operating system. The virtualized nodes in ns-3 are connected through a TUN/TAP device of the Linux kernel and a proxy node to the simulation. However, there is no support to modify device parameters of the simulation directly and dynamically by the virtualized nodes, especially for wireless devices. OppBSD [61] integrates the TCP/IP stack of FreeBSD in the network simulator OMNeT++. The Network Simulation Cradle [62] project provides support for using the real network stacks of Linux, FreeBSD and OpenBSD with the network sim-

ulator ns-2. The integration of a real TCP/IP stack provides results that are closer to a real-world network. Nevertheless, it does not support the testing of native unaltered applications. When injecting real network traffic into a network simulator, there is always the problem that the simulation may not keep pace with the real network. The simulation may be too slow. In order to cope with the problem of a simulator overload during network emulation, reference [63] introduces the concept of synchronized network emulation. They replace real hosts with virtualized hosts using XEN. A central synchronizer component then controls the time flow of the virtual hosts by an adapted scheduler for XEN. It keeps them synchronized with the network simulator OMNeT++ [41]. Synchronized network emulation represents a valuable extension to avoid scalability problems.
Another interesting approach is a wireless emulator using hardware channel simulators [64, 65, 66]. In [64] the unaltered network nodes are packed in radio frequency (RF)-shielded boxes and their radio interfaces are connected to the hardware channel simulator, which then emulates the signal propagation using a field-programmable gate array (FPGA). The channel simulator supports directional antennas and mobility. The system presented in [65] supports 15 nodes operating in a 2.4GHz industrial, scientific and medical (ISM) band. The main advantage of an FPGA-based wireless emulator is the provided repeatability in combination with a real media access control (MAC) layer and a realistic physical layer supporting multipath fading. The main drawbacks are the costs and the limited number of nodes. MeshTest [66] is a laboratory-based multi-hop wireless testbed that can subject real wireless nodes to reproducible mobile scenarios. It uses shielded enclosures and an RF matrix switch to dynamically control the attenuation experienced between pairs of nodes. The testbed is an ideal platform for experimenting with MANET and DTN (delay tolerant networking) implementations, offering convenient experimental control and data management. The Emulated Wireless Ad Hoc Network Testbed (EWANT) [67] is a reduced-scale MANET testbed with emulated RF environment using in-line attenuation and RF multiplexing. Mobility is simulated by discrete switching between different antennas connected to the outputs of the 1:4 RF multiplexers attached to the wireless cards.
VirtualMesh [68, 69, 70] provides instruments to test the real communication software including the network stack inside a controlled environment. The main concept of VirtualMesh is to intercept and redirect real traffic generated by real nodes to a simulation model, which then handles network access and the behaviour of the physical medium. VirtualMesh has been implemented by capturing real traffic through a virtual interface at the mesh nodes. The traffic is then redirected to the network simulator

OMNeT++. The network stack is split into two parts: First, the application, transport and Internet layer are handled by the real/virtualized node. Second, at the MAC layer, the traffic is captured by a virtual network interface and then redirected to the simulation model. The simulation model calculates the network response according to the virtual network topology, the propagation model, the background interferences and the current position of the nodes. Only the MAC layer and the physical medium are simulated. All the other layers remain unchanged and work just as in a real testbed of embedded Linux nodes. VirtualMesh has proven to be scalable, to have minimal influence on throughput and to introduce only negligible delays (less than 0.4 ms per hop). VirtualMesh combines the advantages of real-world tests performed on embedded Linux systems with the flexibility and the controlled environment of a network simulator. The main advantages are: the real communication software is used, the real network stack is tested, the effects of temporary unavailable nodes can be evaluated, background traffic/interferences can be controlled and different mobility tests can be easily performed. The real implementation of the communication software can be tested. Accordingly, the behaviour of the Linux network stack is embedded in a controlled testing environment. There are no irrepressible influences on the experiments such as interferences from neighbouring networks and power lines, steel structures of buildings or changing weather conditions. In addition, the underlying simulated network enables large-scale experiments. It supports changing topologies and different mobility scenarios. This makes automated testing of the real communication software with a high variety of scenarios possible. VirtualMesh consists of an arbitrary number of computers hosting the simulation model and real or virtualized mesh nodes. A dedicated Ethernet network interconnects the nodes and the model. The wireless interfaces of the nodes are replaced by virtual interfaces, which communicate over the service network to the simulation model. Besides real nodes, the architecture supports virtualized hosts. Host virtualization is performed by XEN, but other virtualization techniques could be used too. Host virtualization provides additional scalability of the system. One standard server machine may hold up to ten virtual mesh nodes without any problem.

3 Background on WISEBED

3.1 The WISEBED project

WISEBED (www.wisebed.eu) [2] is a project of the EU FIRE program [16]. Its aim is to provide an infrastructure of interconnected testbeds of large-scale wireless sensor networks for research and experimentation purposes. The WISEBED experimentation infrastructure interconnects different testbeds across Europe and forms a federation of distributed test laboratories, interconnects the wireless sensor network testbeds with the Internet and especially with other testbeds from FIRE in order to provide a virtual laboratory to enable testing and benchmarking in a controlled way. WISEBED allows researchers to use the experimentation facilities remotely, thus reducing the need for a local, private testbed and, more importantly, reducing the cost for research and integrates simulated and physical sensor node support large-scale sensor network experiments. Since WISEBED interconnects heterogeneous and previously incompatible sensor nodes among each other, users are able to set up a testbed with nodes equipped with different sensors, memory sizes and energy supplies. WISEBED provides services for allowing algorithms and applications to be tested in large-scale environments by providing a repository of algorithms, mechanisms and protocols (Wiselib library) that can be directly used in future applications and experiments as reference for benchmarking purposes. Within the WISEBED project a sophisticated testbed management system including federated authentication and authorization using single sign-on, allocation and reservation of testbeds and sensor nodes, deployment of experiment software, and experiment monitoring has been developed. WISEBED introduced the notion of a virtual link, which enables the interconnection of two different sensor nodes in two remote testbeds as if they were close to each other. By means of a virtual link, such remote nodes can communicate directly. Finally, WISEBED has started to integrate mobile nodes into different testbed sites, but full control of mobile sensors is not yet available.

3.2 The WISEBED approach to virtualisation

As a result of observing the complementary strengths and weaknesses of the the physical, simulated and emulated approaches, the WISEBED project has designed an abstract WSN testbed abstraction called *virtual testbeds*. Virtual testbeds (henceforth VTBs) offer an abstraction of a

user-designed, private WSN testbed in which some of the primary testbed elements are physically real, some are simulated and some are emulated. The full list of 'primary elements' to which we refer, each of which can be either physical, simulated or emulated, is as follows:

1. Sensors (e.g. temperature or depth sensors).
2. Sensor *input* (i.e. what the sensors observe, such as the current temperature or water depth).
3. Nodes (the CPU+memory+radio device to which sensors are attached).
4. Connectivity (which is a function of node location and radio characteristics).
5. Power to nodes.
6. Mobility of nodes.

Users design their VTBs in such a way that their mix of the above physical, simulated and emulated elements is appropriate to their goals. They then instantiate their testbed, deploy their software onto it, and observe the outputs and behaviour of their experimental system as if it were running on a single physical testbed. The key benefit of the VTB abstraction is that it allows users to exert fine-grained control over the makeup of their own personal testbed environment.
To date, the project has explored virtualisation of most of the above primary elements, but the least explored (apart from 'Power to nodes' which requires specialised hardware development) is virtual mobility – to date the general issue mobility has only been considered in WiseML and in simulation environments.

3.3 WISEBED's federated environment

The VTB abstraction has been implemented in the federated WISEBED environment. In this implementation, multiple VTBs, e.g. one per user or one per experiment, can be dynamically instantiated, and are simultaneously supported by the WISEBED environment.
A key characteristic of the WISEBED environment is that its underlying physical infrastructure adopts a *federated* approach that incorporates physical equipment from a number of different sites across Europe where

each site communicates with the others via a per-site *portal server*. Federation enhances the flexibility of the VTB abstraction in two key dimensions:

- *Scalability* – this is enhanced by combining physical nodes from different federated sites into a single VTB (of course, when a VTB needs to be even larger than the combined set of available physical nodes, scalability can be further enhanced by adding simulated or emulated nodes).
- *Heterogeneity* – users can explore highly heterogeneous WSN environments by building VTBs that incorporate different kinds of physical nodes taken from a number of diverse sites to obtain their required mix.

3.4 WiseML

VTBs are specified by users in terms of an XML schema called *WiseML* [71] which supports the specification of the following types of information. WiseML is in fact a multi-purpose format used throughout the WISEBED federation for specifying and recording various aspects of experimental configuration and output data. The generation of WiseML specifications of VTBs is typically carried out by a GUI-based tool.
The basic shape of a WiseML specification is as follows:

```
<wiseml version="1.0" xmlns="http://wisebed.eu/ns/wiseml/1.0">
    <setup>
        ...
        <defaults>
        <node>
            <capability> <name>temperature</name>
            </capability>
            ...
        </node>
        ...
        </defaults>

        <node id="urn:wisebed:node:tud:M4A0P2OV">

        </node>
    </setup>
```

```
    <scenario id="...">
        ...
    </scenario>

    <trace id="...">
        <timestamp>2</timestamp>
        <node id="...">
            <data key="temperature">12</data>
        </node>
        ...
    </trace>
</wiseml>
```

As can be seen there are three sections in a WiseML specification which are further described as follows:

– The *setup* section describes the experimental setup and the data to be collected. There are two basic concepts: *attributes* (compulsory features of elements, such as name) and *capabilities* (optional user-defined characteristics such as pressure gradient).
The setup section has three sub-sections as follows:

- *setup description* – geographical placement of nodes, starting time information, mobility interpolation
- *node-related information* – attributes such as position, plus capabilities and default values
- *link-related information* – attributes such as default RSSI, is-virtual, etc., plus capabilities and default values, such as LQI.

The general approach is to define defaults for nodes and links, and then to override the defaults for (presumably small numbers of) non-default nodes/links.

– The *scenario* section describes changes to be applied to an experiment such as simulated node failure (enableNode, disableNode tags tags), simulated interference at the virtual link level (enableLink, disableLink) or simulated sensor noise level or failure (achieved by making modifications of default sensor values). Here is an example:

```
<scenario id="...">
  <timestamp>0</timestamp>
  <enableNode id="..." />
  <disableNode id="..." />
  <enableLink source="..." target="..." />
  <disableLink source="..." target="..." />
  <node id="...">
    <position>
      <x>0</x>
      <y>1</y>
      <z>2</z>
      <phi>0</phi>
      <theta>1</theta>
    </position>
    <data key="lqi">50</data>
  </node>
</scenario>
```

– Finally, the *dynamic information* section captures actual collected data such as sensor readings, network layer information (links and rssi values) or mobility information. For time description, the dynamic information section makes use of a timestamp tag, marking moments in time relative to a global start time specified in the setup section. Intervals can also be emulated, and if data has a continuous characteristic (such as a sine wave) the user can define an interpolation type in the setup section. Finally, for a mobility description we can specify positions at successive points with the user specifying the interpolation type between points. Here is an example:

```
<trace id="...">
    <timestamp>2</timestamp>

    <node id="...">
        <position>
            <x>2</ x><y>4</ y><z>6</ z>
        </position>
        <data key="ranger">12</data>
        <data key="...">...</data>
    </node>

    <link source="..." target="...">
```

```
        <rssi>12</rssi>
        <data key="lqi">100</data>
    </link>
    <timestamp>4</timestamp>
</trace>
```

3.5 Virtual links

So-called *virtual links* are an important element in the underpinning of WISEBED's VTB concept. They are used to define potential connectivity between pairs of nodes that are physically distant but virtually close. Specifying a virtual link in WiseML represents the possibility of 1-hop unidirectional communication between two nodes in the VTB[1]. The WiseML specification of a virtual link includes the UIDs of the nodes that will participate in the link, and the link's capabilities in terms of its LQI, packet error rate, etc.

Each physical testbed provider in WISEBED uses its own private mechanisms (e.g. based on backbone management networks) to implement the required virtual link machinery, and Internet tunnels between testbeds are used where the linkage is across sites. A detailed discussion of the implementation of virtual links is available in the literature [72].

Within the sensor nodes themselves, the WISEBED Software Development Kit's *radio stacking framework* is used to engineer the end points of virtual links. This framework is used to deploy 'pseudo' radio drivers that appear to software on the node as 'real' radio drivers; however they transparently divert (selected) outgoing packets to the virtual link machinery, and insert incoming packets arriving from the virtual link machinery.

[1] So, *two* virtual links are needed to represent mutual connectivity between two nodes; this allows us straightforwardly to model situations in which a node A can send to a node B, but B cannot send to A.

4 Virtual mobility

This section is structured as follows: first Section 4.1 defines virtual mobility. Section 4.2 then examines the support already offered by WiseML for the specification of virtual mobility in a VTB environment. Finally, Section 4.3 then raises a number of issues that should be addressed in an infrastructure to support virtual mobility.

4.1 Definition

We define *virtual mobility* in the context of a WISEBED VTB as follows: *a VTB features virtual mobility if, during the course of a users's experiment on the VTB, the positions of nodes (be they physical, simulated, or emulated), as initially specified in the VTB description (i.e. using the WiseML setup section), change*. Note that this definition is in the context of VTB specifications in which node positions are specified in virtual terms, relative to the 3-D space defined in the setup section of the VTB's WiseML specification.

4.2 Existing virtual mobility support in WiseML

From Section 3.4, we can see that there is already a degree of support for mobility in WiseML. To summarise, the mobility related concepts in WiseML are the following:

- In the setup section there is an interpolation tag that specifies which type of interpolation (e.g. cubic) should be used between positions at adjacent timestamps.
- In the scenario section it is possible to specify a new location for a node at a particular timestamp.
- In the dynamic information section, information on the the post-hoc mobility of nodes can be recorded just like sensor readings: at certain moments in time the position of the node is given and in between an interpolated position can be used.

It therefore seems to be possible to specify virtual mobility for a node by providing a list of scenarios with increasing timestamps and for each of

which the coordinates of mobile nodes is made to change. This could, for example, be derived from a standard mobility model.
The presence of links in the scenario section, however, would seem to considerably complicate the picture because these would have to be changed in very close coordination with node positions. For virtual mobility a more likely possibility is to employ only node-related specification and to leave reachability information to be deduced by a 'virtual mobility interpreter' as discussed in Section 4.3.

4.3 Virtual mobility support in VTBs

The basic concept is this: Where we have a VTB featuring virtual mobility, all communication between virtually mobile nodes and other nodes (whether virtually mobile or fixed) needs to be channeled through a Virtual Mobility Interpreter (VMI). The VMI maintains a list of the current (virtual) locations of all the nodes involved, and communication is channeled to the VMI using components from the standard virtual link machinery (i.e. via the SDK's virtual radio framework and the local physical testbed's portal server). When the VMI receives a radio packet it pushes it through a radio channel model, and thereby defines a virtual area within which the packet can be heard. The VMI then pushes the radio packet to all nodes within this virtual area. Again, this pushing is done through the standard virtual link machinery.
Drilling down into this basic design it is clear that there are many issues arising:

1. *Specification*: how do we *specify* virtual mobility? The most likely answer is to use WiseML which, as discussed above, seems to have most or all of the required descriptive capability. But there are other options such as the 'BonnMotion format'[2]. Another option is based on representing time as intervals[3]. A further option would be to employ vector-based descriptions of mobility – these would not need to rely on interpolation between successive discrete instances of time.

[2]The BonnMotion format [51] specifies a plain text file, where every line describes the motion of one host. A line consists of one or more (t, x, y) triplets of real numbers, like: t1 x1 y1 t2 x2 y2 t3 x3 y3 t4 x4 y4 ... the meaning is that the given node gets to (xk,yk) at tk; there's no separate notation for waiting, instead x and y are simply repeated; see http://www.cs.uni-bonn.de/IV/BonnMotion/. The BonnMotion format is actually conceptually identical to the WiseML approach.

[3]The WISEBED project considered this option but made a decision to support the expression of time only as a series of discrete events.

And a more left-field option would be to adopt the virtual time concept proposed by RWTH [63]. At the present time, however, the native WiseML approach seems to be adequate.

2. *Centralised vs node-autonomous specification*: As well as the obvious centralised approach whereby a consolidated mobility specification is fed to the VMI, another possibility that might well be useful is to have individual nodes autonomously determine their own virtual mobility – e.g. according to private scripts owned by each node (or simply recording their position in the case of a physically moving physical node). In this case, the individual nodes would need to continually update the VMI with their current position. The option of autonomous nodes would also facilitate the modelling of adaptive mobility – or at least it would delegate the problem of how to adapt mobility to individual nodes.

3. *Static vs dynamic specification*: Another design dimension is whether a mobility specification is fixed and deterministic prior to experiment runtime, or if it may change during runtime. It is not clear at the moment if the latter case is either feasible or useful.

4. *Distributed VMIs*: To scale up virtual mobility the obvious approach is to distribute VMIs – e.g. have one at each physical testbed site (they could be time synchronised using NTP or PTP). This raises further questions of how to optimally place VMIs within the physical infrastructure underlying the VTB to maximise scalability. The communication between VMIs may be a problem – especially if several are involved – such that it is not immediately clear what would be the sensitivity of an experiment outcome to the communication latency between multiple VMIs (and, indeed, between individual nodes and the VMI or VMIs).

5. *VMI realisation*: what technology do we use to realise the VMI? One possible approach would be to use specialised hardware based engines (e.g. as discussed in Section 2.3) to support high resolution and high throughput of modelled radio channels. However, this approach is likely to be expensive and it had also been shown not to scale very well. Another approach, which we currently favour, is to use a simulator engine such as OMNet++ as the basis of the VMI. Experience at UBERN has indicated that real-time throughput can be achieved with this simulator at a reasonable scale [68, 69][4].

[4]The choice of a realisation for the VMI also impacts to some extent the above-

6. *Scope of a VMI*: When we have virtual mobility in a VTB do we have to force *all* communication through a VMI - or can we leave some nodes in the VTB to communicate using their default mechanisms (such as physical radios, virtual links) outside of the VMI context – e.g. if some nodes are out of communication range of anything that might move. But given that nodes are moving how do we know which areas of the VTB are 'safe' from virtual mobility? Maybe we can deduce this in static mobility scenarios, but presumably not when nodes can autonomously virtually move.

7. *Co-existence of virtual and physical mobility*: A special case seems to emerge when we consider integrating *physical* mobility into a VTB that employs virtual mobility (this only arises, of course, in the case of physical nodes). Here, we want the physical mobility to be consistent with virtual mobility elsewhere in the VTB so that, for example, if we have two physical nodes within range of each other, and one moves physically and the other virtually, but in the same direction and at the same speed, we expect the (virtual) distance and therefore (virtual) connectivity between the two to stay the same.

discussed choice of a virtual mobility specification format because a mapping will be needed from the specification format to the underlying VMI realisation. Our current understanding, however, is that a mapping from WiseML to the native OMNet++ format is not a problematic issue.

5 Outline design

Having considered the issues above we now present our initial design of an infrastructure to support virtual mobility. This is presented as a 'straw man' for the sake of concreteness in the hope that it will stimulate further thought and lead to a more refined design.
We proceed by first considering a 'simple case' design and then, in the following section, considering complicating factors and extensions.

5.1 Consideration of the simple case

The main players in our design are as follows (see Figure 1):

- The set of nodes that comprise the VTB.
- The virtual space in which nodes live and in which virtual mobility takes place.
- A set of VMIs.
- Processes called Portal Server Agents (PSAs) running on the portal servers of each physical testbed site (simulator servers that host simulated nodes will also have an associated PSA).

Each node has a dynamically-varying coordinate attribute (x, y, z) which places it somewhere in virtual space; and virtual space is statically divided up between VMIs[5]. The individual 'sub-spaces' of each VMI slightly overlap so that each VMI can 'see' nodes just over its border. The motivation for this is to handle cases where packets travel over borders, and the VMIs on each side of the border need to be aware of all in-range nodes so that they can factor them into their modeling of radio interference. We refer

[5]This raises issues of how to determine the optimal number of VMIs per virtual testbed space, and how best to divide virtual space among these VMIs. This is for further study, but heuristics such as the following seem plausible: (i) VMs should be allocated to areas whose nodes tend to be supported by a common underlying physical testbed; and (ii) VMs should allocated to areas between which there is not likely to be much virtual mobility. The question of whether we want to constrain how virtual space is divided up is also for further study: for example, do we insist that each VMI takes a 'slice' of virtual space such that each has at most two neighbours, or do we allow unconstrained partitions (maybe even such that a VMI can be responsible for several irregularly-shaped non-contiguous areas)? Dynamic partitioning for load-balancing purposes is also worth considering (e.g., to cover extreme situations such as all nodes moving to a single corner of virtual space).

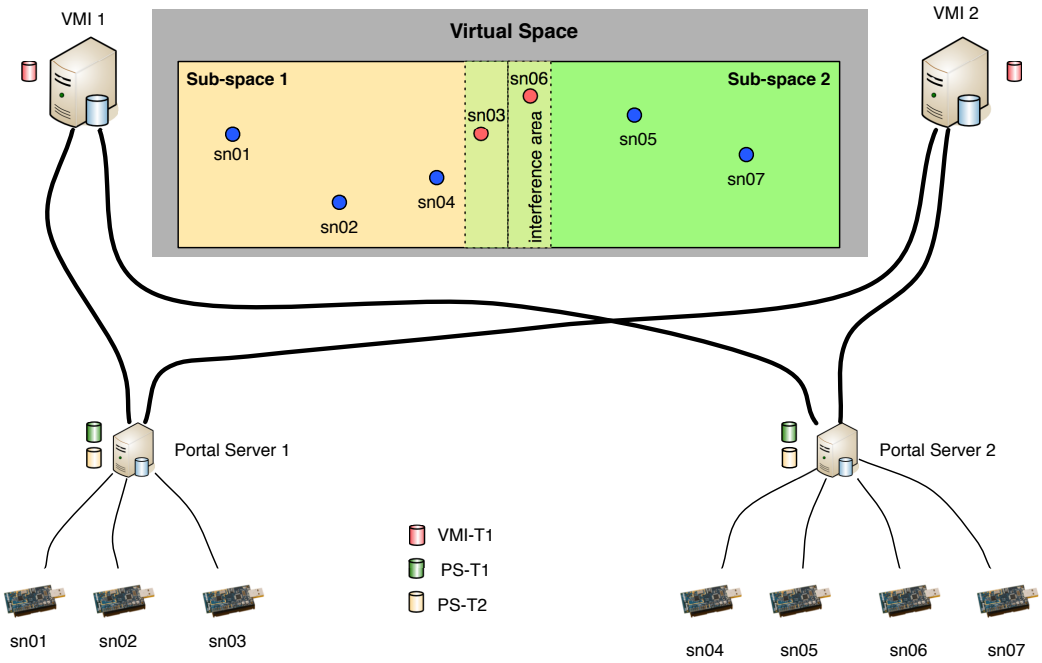


Figure 1: General architecture showing the main components with two PSAs and two VMIs

to these overlaps between VMI sub-spaces as *interference areas* and the non-overlapped areas as *core areas*.
So, at any given time, each virtually-mobile node is associated with at least one VMI: we will say that nodes within a VMI's core area are *supervised by* the VMI and that nodes within the VMI's interference area are *secondarily-supervised* by the VMI. We will also say that each node is *managed by* some specific PSA at the physical, implementation, level. The 'supervised by' and 'secondarily-supervised' mappings will change as the node virtually moves; but the 'managed by' mapping will be invariant as it is part of the physical infrastructure (barring physical mobility).
The design then incorporates the following structures and functions:

- Each PSA has a table PSA-T1 that maps each of the nodes it is managing to the node's current coordinate attribute value.
- Each PSA has a table PSA-T2 that maps each VMI in the system to the bounding coordinates of the VMI's associated sub-space.
- Each PSA has a 'rough' model of radio propagation for the radio type supported by the sending node – this is likely to be as simple as the radius of a circle drawn from the sender's coordinates[6].
- Each VMI has a table VMI-T1 that maps each of the nodes it is supervising to the node's current coordinate attribute value and the PSA that manages it.
- Each VMI has a detailed model of radio propagation that also handles radio interference issues.

We are now in a position to describe how virtual mobility occurs. We assume here the case where virtual mobility is deterministically defined by a single mobility script which is interpreted tick-by-tick[7] by a so-called *script interpreter*[8].

1. The script interpreter performs initialisation: it divides virtual space into a suitable number of sub-spaces, instantiates a VMI for each of these, and passes each VMI the information it needs to populate

[6]The 'rough' model should tend overestimate the propagation range as overestimates can be corrected later by the detailed modelling performed by the VMIs.

[7]These ticks will occur in real-time, but must be of a sufficiently coarse grain that all the required inter-tick actions have time to occur.

[8]This may run in any convenient location in the distributed system – e.g., on the portal server that manages most of the nodes in use in the virtual testbed (if there is such).

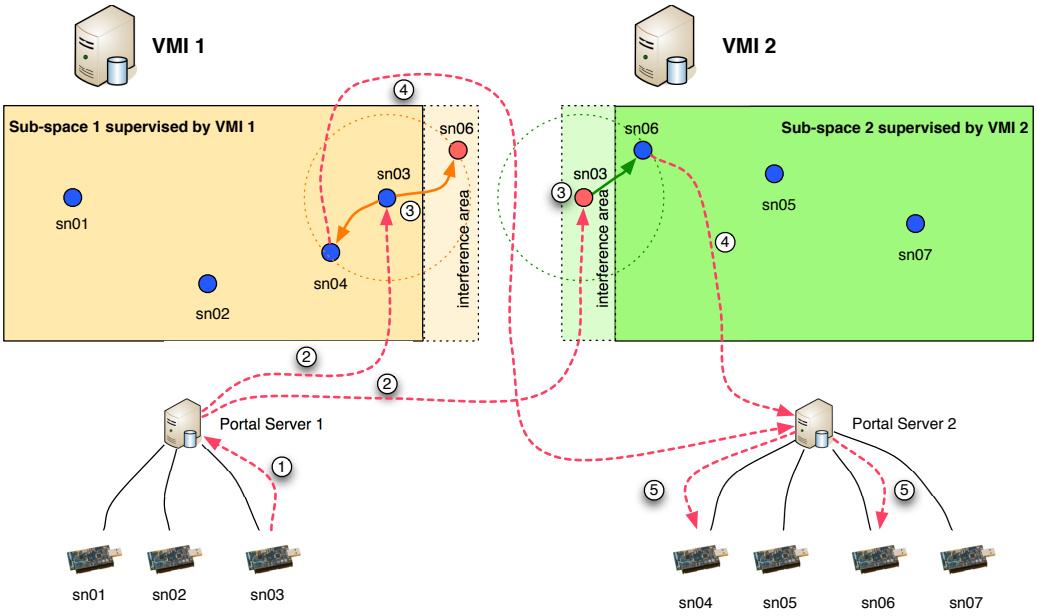


Figure 2: Example packet flow: sn03 sends to sn04 and sn06

its VMI-T1 table (i.e. the list of nodes it is initially supervising and secondarily-supervising, their initial coordinate attribute values, and their managing PSA). It also similarly initialises all the involved PSAs.

2. On each tick the script interpreter sends an update of this information to each VMI. As well as updating node coordinates, this will cause deletions of nodes from a VMI when nodes moves out of the VMI's sub-scope, and additions when nodes move into a sub-scope.

3. On each tick, each VMI sends to each of its associated PSAs the required information to update the PSA's PSA-T1 table with new co-ordinate values for its managed nodes.

Having sketched how virtual mobility works, we are now in a position to describe what happens when a virtually-mobile node sends a packet (see Figure 2):

1. When a node sends a packet, the packet is intercepted by the node's radio stacking framework which forwards it over the management backbone to the node's PSA.

2. The PSA consults PSA-T1 to determine the sending node's current coordinate attribute value.

3. The PSA runs its 'rough' model of radio propagation and from this it derives the bounding coordinates of an area in virtual space likely to be reachable by the packet.

4. The PSA consults PSA-T2 to derive a list of all the VMIs that own points in the reachable area – i.e., all VMIs that potentially supervise or secondarily-supervise nodes that might be reachable by the packet.

5. The PSA forwards the packet to all those VMIs.

6. The VMIs push the packet through their radio models; as a result, they determine the actual, accurate, spatial extent to which the packet should propagate.

7. The VMIs consult their tables VMI-T1 to determine the set of supervised (but *not* secondarily-supervised) nodes that currently lie within this extent and thus the PSAs that are managing each of these nodes.

8. The VMIs forwards the packet to the respective PSAs.

9. The PSAs forward the packet to the respective receiver nodes.

Finally, we exemplify this protocol with specific reference to Figure 2. Note that nodes sn03 and sn06 are in the interference areas of their respective sub-spaces: sn03 is supervised by VMI 1 and secondarily-supervised by VMI 2; whereas sn06 is supervised by VMI 2 and secondarily-supervised by VMI 1 (the step numbers in the below correspond to those above).

1. Node sn03 sends a packet; the packet is intercepted and transferred to PSA 1 (Figure 2, step 1).

2. PSA 1 consults PST1 to determine sn03's location.

3. PSA 1 runs its 'rough' propagation model and determines the bounding coordinates of the area in virtual space likely to be reachable by the packet.

4. PSA 1 determines that the packet should be forwarded to both VMI 1 and VMI 2 (Figure 2, step 2).

5. PSA 1 forwards the packet to VMI 1 and VMI 2.

6. The propagation of the packet is modelled by both VMIs (Figure 2, step 3) using a suitable propagation model and taking interference into account.

7. As a result, VMI 1 determines that the packet should be received by sn04. VMI 1 also determines that the packet should be received by sn06, but as it only secondarily-supervises this node it should not forward the packet in this case. In parallel, VMI 2 determines that the packet should be received by sn06.

8. VMI 1 forwards the packet to PSA 2 (destination: sn04); and VMI 2 also forwards the packet to PSA 2 (destination: sn06) (Figure 2, step 4).

9. PSA 2 forwards the packets respectively to sn04 and sn06 (Figure 2, step 5).

5.2 Consideration of more complex cases

The above design fails to consider a number of complicating factors such as:

- Time and race conditions
- Automomously mobile nodes
- Accommodating physical mobility alongside virtual mobility
- Optimising by avoiding use of the VMI where possible.

We now consider such cases.

5.2.1 Time and race conditions

Race conditions can occur in the above design. For example, it may be that a VMI 'loses' a node (removed by the script interpreter's update) just before a PSA sends it a message for that node. More generally, race conditions occur on each 'tick' when the script interpreter initiates a cascading update to the VMIs (node position updates, nodes arrive at/ leave a VMI), and then from the VMIs to the PSAs (node position updates).
Essentially, there are two independently time-triggered processes going on that may interfere with each other:

- The tick-based updates discussed above, and
- Nodes sening packets.

We need to identify in detail the pathological effects that might arise from such factors and design consistency mechanisms that address the problems without significantly slowing the packet propagation process (e.g. can we simply update the PSAs first and then if a VMI gets a message for node it doesn't currently supervise, it can wait until that node arrives?).
In case it is not possible to completely solve these issues, we should log packets lost due to infrastructure errors and clearly distinguish these from application errors when reporting to the user.

5.2.2 Automomously mobile nodes

In the above outline design, we assumed that virtual mobility was driven by a central script interpreter. We now consider the case in which each node autonomously determines its mobility, meaning that nodes must proactively inform PSAs/ VMIs of their virtual positions.
Our initial view is that this can be achieved relatively straightforwardly: rather than position updates coming from the script interpreter, they now come from each individual node. The main impact that this would seem to have would be on the consistency mechanisms discussed in Section 5.2.1. It may be that different mechanisms are needed for the script interpreter and autonomous node cases.

5.2.3 Accommodating physical mobility

Physical mobility would probably be implemented by attaching a sensor mote to some sort of robotic device that could move around a floor, and could also host an out-of-band wireless network that could communicate with the management backbone network used for the rest of the system.
We would then apply a design similar to that discussed in Section 5.2.2 in which autonomous nodes (including the physically mobile ones) repeatedly report their positions to the PSAs/ VMIs. In this case, the reporting process would have to apply a mapping from the floor coordinates to the virtual space's coordinate system. This mapping, which would be established statically at the start of the experiment, would correspond to exactly where in the virtual space the floor space (and thus the the limits of the node's mobility) was supposed to be located.

When a physically mobile node sent a message, it would be handled by the usual PSA/VMI machinery and would therefore have exactly the same semantics as the virtually mobile case. The same would apply for the case of a physically mobile node receiving packets from virtually mobile nodes. The same would even apply for one physically mobile node communicating with another physically mobile node.

5.2.4 Optimising by avoiding use of the VMI where possible

It seems that a situation may often arise in which virtual mobility is confined to a small area within virtual space, In such a situation, it would seem unnecessary to direct *all* traffic in the virtual testbed through VMIs – it would seem in fact to be much preferable to avoid using VMIs in 'static' areas and instead use the default communication mechanism such as the physical radio. The problem, however, is that to the extent that virtual mobility is unpredictable, it is impossible to be sure that on the next tick a mobile node may not end up at any arbitrary point in virtual space – so it would seem that VMI-mediated communication must be used universally to prepare for such an eventuality.

5.2.5 Integration of energy measurements and models

Wireless sensor network deployments are usually energy-constrained. A testbed should therefore support energy measurements and evaluations. One option is to power the nodes by special energy measurement circuits such as the Sensor Node Management Devices (SNMD) [73]. These boards have been developed as a cost- effective alternative to using high-frequency multimeters or oscilloscopes for side- effect free high-resolutive energy-measurement of wireless sensor nodes. They provide continous current and voltage measurements with resolutions up to 20'000Hz. Although, they are a convenient measurement tool for lab environments, they may be too expensive for large deployments. Therefore, an alternative solution is the integration of a newly developed fine-grained software-based energy estimation model that is currently evaluated and verified with the nodes powered with SNMD devices at the University of Bern.

6 Conclusions and next steps

In this technical report, we presented an architecture for adding virtual mobility to a federated testbed for wireless sensor networks. Our concept adds virtual mobility by embedding physical, emulated and simulated nodes into the same virtual space. The traffic of physical is therefore intercepted and redirected to a simulation model responsible for a virtual sub-space. Our concept provides unlimited mobility of all nodes within the virtual space.

The next steps for the implementation of virtual mobility are defining Master and Bachelor projects and running a first initial experiment. Topics of Master and Bachelor thesis are different prototype implementations, extension of the virtual link concept and VirtualMesh, virtual space partitioning, integration into WISEBED and runtime optimisation.

A first initial experiment shall proof the feasibility of virtual mobility among the two wireless sensor network testbeds at the University of Bern and the University of Lancaster. The scenario contains three nodes in a line in Bern and three nodes in a line in Lancaster. VTB puts them all in a single line. A seventh node then virtually moves along the line receiving messages sent by the six fixed nodes. Our main interest is to evaluate the amount of discontinuity that we get when the seventh node passes between the 3rd and 4th nodes, i.e. crosses the border between the virtual sub-spaces.

7 Acknowledgements

The authors would like to acknowledge the support of the EC in funding this research under the Framework Programme 7. We would like specifically to thank COST Action IC 0906 for providing funding under the STSM scheme.

References

[1] D. Dudek, C. Haas, A. Kuntz, M. Zitterbart, D. Krüger, P. Rothenpieler, D. Pfisterer, and S. Fischer, "A wireless sensor network for border surveillance (demo)," in *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, (Berkeley, CA), ACM, Nov. 2009.

[2] I. Chatzigiannakis, C. Koninis, G. Mylonas, S. Fischer, and D. Pfisterer, "WISEBED: an open large-scale wireless sensor network testbed," in *Proceedings of the 1st International Conference on Sensor Networks Applications, Experimentation and Logistics*, Sept. 2009.

[3] P. D. 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.

[4] G. Werner-Allen, P. Swieskowski, and M. Welsh, "Motelab: A wireless sensor network testbed," in *Fourth International Conference on Information Processing in Sensor Networks (IPSN)*, 2005. Special Track on Platform Tools and Design Methods for Network Embedded Sensors (SPOTS), IEEE, Piscataway, NJ.

[5] V. Handziski, A. Kopke, A. Willig, and A. Wolisz, "Twist: A scalable and reconfigurable wireless sensor network testbed for indoor deployments," Tech. Rep. TKN-05-008, November 2005.

[6] "Tutornet: A tiered wireless sensor network testbed." http://enl.usc.edu/projects/tutornet/.

[7] B. N. C. et al., "Mirage: A microeconomic resource allocation system for sensornet testbeds," in *2nd IEEE Workshop on Embedded Networked Sensors*, 2005.

[8] E. E. et al., "Kansei: A testbed for sensing at scale," in *5th International conference on Information processing in sensor networks (IPSN)*, 2006.

[9] 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)*, June 2007.

[10] P. Levis, N. Lee, M. Welsh, and D. Culler, "Tossim: accurate and scalable simulation of entire tinyos applications," in *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, (New York, NY, USA), pp. 126–137, ACM Press, 2003.

[11] University of Southern California, Information Sciences Institute (ISI), "Ns-2: Network simulator-2." http://www.isi.edu/nsnam/ns/.

[12] L. Girod, T. Stathopoulos, N. Ramanathan, J. Elson, D. Estrin, E. Osterweil, and T. Schoellhammer, "A system for simulation, emulation, and deployment of heterogeneous sensor networks," in *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, (New York, NY, USA), pp. 201–213, ACM Press, 2004.

[13] H. Wu, Q. Luo, P. Zheng, B. He, and L. M. Ni, "Accurate emulation of wireless sensor networks," in *Proceedings of Network and Parallel Computing (NPC'2004)*, p. 576583, Oct. 2004.

[14] L. Peterson, T. Anderson, D. Culler, and T. Roscoe, "A blueprint for introducing disruptive technology into the internet," *SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 1, pp. 59–64, 2003.

[15] K. Jang, S. Kim, G. M. Voelker, and S. Moon, "Implementation and evaluation of a mobile wireless planetlab node," in *SOSP Workshop on Real Overlays and Distributed Systems (ROADS '09)*, (Big Sky, Montana, US), October 11-14 2009.

[16] European Commission, "FIRE - Future Internet Research and Experimentation, ICT Research in FP7." http://cordis.europa.eu/fp7/ict/fire/.

[17] "OneLab - Developing testbeds for the Future Internet." http://www.onelab.eu/.

[18] "Panlab - Pan European Laboratory Infrastructure Implementation." http://www.panlab.net/.

[19] "FEDERICA - Federated E-infrastructure Dedicated to European Researchers Innovating in Computing network Architectures." http://www.fp7-federica.eu/.

[20] D. Raychaudhuri, I. Seskar, M. Ott, S. Ganu, K. Ramachandran, H. Kremo, R. Siracusa, H. Liu, and M. Singh, "Overview of the ORBIT radio grid testbed for evaluation of next-generation wireless network

protocols," in *IEEE Wireless Communications and Networking Conference (WCNC 2005)*, vol. 3, pp. 1664 – 1669, March 2005.

[21] B. Blywis, M. Günes, F. Juraschek, and J. Schiller, "Trends, advances, and challenges in testbed-based wireless mesh network research," *ACM/Springer Mobile Networks and Applications (MONET)*, February 2010. Special Issue on Advances In Wireless Testbeds and Research Infrastructures.

[22] J. Bicket, D. Aguayo, S. Biswas, and R. Morris, "Architecture and evaluation of an unplanned 802.11b mesh network," in *MobiCom '05: Proceedings of the 11th annual international conference on Mobile computing and networking*, (New York, NY, USA), pp. 31–42, ACM, 2005.

[23] K. Chebrolu, B. Raman, and S. Sen, "Long-distance 802.11b links: performance measurements and experience," in *MobiCom '06: Proceedings of the 12th annual international conference on Mobile computing and networking*, (New York, NY, USA), pp. 74–85, ACM, 2006.

[24] V. Angelakis, M. Genetzakis, N. Kossifidis, K. Mathioudakis, M. Ntelakis, S. Papadakis, N. Petroulakis, and V. A. Siris, "Heraklion mesh: an experimental metropolitan multi-radio mesh network," in *WinTECH '07: Proceedings of the second ACM international workshop on Wireless network testbeds, experimental evaluation and characterization*, (New York, NY, USA), pp. 93–94, ACM, 2007.

[25] R. Patra, S. Nedevschi, S. Surana, A. Sheth, L. Subramanian, and E. Brewer, "Wildnet: Design and implementation of high performance wifi based long distance networks," in *in 4th USENIX Symposium on Networked Systems Design and Implementation*, pp. 87–100, 2007.

[26] D. Wu, D. Gupta, and P. Mohapatra, "Quail ridge wireless mesh network: Experiences, challenges and findings," pp. 1 –6, may. 2007.

[27] H. Soroush, N. Banerjee, A. Balasubramanian, M. D. Corner, B. N. Levine, and B. Lynn, "Dome: a diverse outdoor mobile testbed," in *HotPlanet '09: Proceedings of the 1st ACM International Workshop on Hot Topics of Planet-Scale Mobility Measurements*, (New York, NY, USA), pp. 1–6, ACM, 2009.

[28] M. Hempel, H. Sharif, T. Zhou, and P. Mahasukhon, "A wireless test bed for mobile 802.11 and beyond," in *IWCMC '06: Proceedings of the 2006 international conference on Wireless communications and mobile computing*, (New York, NY, USA), pp. 1003–1008, ACM, 2006.

[29] H. Lundgren, D. Lundberg, J. Nielsen, E. Nordström, and C. Tschudin, "A large-scale testbed for reproducible ad hoc protocol evaluations," in *Proc. IEEE Wireless Communications and Networking Conference 2002 (WCNC'02*, 2002.

[30] N. H. Vaidya, J. Bernhard, V. V. Veeravalli, P. R. Kumar, and R. K. Iyer, "Illinois wireless wind tunnel: a testbed for experimental evaluation of wireless networks," in *E-WIND '05: Proceedings of the 2005 ACM SIGCOMM workshop on Experimental approaches to wireless network design and analysis*, (New York, NY, USA), pp. 64–69, ACM, 2005.

[31] P. Hurni, T. Staub, G. Wagenknecht, M. Anwander, and T. Braun, "A secure remote authentication, operation and management infrastructure for distributed wireless sensor network testbeds," in *First Workshop on Global Sensor Networks (GSN '09) co-located with KiVS'09, Kassel, Germany*, pp. 1–6, Electronic Communications of the EASST, March 2009.

[32] CONET project, "Testbed Directory." http://www.cooperating-objects.eu/testbed-simulation/testbed-federation/testbed-directory, 2010.

[33] R. Murty, G. Mainland, I. Rose, A. Chowdhury, A. Gosain, J. Bers, and M. Welsh, "Citysense: An urban-scale wireless sensor network and testbed," *Technologies for Homeland Security, 2008 IEEE Conference on*, pp. 583–588, May 2008.

[34] Wiki for the Kansei GENIE project, http://sites.google.com/site/siefastgeni/.

[35] Global Environment for Network Innovations, "GENI." http://www.geni.net/, 2010.

[36] Very large scale open wireless sensor network testbed, http://www.senslab.info.

[37] SENSEI project website, http://ict-sensei.org/.

[38] D. J. et al., "Mobile emulab: A robotic wireless and sensor network," in *25th IEEE Conference on Computer Communications (INFOCOM)*, 2006.

[39] P. De, A. Raniwala, R. Krishnan, K. Tatavarthi, J. Modi, N. A. Syed, S. Sharma, and T.-c. Chiueh, "Mint-m: an autonomous mobile wireless experimentation platform," in *MobiSys '06: Proceedings of the 4th international conference on Mobile systems, applications and services*, (New York, NY, USA), pp. 124–137, ACM, 2006.

[40] D. Jayasingha, N. Jayawardhane, P. Karunanayake, G. Karunarathne, and D. Dias, "Wireless sensor network testbed for mobile data communication," pp. 97 –103, dec. 2008.

[41] A. Varga, "The omnet++ discrete event simulation system," in *European Simulation Multiconference (ESM'2001)*, (Prague, Czech Republic), June 6-9 2001.

[42] A. Boulis, "Castalia: revealing pitfalls in designing distributed algorithms in wsn," in *SenSys '07: Proceedings of the 5th international conference on Embedded networked sensor systems*, (New York, NY, USA), pp. 407–408, ACM, 2007.

[43] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, "Cross-level sensor network simulation with cooja," pp. 641 –648, nov. 2006.

[44] J. Eriksson, F. Österlind, N. Finne, N. Tsiftes, A. Dunkels, T. Voigt, R. Sauter, and P. J. Marrón, "Cooja/mspsim: interoperability testing for wireless sensor networks," in *Simutools '09: Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, (ICST, Brussels, Belgium, Belgium), pp. 1–7, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2009.

[45] R. de Paz Alberola and D. Pesch, "Avroraz: extending avrora with an ieee 802.15.4 compliant radio chip model," in *PM2HW2N '08: Proceedings of the 3nd ACM workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks*, (New York, NY, USA), pp. 43–50, ACM, 2008.

[46] W. Li, X. Zhang, W. Tan, and X. Zhou, "H-tossim: Extending tossim with physical nodes," *Wireless Sensor Networks*, vol. 1, pp. 324–333, Nov 2009.

[47] F. Österlind, A. Dunkels, T. Voigt, N. Tsiftes, J. Eriksson, and N. Finne, "Sensornet checkpointing: Enabling repeatability in testbeds and realism in simulations," in *EWSN '09: Proceedings of the 6th Euro-*

pean Conference on Wireless Sensor Networks, (Berlin, Heidelberg), pp. 343–357, Springer-Verlag, 2009.

[48] S. Park, A. Savvides, and M. B. Srivastava, "Sensorsim: a simulation framework for sensor networks," in *MSWIM '00: Proceedings of the 3rd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems*, (New York, NY, USA), pp. 104–111, ACM, 2000.

[49] L. Girod, N. Ramanathan, J. Elson, T. Stathopoulos, M. Lukac, and D. Estrin, "Emstar: A software environment for developing and deploying heterogeneous sensor-actuator networks," *ACM Trans. Sen. Netw.*, vol. 3, no. 3, p. 13, 2007.

[50] Y. Wen, W. Zhang, R. Wolski, and N. Chohan, "Simulation-based augmented reality for sensor network development," in *SenSys '07: Proceedings of the 5th international conference on Embedded networked sensor systems*, pp. 275–288, ACM, 2007.

[51] N. Aschenbruck, R. Ernst, E. Gerhards-Padilla, and M. Schwamborn, "Bonnmotion: a mobility scenario generation and analysis tool," in *SIMUTools '10: Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques*, (ICST, Brussels, Belgium, Belgium), pp. 1–10, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2010.

[52] M. Engel, M. Smith, S. Hanemann, and B. Freisleben, "Wireless ad-hoc network emulation using microkernel-based virtual linux systems," in *5th EUROSIM Congress on Modeling and Simulation*, (Cite Descartes, Marne la Vallee, France), pp. 198–203, September 6-10 2004.

[53] T. Krop, M. Bredel, M. Hollick, and R. Steinmetz, "Jist/mobnet: combined simulation, emulation, and real-world testbed for ad hoc networks," in *WinTECH '07*, (New York, NY, USA), pp. 27–34, ACM, 2007.

[54] A. Zimmermann, M. Gunes, M. Wenig, U. Meis, and J. Ritzerfeld, "How to study wireless mesh networks: A hybrid testbed approach," *Advanced Information Networking and Applications, 2007 (AINA '07)*, pp. 853–860, May 2007.

[55] Y. Zhang and W. Li, "An integrated environment for testing mobile ad-hoc networks," in *3rd ACM international symposium on Mobile ad*

hoc networking & computing (MobiHoc '02), (New York, NY, USA), pp. 104–111, ACM, 2002.

[56] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar, "An Integrated Experimental Environment for Distributed Systems and Networks," in *Fifth Symposium on Operating Systems Design and Implementation*, (Boston, MA, USA), pp. 255–270, USENIX Association, December 2002.

[57] L. Rizzo, "Dummynet: a simple approach to the evaluation of network protocols," *SIGCOMM Computer Communication Review*, vol. 27, no. 1, pp. 31–41, 1997.

[58] B. White, J. Lepreau, and S. Guruprasad, "Lowering the barrier to wireless and mobile experimentation," in *First Workshop on Hot Topics in Networks (HotNets-I)*, (Princeton, New Jersey, USA), October 28-29 2002.

[59] R. Beuran, L. T. Nguyen, K. T. Latt, J. Nakata, and Y. Shinoda, "Qomet: A versatile wlan emulator," *Advanced Information Networking and Applications, International Conference on*, vol. 0, pp. 348–353, 2007.

[60] M. Lacage, M. Weigle, C. Dowell, G. Carneiro, G. Riley, T. Henderson, and J. Pelkey, "The Network Simulator ns-3." http://www.nsnam.org/, 2009.

[61] R. Bless and M. Doll, "Integration of the freebsd tcp/ip-stack into the discrete event simulator omnet++," in *36th conference on Winter simulation (WSC '04)*, pp. 1556–1561, 2004.

[62] S. Jansen and A. McGregor, "Performance, validation and testing with the network simulation cradle," in *14th IEEE International Symposium on Modeling, Analysis, and Simulation (MASCOTS '06)*, (Washington, DC, USA), pp. 355–362, 2006.

[63] E. Weingärtner, F. Schmidt, T. Heer, and K. Wehrle, "Synchronized network emulation: matching prototypes with complex simulations," *SIGMETRICS Perform. Eval. Rev.*, vol. 36, no. 2, pp. 58–63, 2008.

[64] G. Judd and P. Steenkiste, "Repeatable and realistic wireless experimentation through physical emulation," in *2nd Workshop on Hot Topics in Networks (Hot-Nets II)*, (Boston, MA, USA), November 2003.

[65] K. Borries, G. Judd, D. Stancil, and P. Steenkiste, “FPGA-Based Channel Simulator for a Wireless Network Emulator,” in *IEEE 67th Vehicular Technology Conference (VTC2009-Spring)*, (Barcelona, Catalunya, Spain), April 2009.

[66] B. Walker, I. D. Vo, M. Beecher, and C. Clancy, “A demonstration of the meshtest wireless testbed,” *Testbeds and Research Infrastructures for the Development of Networks & Communities, International Conference on*, vol. 0, p. 1, 2009.

[67] S. Sanghani, T. Brown, S. Bhandare, and S. Doshi, “Ewant: the emulated wireless ad hoc network testbed,” vol. 3, pp. 1844 –1849 vol.3, mar. 2003.

[68] T. Staub, R. Gantenbein, and T. Braun, “VirtualMesh: An Emulation Framework for Wireless Mesh and Ad-Hoc Networks in OMNeT++,” *SIMULATION: Transactions of the Society for Modeling and Simulation International, Special Issue on Software Tools, Techniques and Architectures for Computer Simulation*, first published on July 2010.

[69] T. Staub, R. Gantenbein, and T. Braun, “VirtualMesh: An emulation framework for wireless mesh networks in OMNeT++,” in *2nd International Workshop on OMNeT++ (OMNeT++ 2009) held in conjuction with the Second International Conference on Simulation Tools and Techniques (SIMUTools 2009)*, (Rome, Italy), March 6-7 2009.

[70] T. Staub, R. Gantenbein, and T. Braun, “VirtualMesh.” http://www.iam.unibe.ch/rvs/research/software.html, 2009.

[71] The WISEBED Project, “WiseML.” http://www.wisebed.eu/images/stories/deliverables/d4.1.pdf, 2009.

[72] T. Baumgartner, I. Chatzigiannakis, M. Danckwardt, C. Koninis, A. Kroller, G. Mylonas, D. Pfisterer, and B. Porter, “Virtualising testbeds to support large-scale reconfigurable experimental facilities,” in *in Proceedings of the 7th European Conference on Wireless Sensor Networks (EWSN 2010), eds. J. Sa Silva and B. Krishnamachari and F. Boavida (LNCS 5970)*, 2010.

[73] A. Hergenröder, J. Wilke, and D. Meier, “Distributed Energy Measurements in WSN Testbeds with a Sensor Node Management Device (SNMD),” in *Proceedings of the 23th International Conference on Architecture of Computing Systems*, (Hannover, Germany), pp. 341–438, VDE Verlag, 2010.