SEVENTH FRAMEWORK PROGRAMME
THEME 3
Information and Communication Technologies

Grant agreement for:
Collaborative project, Small and medium-scale focused research project (STREP)

Deliverable D4.3:

Compendium of Experimental Scenarios, Traces and Benchmarks

Project acronym: WISEBED
Project full title: Wireless Sensor Network Testbeds
Grant agreement no.: 224460

Responsible Partner: TUD
Report Preparation Date: May 31, 2011

Contents

1 Introduction **3**

2 Experimental Scenarios and Traces **3**

2.1 Scenarios 3

2.2 Collected Data Sets 3

3 Usage Guide **14**

3.1 Importing Data Sets 14

3.2 Visualizing Data Sets 21

3.3 Benchmarking Data Sets 22

3.4 Benchmarking Algorithms 23

4 Conclusions **24**

References **25**

1 Introduction

In this deliverable we present a compendium of experimental scenarios, traces and benchmarks. In the first part a series of experimental scenarios and traces is presented. In the second part a usage guide is provided, detailing the tools to import, export, visualize and benchmark experimental traces in the WiseML format. The WiseML format is a standardized way of describing the collected data from various experiments. We would like to point out that WiseML has seen great adoption outside the WISEBED project by other European projects, such as CONET and SmartSantander, and commercial companies.

2 Experimental Scenarios and Traces

2.1 Scenarios

In Deliverable 4.1 we identified three major scenarios. They basically indicate various degrees of change (churn) in the underlying network topologies. The first scenario, entitled *Global Facility Management* (GFM), targets static networks such as the ones deployed for environmental monitoring over long periods of time. The second scenario, named *Object Tracking at Borders* (OTB), targets networks made up of static infrastructure in which a limited number of nodes can be mobile, while the third scenario, *Sports Tracking* (ST), targets fully mobile networks. Throughout the WISEBED project data sets for all three scenarios have been collected.

2.2 Collected Data Sets

During the WISEBED project, WiseML was proposed; a standardized way of describing the collected data from various experiments. This common data representation format has the additional advantage of being a simple and robust interface between the tools in the project (i.e., simulators and sensor network management software).

To aid the research community, a repository of experimental traces in WiseML format has been set up on the WISEBED website. Researchers can download data sets and contribute their own sets for other researchers to use. All data sets are formatted in WiseML, which makes them easy to use as there are many tools available to produce, process and analyze this data.

As part of this work package we collected a set of data sets mapping onto the three

scenarios defined in the previous section. In Table 1, an overview of the collected data sets is given with their characteristics expanded out in Table 2. Note that 7 out of the 22 collected data sets come from sources outside WISEBED, either from other European projects like CONET or from industry, like the e-novation data set. Each data set is presented in more detail in the following sections, which are grouped by scenario.

Data Set	Organization	Scenario	WiseML Size (MB)
WebDust	RACTI	GFM	37
lofar-2008	TUD	GFM	3,072
fence	FUB	OTB	1.9
target tracking	UNIGE	OTB	0.72
marathon	UZL	ST	4
UTAH-CIR	Utah	GFM	75
LQI	UNIGE/RACTI	GFM	0.061
powertrace	TUD	GFM	15,360
Pampa	ULANC	GFM	1.5
E-novation	E.Novation	ST	200
NULLMAC/XMAC	UBERN	GFM	47
Intellab	Intel	GFM	524
Tunnelvineyard	D3S (CONET)	OTB	225
BCNsolar	UPC	GFM	127
BCNlibrary	UPC	GFM	50
solar longterm	UZL	GFM	58
CitySense	Harvard	GFM	875
Fleming	BSRC Fleming	ST	98
Greenhouse	UTH	GFM	16
heartbeat	UZL	ST	8
ECC-evaluation	UBERN	GFM	163
hallway	TUBS, UZL	OTB	0.113

Table 1: Collected traces (overview)

Data Set	#Nodes	Sensors	Node Mobility	Deployment	#Measurements	Duration
WebDust	65	voltage, microphone, thermistor, accelerometer, light, magnetometer, temperature, humidity	no	indoor	332,742	8 months
lofar-2008	81	temperature, humidity, voltage	no	outdoor	80,474,427	6 months
fence	10	acceleration	no	outdoor	30,724	15 min
target tracking	31	event, intensity, trace type	yes	indoor	6,594	15 min
marathon	20	time, heart rate, packet counter, position	yes	outdoor	51,102	4 hours
UTAH-CIR	44	channel gain	no	indoor	467,700	16 days
LQI	7	LQI	no	indoor	393	3 hours

Data Set	#Nodes	Sensors	Node Mobility	Deployment	#Measurements	Duration
powertrace	24	current	no	indoor	176,055,393	76 min
Pampa	7	channel gain	no	indoor	6,268	60 min
E-novation	118	speed, course, mileage, satellite	yes	outdoor	2,929,088	2 months
NULLMAC/XMAC	7	temperature, humidity, voltage, light	no	indoor	312,544	6 days
Intellab	54	temperature, humidity, voltage, light	no	indoor	11,099,015	37 days
Tunnelvineyard	20	PRR, LQI	no	outdoor	2,508,000	5 hours
BCNsolar	1	voltage, current, charge, temperature, luminance	no	outdoor	944,412	12 months
BCNlibrary	7	temperature, humidity, light	no	indoor	585,840	14 months
solar longterm	8	temperature, voltage, current, capacity, lqi, solar, solar hours	no	outdoor	747,201	11 months
CitySense	19	wind direction, wind speed, temperature, air pressure, humidity, rain intensity, rain duration	no	outdoor	5,164,454	9 months
Fleming	5	temperature, humidity, voltage, mic, ammonia	yes	outdoor	117,454	2 months
Greenhouse	7	temperature, humidity, TSR, PAR, moisture, voltage	no	outdoor	91,656	3 months
heartbeat	1	heart rate	yes	indoor	69,025	2 days
ECC-evaluation	7	temperature, humidity, light	no	indoor	524,213	3 days
hallway	40	pressure	no	indoor	656	1 day

Table 2: Collected traces (characteristics)

2.2.1 Global Facility Management

WebDust The WebDust data set was collected in several buildings of RACTI at Patras, over a period of eight months. It contains recordings of a multitude of sensor nodes (65 in total) measuring light, temperature, humidity, acceleration, magnetic-field levels and barometric pressure.

The WebDust hardware setup is a heterogeneous set of sensor nodes and sensors,

organized in a multi-tier architecture.

The WebDust data set can be used for the design of algorithms for environmental monitoring and control or data aggregation. The heterogeneous set of sensors nodes and sensors makes it an interesting benchmark for data collection and fusing, as well as input for environmental control algorithms.

Lofar-2008 The lofar-2008 data set was collected by researchers from Delft University of Technoloy in an outdoor experiment targeted at precision agriculture (more details and results are presented in [9]). A static network was deployed in a potato field, which monitored the environmental characteristics of the field over a period of several months. All the nodes reported measured parameters (temperature and relative humidity) back to the gateway every ten minutes. Additionally, information such as statistics on neighbors, communication links quality, battery power, routing topology, etc. is embedded in the data set.

The outside deployment makes the lofar-2008 data set interesting, for example, by showing intermittent connectivity and the slow changes in topology characteristics reflect this.

The data set can be used to develop algorithms concerning mainly routing and data collection. Apart from that, information on failures is present in it as well. Encoded together with the readings information on several types of failures is available: communication failures (both missing data and duplicates are represented), undesired reboots of the hardware platforms, and time synchronization failures.

UTAH-CIR The UTAH-CIR data set was originally collected by Motorola Labs, Florida Communications Research Lab and the University of Utah. It contains over 9,300 measurements of indoor radio impulse responses in a 44-node network [3]. Researchers at the University of Utah used this data set to test signature-based localization algorithms. Wireless communication is prone to data errors and this data set may help the development of more robust communication algorithms. In addition, this data set can help in developing new localization techniques as well as test time-synchronization algorithms.

LQI The LQI data set was collected in the TCS Lab of the University of Geneva, Switzerland, in cooperation with CTI in Patras, Greece. A mobile node was placed in one of the grid positions and sent 200 messages to each of the 6 stationary gathering nodes. Each gathering node stored and transmitted to a central base station the LQI value for each of the messages it received and for each mobile node position. The

LQI values for each gathering node were then used to draw useful conclusions about the nature of wireless signal quality, with respect to transmission distance and angle. This data set can help develop localization algorithms and study further the use of signal quality in distance estimation or other algorithms.

Powertrace The powertrace data set was collected on the PowerBench testbed of Delft University of Technology. The data set contains detailed energy consumption measurements of BMAC and Crankshaft MAC protocol in various setups. The experiments were performed on T-Nodes, connected with a shunt resistor to a custom A/D converter board. The A/D converter sampled at a rate of 5 kHz, high enough to measure energy consumed by short events, such as a carrier sense operation by the CC1100 radio used in the PowerBench testbed. The data set is of particular use for research into creating more accurate energy consumption models for simulation purposes.

Pampa The Pampa data set is an example of probabilistic data dissemination through a sensor network. There is one source node sending data packets and the remainder of nodes taking delivery of those packets and aiding in their routing to further nodes in range. The source node sends packets with increasing IDs and the remaining nodes report the successful delivery of each successive packet ID. Nodes also report the total number of messages that they have sent (the intent of the Pampa protocol being to minimize total messages sent while maximizing delivery success rate).

NULLMAC/XMAC The NULLMAC/XMAC data set was collected on the WISEBED testbed at the University of Bern. The nodes and links that were used are displayed in Figure 1. The basic setup consists of a TCP server (receiving segments), a TCP client (sending segments), and a node chain of variable length, ranging from 2 to 6 hops.

The experiments examined the behavior of TCP on top of Contiki's NULLMAC / CSMA-variant and 3 different radio duty-cycling protocols: X-MAC, Low-Power-Probing and ContikiMAC. For each setting, the amount of successfully delivered segments (of roughly 30 bytes each) was measured in 15 experimental runs of 15 minutes each.

Intellab The Intellab data set was collected from 54 sensors deployed in the Intel Berkeley Research lab between February 28th and April 5th, 2004. Mica2Dot sensors with weather boards collected timestamped topology information, along with humidity, temperature, light and voltage values once every 31 seconds. Data was collected

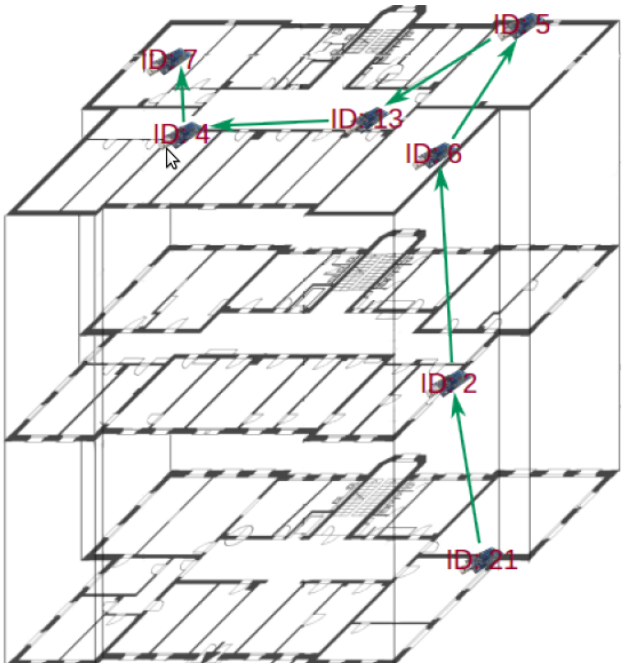


Figure 1: Node setup for the NULLMAC/XMAC experiment

using the TinyDB in-network query processing system, built on the TinyOS platform. The original data set was presented in a CSV file and converted with a custom script to WiseML.

Tunnelvineyard The Tunnelvineyard data set was collected by the D3S cross-institutional research group located in Trento, Italy as part of the TRITon project [12]. The TRITon project aims to improve safety and reduce maintenance costs of road tunnels, using a WSN-based control infrastructure. The data set was collected in three different outdoor deployments:

- an operational road tunnel, enabling assessment of the impact of vehicular traffic.
- a non-operational tunnel, providing insights into analogous scenarios (e.g., underground mines) without vehicles.
- a vineyard, serving as a baseline representative of the existing literature.

The setup, replicated in each deployment, used mainstream WSN hardware, and popular MAC and routing protocols. The goal of the experiments was to analyze and compare the deployments with respect to reliability, stability, and asymmetry of links, the accuracy of link quality estimators, and the impact of these aspects on MAC and routing layers. The analysis revealed that a number of criteria commonly used in the design of WSN protocols do not hold in tunnels. The data set is useful for designing and testing networking solutions operating efficiently in similar environments.

BCNsolar On the roof of a UPC building (see Figure 2), a solar-powered sensor is installed that is dedicated to capture and send data about weather and solar energy. It consists of an iSense node with a Environmental Module and a Solar Module. Inside an office in the same building, there is a PC with another iSense node attached via USB, dedicated to receive and process the data sent by the solar node.

Figure 2: iSense node with Environmental and Solar Modules on a UPC building roof

The software in the PC has two components: one stand-alone Java application that receives packets through the USB port and stores the data in daily rotated files, suitable for processing by external tools.

A web application[1] offers a public interface to access the captured data. It offers the possibility to download the data in three formats: the raw format in which the data is stored, a more human-readable format (an HTML table) and also in WiseML. It also allows to select a time interval and generate a graph of different measurements: battery charge, battery current, environmental temperature and environmental light.

BCNlibrary The Biblioteca Rector Gabriel Ferraté (BRGF) is one of the libraries located at the UPC North Campus in Barcelona (see Figure 3). The 6-floor building stores important bibliografic sources and the UPC wanted to monitor its environmental data in order to preserve them in the best state, to guarantee the comfort of the people inside, and to efficiently administrate the energy consumption.

The differences in temperature and humidity during the day are the most important parameters to measure. Significant temperature differences occur due to the strong influence of the sun in the southeastern side of the building, with big glass windows on it. Also the natural/artificial light pattern is an important piece of information to know in order to improve and efficiently manage the artificial lighting of the building.

The measured parameters are temperature, humidity and light. The total monitored area is 6.000 m^2, distributed over 6 floors. The deployment is completely wireless

[1] Available at `http://albcom.lsi.upc.edu/wisebed/`.

Figure 3: Outside and inside of the BRGF library

and consists of 13 sensors, 4 relay stations and 1 base station. Measurements take place every 10 minutes, which produces a total amount of 5.600 measurements per day. All those measurements can be consulted and compared though a web service implemented by the company DEXMA, which also provided a configurable automatic control system that alerts when the temperature or the humidity are not kept within the desired range (nowadays, [21°C,26°C] for temperature and [30%,70%] for humidity). The architecture of the deployment is such that the size of the deployment can be enlarged and the location of the sensors can be modified whenever necessary.

The same web application (see Footnote 1) that allows to consult the BCNsolar data, also gives access to the BCNlibrary data (in WiseML as well as in other formats).

Solar Longterm The Solar Longterm data set provided by an industrial partner of UZL consist of evaluation data from an outdoor solar testbed located in Lübeck, Germany. It was running for one year and consists of 8 iSense sensor nodes with different solar panels. The nodes collect data on the battery voltage and capacity as well as solar hours and temperature values.

CitySense The CitySense data set is based on the data collected from the CitySense project [13]. CitySense consists of wireless sensors deployed across a city, such as on light poles and private or public buildings. Each node consists of an embedded PC equipped with various sensors for monitoring weather conditions and air pollutants. The measurement frequency varies per sensor from sub-minute samples to multi-minute. The network is operational since 2007 and contains over 31 million measurements.

Greenhouse The Greenhouse data set was collected at a concrete and glass greenhouse (240 by 100 m), situated at the farm of the University of Thessaly (UTH), at

the Velestino area near Volos, Greece. The sensors were positioned in a grid layout. Each sensor node was collecting data about the following environmental factors: temperature, humidity, TSR (Total Solar Radiation), PAR (Photosynthetically Active Radiation), and soil moisture. The data collection lasted for 2 months and 91,656 measurements were collected.

ECC-evaluation The ECC-evaluation data set was collected on the WISEBED testbed at the University of Bern.

7 MSB-430 sensor nodes operating with the ScatterWeb operating system have been used to evaluate the performance of a selection of eight different Error Correction Codes (ECCs), ranging from simple bit-repetition schemes over hamming-based codes to complex and powerful Bose-Chaudhuri-Hocquenghem (BCH) codes.

The topology used for evaluating the different codes is displayed in Figure 4. The nodes were placed in a V-shaped topology with the sink in the upper left corner.

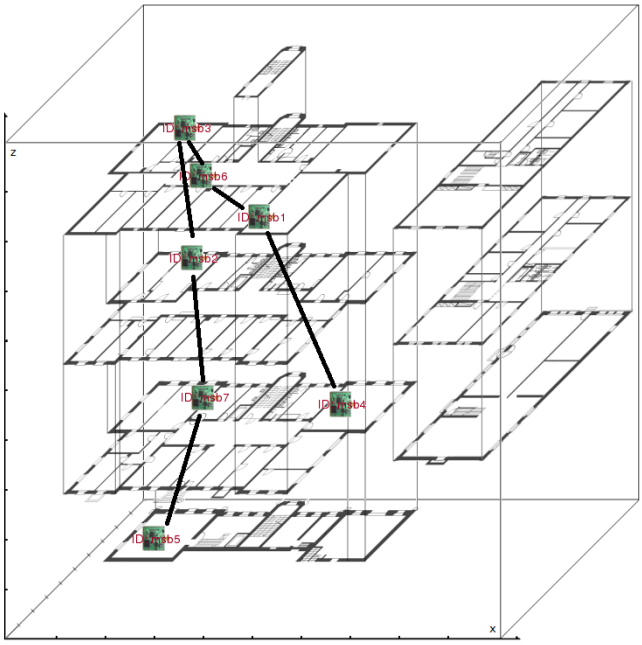


Figure 4: Node setup for the ECC evaluation data set

Nodes were generating 1,000 packets of 32 bytes each encoded with different ECCs and sending it towards the sink (across the links depicted). Every node except for the sink generated a packet at a random uniform interval between 2 and 2.8 seconds.

2.2.2 Object Tracking at Borders

Fence This data set belongs to the class of limited mobility networks, with some mobile nodes moving within a static network. The deployment consists of 10 Scat-

terWeb MSB nodes, each attached to a steel construction fence, in the patio of the CS institute of FUB. Each fence is 3.5 m wide and 2 m high. Each sensor node is attached at the right side of a fence at a height of 1.65 m. The sensor node is equipped with an accelerometer, which helps in measuring the movement of the fence under different events. The possible events occurring at these fences include a person kicking, leaning, shaking, peeking, and climbing over the steel construction fence.

Each sensor node is attached to a Freescale Semiconductor MMA7260Q accelerometer, which is sampled at 10 Hz with a sensitivity setting of 1.5 g. The raw data is shown in terms of intensity I being defined as follows:

$$I = |(v_{x\_last} - v_{x\_current})| + |v_{y\_last} - v_{y\_current}| + |v_{z\_last} - v_{z\_current}|$$

where $\overrightarrow{\mathbf{v}}_{last}$ is the previous three dimensional vector and $\overrightarrow{\mathbf{v}}_{current}$ is the present.

The data set has been used to study the possibilities of minimizing the traffic on the wireless channel of a WSN.

Hallway The hallway data set has been collected on the WISEBED testbeds at TUBS and UZL, with nodes connected via virtual links. There are 30 nodes from the TUBS testbed, and 10 nodes from UZL. Each node in Lübeck is connected to four nodes in Braunschweig. The nodes in Braunschweig are connected to load sensors, installed beneath floor tiles in a hallway [2] and being able to detect passing people.

The output of the TUBS testbed is raw load sensor data, generated whenever a single sensor is loaded. Data is then sent via a virtual link to a node in the UZL testbed, printing a debug output when a certain number of messages has been received.

In this data set there is a person passing the hallway, producing corresponding output. Whenever an UZL node outputs true, the person has been identified in a certain section of the Braunschweig hallway.

Target Tracking The target tracking dataset was collected on the UNIGE Testbed (26 isense nodes) with similar results observed on CTI (20 isense nodes) and UZL (44 isense nodes) testbeds. That is excluding tagged assets. These assets can be mobile or not and are either humans or objects holding an isense node that beacons periodically.

The concept of target tracking [11] is a two step process. First, it includes the detection of the target asset and the efficient diffusion of this information in the WSN topology, in the form of gradient based destination oriented directed acyclic graph (DODAG) for each asset. Secondly it involves localization of the target asset with

the use of generated software agents (generated by tracker assets) that follow the DODAG gradient towards the detection point and report back. Both target assets and tracker assets can be mobile and multiple trackers and targets are supported.

The context of the dataset traces are divided in two categories:

- **Measurement of load of messages**, as a means of approximating workload of each sensor, noise levels, energy consumption and distance sensitivity of all previous regarding the detection points. Samples regarding the load of all types of messages (diffusion phase, detection phase, querying phase, ingoing-outgoing) are taken every one second from each sensor. Since the diffusion phase might potentially include variable message payloads, number of bytes for all previous types are also collected to provide precise analogies.
- **Measurement of reporting time**, is the time needed for a generated agent to traverse the topology in order to locate a mobile asset and return to a (mobile or not) tracker.
- **Measurement of reporting agent success ratio**, is the number of all generated agents sent from a specific tracker versus the number of agents returned. The success ratio can be affected by either agents tracking the same asset overlapping due to latency, unreliable communications or timeouts of roving time.

The experiments where performed initially with one target and one tracker and upscaled to three trackers and three targets. For each set of experiements we permutated with diffusion phase back-off timers, asset beaconing periods, immobility of assets, partial mobility of assets (only targets), and full mobility of assets.

2.2.3 Sports Tracking

Marathon The marathon data set was collected during the 2006 Flensburg marathon using the MarathonNet wireless sensor network [15]. During the race, runners wore a small sensor node, which communicated with base stations placed along the track. The base stations collected runner id, timestamp, position and heart rate. With this information, biometric data can be correlated with time and location information and used by the participants to analyze their performance. In addition, it can be used to follow the course of the race in real-time.

This data set can be used in developing routing algorithms for topologies with varying degree of connectivity and density, in applications with low-latency requirements.

E-novation The e-novation data set, provided by an industrial partner of TUD, was obtained from a collection of 118 trucks equipped with GPS-enabled wireless sensor nodes, driving through Europe and the United States. The sensor nodes collected speed, course, mileage, location and number of GPS satellites. The data set is of particular use for modeling vehicular movements.

Flemming The animal house unit of Biomedical Sciences Research Center "Alexander Fleming" provided a husbandry of experimental mice (it is used by the biomedical research community). The main goal of the monitoring system was to provide 24 hour monitoring of the mice mothers and newborn mice, and to investigate the impact of various environmental factors such as temperature, humidity, ammonia concentration and man-made noise to newborn mice development. The data collection system was based on a combination of static nodes (to be deployed in particular places in the house) and mobile nodes of very small size (to be attached to mice). The data collection lasted for 2 months and 117,454 measurements were collected.

Heartbeat The Heartbeat data set was collected by a member of the UZL WISEBED group. It contains heart rate values with a frequency of one value each second. The heart rate values were provided by a Polar belt and transmitted wireless to a Pacemate sensor node. The data set contains two traces. The first one started in the afternoon and collected heart rate values over the whole night till the next morning. The second trace was recorded during a usual day in the office from the morning till the end of working day.

3 Usage Guide

This usage guide presents a number of tools to import, export and visualize data traces formatted in WiseML. Secondly it introduces methodologies for benchmarking experimental traces and benchmarking algorithms using data traces.

3.1 Importing Data Sets

There are several ways to obtain data sets for evaluating algorithms, or that can be used as input for simulators. One possibility is to use traces from data repositories like CRAWDAD [3], which provides experimental traces in their original format, and the WISEBED WiseML data repository [17], which provides data sets in the

WiseML format. Using a common standard is to be preferred over proprietary formats. Existing data sets can be converted to WiseML for usage in WiseML-enabled tools. Experimental traces in WiseML format can also be natively generated by several tools.

Some testbed management infrastructures use WiseML for defining an experimental setup and use the WiseML-format for exporting their experimental results. In a similar fashion, several simulators can use WiseML-formatted experimental traces as input for setting up or controlling a simulation (topology, nodes, links, failures). The same WiseML can be used for input in a simulator or an actual testbed. Additionally, simulators can export their results in WiseML format for further analysis, or as input for an actual testbed deployment. More specialized simulators, like mobility simulators, can export their mobility models as input for other simulators.

The following sections detail several testbed management infrastructures and simulators that include native WiseML support.

3.1.1 TARWIS

The testbed management application TARWIS (described in-depth in WP2 deliverables D2.2 and D2.3) integrates the WiseML language for several purposes. It uses WiseML for reading and parsing the necessary information about its underlying network resources. Furthermore, it uses WiseML for storing and generating the output of the experiment log and debug traces in a common well-defined format.

Network resources definition In order to read the network resources (node type, sensors, positions, etc), TARWIS calls the *getNetwork()* function of the SessionManagementService API, and retrieves a WiseML document listing the entire network configuration. It uses the retrieved positions to display the nodes of the network in a network graph. Listing 1 shows one *instantiation* of a node entry. The node type and configuration are described in the *defaults* section.

```
<node id="urn:wisebed:node:ubern:1">
    <position>
        <x>69</x>
        <y>20</y>
        <z>52</z>
    </position>
    <gateway>true</gateway>
    <description>Node 1 - Office 205 (2nd Floor)</description>
</node>
```

Listing 1: Node entry in SessionManagementService at UBERN

Experiment log and debug traces As soon as an experiment is scheduled and configured within TARWIS, TARWIS retrieves experiment output (e.g. debug information, sensor values) via the *receive* function and stores that in an internal database. As soon as the experiment ends, all output is exported to a WiseML-file. This WiseML-file comprises all important information about an experiment run, (e.g. where the experiment took place geographically, what kind of nodes were used, what their sensor configuration was, etc). Storing all this experiment-related information in one WiseML file offers many advantages besides the possibility to easily use it for post-experiment analysis. As it includes all crucial information of an experiment, it further allows to make the experiment data public to other research partners in a common well-defined language, giving them the opportunity to repeat the same or similar experiment, e.g. trying to improve the results. Hence, having integrated WiseML into the testbed management system pushes research on wireless sensor networks one crucial step towards transparency and repeatability of sensor network experimentation.

Experiment Runs Experimental research is usually driven by the need to obtain statistically significant data. Researchers therefore usually have to carry out the very same experiments multiple times in a row, in order to obtain the necessary amount of data upon which they can rely for a sound analysis. In order to ease the burden for experimental researchers to schedule the same experiment over and over again, TARWIS integrates the *Runs* option in order to easily define how many *experiment runs* the experiment shall consist of. The entire duration of the reservation is then split into the specified number of runs. After each run, the output is written to an output file and subsequently added to a zip archive. This archive is then made available for download or sent to the user as an attachment to an email.

```
<wiseml> [...]
<trace id="experiment_UBERN_uniqueID_23453323">
    [...]
    <timestamp>3605.164612</timestamp>
    <node id="urn:wisebed:node:ubern:9">
        <position>
            <x>85</x>
            <y>80</y>
            <z>52</z>
        </position>
        <data key="textOutput">Temperature 2 15</data>
        <data key="textOutput">Light 1 202</data>
    </node>
    <timestamp>3685.164612</timestamp>
    <node id="urn:wisebed:node:ubern:3">
        <data key="textOutput">Light 2 480</data>
        <data key="textOutput">Light 1 223</data>
    </node>
    [...]
</trace>
```

Listing 2: Excerpt from a TARWIS-generated Experiment Trace

The WiseML code sample above lists two trace events retrieved in a small experiment at the UBERN testbed. For each output line, one can determine the exact time (with μsecs precision) relative to the experiment start time (c.f. the *timestamp* tag), the position of the node (hence, with mobile nodes, the node movement can also be captured) and the output itself. The WiseML-file generated by TARWIS can therefore describe to a very high degree what has happened at a certain time during the experiment.

Experiment results are stored at each TARWIS installation, in a designated TARWIS database. Researchers carrying out experiments can choose to declare experiments *public*. When choosing *public*, other WISEBED members can monitor (but not interact with) the ongoing experiment at run-time, and download the resulting WiseML-file. The experiment *owners* can further delete or upload WiseML files to the TARWIS database.

3.1.2 Testbed Runtime

The Testbed Runtime is a set of programs that together form a wireless sensor networks testbed infrastructure. It implements the APIs defined by WISEBED, namely RS (Reservation System), SNAA (Sensor Network Authentication and Authorization) and iWSN (Wireless Sensor Network API). The project contains three types of programs:

1. Testbed infrastructure software (iWSN, RS and SNAA implementations to be deployed on an actual testbed),
2. Testbed federators (iWSN Federator, RS Federator and SNAA Federator, used to federate multiple testbeds), and
3. Testbed clients (Scripting Client, Debugging GUI Client, Experimentation Scripts)

The iWSN infrastructure software is the core of the whole sensor networks testbed infrastructure as it realizes the actual experimentation logic. It is responsible for programming the sensor nodes, running the experiment and obtaining the results. The iWSN Federator component can be used to federate several testbeds. It delegates these tasks to other iWSN instances running on various locations and collects the results from them, thereby providing the user with a single endpoint to work with for federated experiments.

The RS (Reservation System) module implements a system that is responsible for reserving a set of nodes for a period of time for a specific user. There are three implementations: “Single URN Prefix” (an implementation handling reservations for exactly one testbed), Federator (delegating to e.g. Single URN Prefix or other RS

implementations) and "Dummy" (for debugging purposes or desktop usage where no actual reservations have to be made).

The SNAA (Sensor Network Authentication and Authorizing) module is responsible for authenticating users and authorizing a set of actions of a specific user. There are several implementations, allowing machine-driven authentication with Shibboleth federations, all backends of the Java Authentication Authorization API including Kerberos, Unix, Windows, Solaris, PAM, and htpasswd files. The wide range of implementations gives providers of testbeds several options, so they may choose whatever fits their current infrastructure and purpose of the testbed. As with the RS module, a Federator module enables delegation of the authentication and authorization tasks. It can be used to federate several testbeds to virtually create a larger one that appears as a single testbed to users of the federator.

The Testbed Runtime components are written in Java, which makes them deployable on a wide variety of platforms that support the Java VM. The modular setup of the Testbed Runtime leverages support for a wide range of testbed setups. From a very simple personal desktop testbed consisting of just a few nodes, to a single testbed setup, to very large, federated multi-tier architectures, with multiple testbed servers and gateways. It has been successfully deployed not only on server hardware but also on commodity hardware (netbooks acting as gateways to the WSN) as well as on low-priced embedded hardware (e.g. Sheeva Plug, a fanless ARM based system). Furthermore, it ships with the WISEBED Virtual Machine so that users can have their own personal desktop testbed in a matter of minutes, giving them the opportunity to use all the client applications available for the WISEBED APIs.

The Testbed Runtime provides several testbed clients. The Scripting Client makes it easy to perform highly interactive and/or automated experiments. It is based on the BeanShell interpreter that executes Java 1.4-like scripts and provides various helpful additional APIs to the script programmer. Therefore, the Scripting Client makes it easy to automate the whole process of authenticating to the testbed, reserving nodes, creating an experiment, receiving the result and ending the experiment. Testbed Runtime also ships with a set of predefined and commonly used scripts for the aforementioned use cases, called Experimentation Scripts assembly.

Basically all components of Testbed Runtime either produce or consume WiseML files:

- The iWSN infrastructure software delivers a testbed self-description upon calling the iWSN API method `getNetwork`.
- The iWSN Federator uses a WiseML merger component for stream-based merging of several WiseML files which was originally written for the federation purpose but can be used in other contexts, too.

- The experimentation scripts all interpret WiseML to learn about the available nodes, their types and features and some produce WiseML traces as they receive experiment outputs from the testbed.
- The WiseML player component (currently in development) takes a WiseML file as input and executes iWSN API calls according to the `setup` part of the WiseML file, thereby creating an actual playback functionality for traces.
- WiseUI, a Web-based WISEBED client comparable to TARWIS, interpretes the testbed self-descriptions in WiseML format to display e.g. available nodes. The project is currently in development and its source code is available under `https://github.com/itm/wiseui`.

Testbed Runtime is developed as an open source project. Source code, releases and documentations can be downloaded from the projects homepage: `https://github.com/itm/testbed-runtime`.

3.1.3 Shawn

Shawn [8, 6] is an open-source discrete event simulator. It focuses on algorithms, large-scale networks (up to 10^6 nodes), and speed. One central approach of Shawn is to simulate the effect caused by a phenomenon, not the phenomenon itself. For example, instead of simulating a complete MAC layer including the radio propagation model, only the effects (i.e., packet loss and corruption) are modeled in Shawn. This has several implications for simulations: they get more predictable and meaningful, and there is a huge performance gain, because such a model can often be implemented very efficiently. The authors argue that it makes sense to simplify the structure of some low-level parameters since their time-consuming computation can be replaced by fast simulations, as long as the interest in the large-scale behavior of the macro-system focuses on unaffected properties.

Shawn has been extended with the ability to read and write WiseML files. It is possible to both play back data generated by real testbeds, as well as data generated by other simulators. The latter has successfully been done with output from the COOJA simulator where mobile nodes moved randomly in the topology.

3.1.4 UNIGE WSNGE

The WSNGE simulator toolkit [7] from UNIGE also uses WiseML as an export option. The WiseML it produces focuses on the network description and allows export of node positions, custom flags (such as planarization flags that are supported by the

toolkit) and descriptions for the nodes as well as node IDs in standard URN format. The WiseML files it produces are portable to other software that supports that standard, as shown among others in [10], where one WSNGE WiseML file was visualized in COOJA, resulting in an identical network.

3.1.5 COOJA

COOJA [14] is a flexible Java-based simulator initially designed for simulating networks of sensors running the Contiki operating system [5]. COOJA simulates networks of sensor nodes. COOJA can execute Contiki programs in two different ways: either by running the program code as compiled native code directly on the host CPU, or by running compiled program code in MSPSim. COOJA is also able to simulate nodes developed in Java at the application level. Java-based nodes enable much faster simulations but do not run deployable code. Hence, they are useful for the development of distributed algorithms. Emulating nodes allows control and retrieval of more fine-grained execution details compared to Java-based nodes or nodes running native code. Finally, native code simulations are more efficient than node emulations and still simulate deployable code. Combining the different levels in the same simulation can give both an efficient simulation as well as fine-grained execution details on selected nodes.

COOJA has been used for rapid prototyping of wireless sensor network mechanisms and applications. Furthermore, it has been used for protocol evaluation. COOJA can import and export its settings and simulation results in WiseML format. Importing WiseML enables COOJA to simulate a predefined topology or node mobility, and using the collected traces as stimuli for the emulated nodes. Exporting WiseML makes it possible to evaluate the simulation results with WiseML visualizers and data analyzers.

3.1.6 BonnMotion

Most simulation-based performance evaluation studies are limited to static networks. However, node mobility is found to have significant impact on the evaluation results. In order to provide WSN simulators such as COOJA and MiXiM with a standardized description of the nodes' mobility, we extended the mobility scenario modeling tool BonnMotion [1] to support WiseML.

BonnMotion is open-source Java software which creates and analyzes mobility scenarios. Currently, Random Waypoint, Gauss-Markov, Manhattan Grid, Reference Point Group Mobility, and the Disaster Area model are supported. For the analysis of the generated scenarios, different metrics such as velocity, relative mobility, dwell

time, link duration, time to link break, node degree, partitions, and k-connectivity are supported.

The native format in which BonnMotion saves the movement traces is waypoint-based. A waypoint is a position at which the movement of a node (e.g., direction or velocity) changes. This implies that during the simulation, for each event, the current node positions have to be calculated based on the waypoints. If there are many events, this may have a negative impact on the runtime of a simulation. However, if an interval-based format is used, the current node positions do not have to be calculated for each event, since the nodes are regarded as stationary for an interval. The positions of the nodes are updated periodically after each interval by a position update event. If smaller intervals are used, accuracy is higher but there are also more events. Thus, the execution time overhead of adding mobility to WSN simulators mainly depends on the size of the intervals.

In addition to the other output formats, we implemented WiseML output in BonnMotion based on the interval format and the `<timespan>` element specified in [4]. As WiseML can be used for scenario specification, this enables the use of all BonnMotion scenarios in testbeds as well as simulators that support WiseML. Thus, the wide range of mobility models supported by BonnMotion can be used for the evaluation of wireless sensor networks.

3.2 Visualizing Data Sets

Visualizing a data set gives useful insights of the experimental results, either as a first evaluation or as an indicator of hidden relations. As researchers often write their own specific visualizer, employing a common data format like WiseML enables the reuse of visualizers.

3.2.1 WeyesBED

The WeyesBED visualization tool allows the visualization of WiseML files. The visualizer aims at seamless trace replay of an experiment described in WiseML, thus allowing interactive visual description, and representation of distributed algorithms and protocols. By being able to monitor every experiment state and their transitions, researchers will be able to gain better insight on the inner workings of their algorithms and potentially detect key-events that lead to problematic states or bugs.

To keep the visualization application as portable as possible, the visualizer is written in Java and uses Java3D as OpenGL frontend.

In essence, WiseML models WSN testbeds as graphs. The visualizer extends this

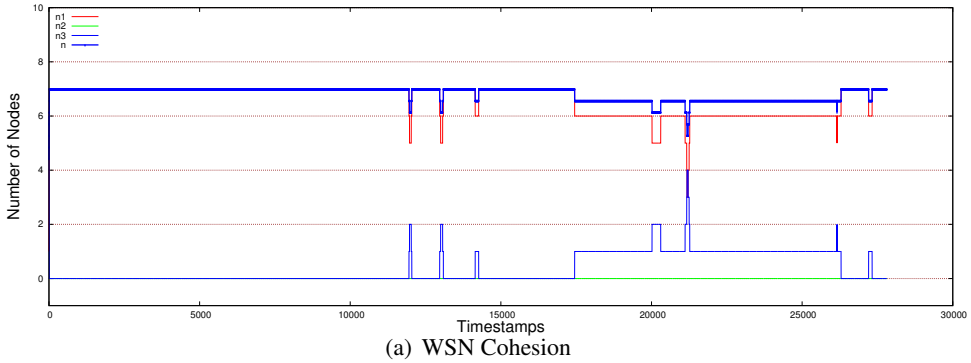
concept to three dimensions. An experiment is visualized as a graph in 3D space. Nodes are displayed as spheres and links as lines that connect spheres. The user can navigate in the 3D space using a mouse and keyboard to switch focus from one entity to a group of entities, or even to the whole network. When a graph entity is selected, its capabilities are displayed on the right part of the screen. Trace navigation and replay is implemented by a video-like interface.

3.3 Benchmarking Data Sets

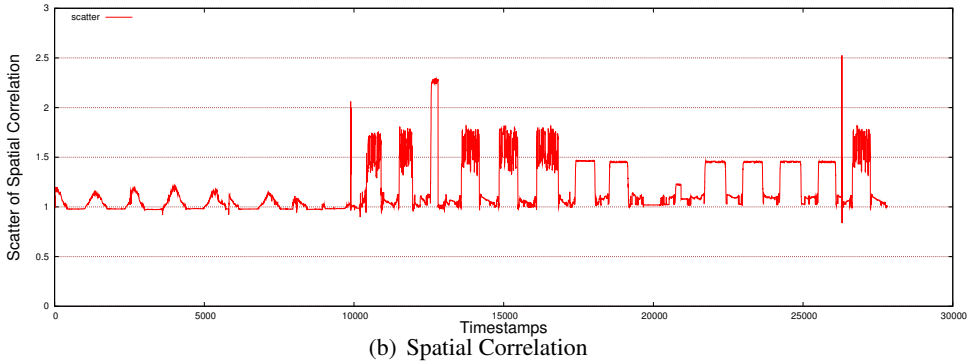
An important advantage of public data sets would be to use them for developing and benchmarking algorithms. In that respect it would be nice if we could classify the data sets such that researchers would know which of the many traces to use for their work, such as the evaluation of a new data aggregation method. Some traces are easier than others, and coarsely one can characterize a trace as having *low complexity level*, from the perspective of an algorithm, if it presents a situation in which the topology of the network was stable, there were no errors and the sensed phenomenon showed little variation. A trace corresponding to a dynamic environment (as in real-life scenarios), in which devices fail and large amounts of noise corrupt the otherwise incomplete data, might be characterized as having a *high complexity level*. This rule-of-thumb classification, however, needs to be refined and automated to provide researchers with a more descriptive notion of what characteristics a certain data trace provides.

Within the WISEBED project partners from RACTI and TUD have developed a set of three metrics to describe data traces [16]. In particular, they classify the *network cohesion*, the *spatial correlation*, and the *temporal correlation* into three levels (low, average, high). The network cohesion captures the dynamics in the WSN during the deployment, and basically looks at the number of nodes actively reporting data at the sink. The other two metrics classify the correlation in observed sensor readings in relation to readings from neighboring nodes and over time.

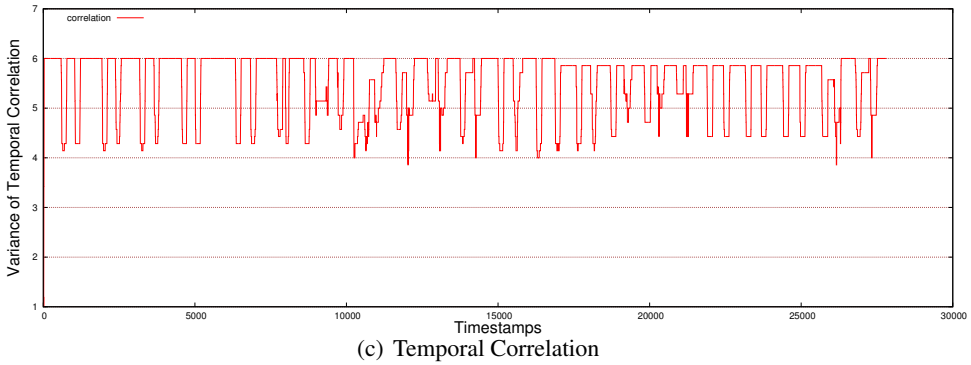
Figure 5 shows an example of how the metrics vary over time for the Greenhouse data set. Note that the spatial correlation metric shows a periodic pattern, which corresponds to a 24-hour rhythm, with data during the night being more correlated than during the day. The variations are quite small though, as the ones for network cohesion and temporal correlation, making this trace one of average difficulty. Table 3 shows the classification of the five data traces analyzed so far.



(a) WSN Cohesion



(b) Spatial Correlation



(c) Temporal Correlation

Figure 5: Greenhouse trace for the total 91,656 measurements

Dataset	*WSN Cohesion*	*Spatial Correlation*	*Temporal Correlation*	Overall *Complexity Level*
Lofar-agro	avg	low	avg	avg
Greenhouse	low	avg	avg	avg
Flemming	high	avg	high	high
CitySense	low	high	high	high
WebDust	high	low	high	high

Table 3: Aggregation Evaluation: Complexity Level

3.4 Benchmarking Algorithms

The next step, after classification, is to use the data traces to benchmark algorithms. To show the feasibility of the data sets from Table 3 have been used in a simulation environment as input to three data aggregation algorithms (TDS, PCSA, and LogLog) repeatedly performing a distinct count of the sensor values in the trace. Full details are provided in [16]. Figure 6 shows an example output, plotting the error for the

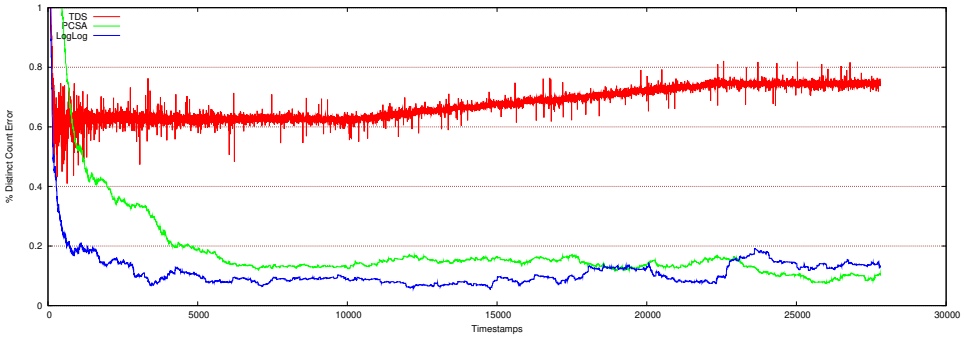


Figure 6: Data aggregation benchmark results for Greenhouse trace

three algorithms over time. All three achieve good results (less than 1% error), which was to be expected as Greenhouse is only of average complexity. Nevertheless, the PCSA and LogLog algorithms clearly outperform TDS. Exactly this kind of information is needed by researchers to select the appropriate algorithm based on running a trace matching the anticipated characteristics of their next deployment.

4 Conclusions

This deliverable presented a number of experimental scenarios, a large number of traces and presented new benchmarks to qualify data sets. A diverse set of data traces has been collected, varying greatly in deployment, size, node mobility and experiment duration. The data sets came not only from WISEBED partners, but from other European projects and industry as well. A public data repository has been set up on the WISEBED website [17], where researchers can access the data sets and contribute their own data sets.

The compendium provides a usage guide for importing, exporting, visualizing and benchmarking data traces and using data traces for benchmarking algorithms, using state of the art tools. WiseML, a common data representation format developed during the WISEBED project, is the enabling technology in the integration of and cooperation between tools. These range from testbed management infrastructures, like TARWIS and Testbed Runtime, to simulators, like Shawn and COOJA, to visualizers, like WeyesBED.

For benchmarking data sets, the usage guide provides a set of metrics, developed within the WISEBED project, to automate the qualitive anaysis and classication of data sets. As a next step, the data sets have been used as input for simulators to benchmark algorithms. The classification of data sets, together with the benchmarking results of algorithm, gives researchers valueable information for selecting the appropriate algorithm based on the scenario and the anticipated characteristics.

References

[1] N. Aschenbruck, R. Ernst, E. Gerhards-Padilla, and M. Schwamborn. BonnMotion - a Mobility Scenario Generation and Analysis Tool. In *Proc. of the 3rd Int. ICST Conference on Simulation Tools and Techniques, Torremolinos, Spain*, pages 1–10, 2010.

[2] T. Baumgartner, S. Fekete, T. Kamphans, A. Kröller, and M. Pagel. Hallway monitoring: Distributed data processing with wireless sensor networks. In P. Marron, T. Voigt, P. Corke, and L. Mottola, editors, *Real-World Wireless Sensor Networks (REALWSN10)*, volume 6511 of *Lecture Notes in Computer Science*, pages 94–105. Springer Berlin / Heidelberg, 2010.

[3] Crawdad. http://crawdad.cs.dartmouth.edu/.

[4] Delft University of Technology (TUD). WiseML description (TR-RS-WISEMLDESCRIPTION-1.0). Technical report, The WISEBED consortium, 2010. `http://www.wisebed.eu/images/stories/deliverables/TR/TR-RS-WISEMLDESCRIPTION-1.0.pdf`.

[5] A. Dunkels, B. Gronvall, 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, Florida, USA, November 2004.

[6] 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.

[7] M. Karagiannis, I. Chatzigiannakis, and J. Rolim. WSNGE: a platform for simulating complex wireless sensor networks supporting rich network visualization and online interactivity. In *Proceedings of the 2009 Spring Simulation Multiconference*, SpringSim '09, pages 35:1–35:8, San Diego, CA, USA, 2009. Society for Computer Simulation International.

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

[9] K. Langendoen, A. Baggio, and O. Visser. Murphy loves potatoes: Experiences from a pilot sensor network deployment in precision agriculture. In *14th Int. Workshop on Parallel and Distributed Real-Time Systems (WPDRTS)*, pages 1–8, apr 2006.

[10] Q. Li, F. Österlind, T. Voigt, S. Fischer, and D. Pfisterer. Making wireless sensor network simulators cooperate. In *Proceedings of the 7th ACM workshop on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks*, PE-WASUN '10, pages 95–98, New York, NY, USA, 2010. ACM.

[11] A. Marculescu, S. Nikoletseas, O. Powell, and J. Rolim. Efficient tracking of moving targets by passively handling traces in sensor networks. In *Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE*, pages 1 –6, Dec. 2008.

[12] L. Mottola, G. P. Picco, M. Ceriotti, S. Gună, and A. L. Murphy. Not all wireless sensor networks are created equal: A comparative study on tunnels. *ACM Trans. Sen. Netw.*, 7(2):1–33, 2010.

[13] R. Murty, A. Gosain, M. Tierney, A. Brody, A. Fahad, J. Bers, and M. Welsh. CitySense: A vision for an urban-scale wireless networking testbed. In *2008 IEEE International Conference on Technologies for Homeland Security*, 2008.

[14] F. Österlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt. Cross-level sensor network simulation with COOJA. In *Proceedings of the First IEEE International Workshop on Practical Issues in Building Sensor Network Applications (SenseApp 2006)*, Tampa, Florida, USA, Nov. 2006.

[15] D. Pfisterer, M. Lipphardt, C. Buschmann, H. Hellbrueck, S. Fischer, and J. H. Sauselin. MarathonNet: Adding value to large scale sport events - a connectivity analysis. In A. Press, editor, *Proceedings of the International Conference on Integrated Internet Ad hoc and Sensor Networks (InterSense'06)*, page 12, May 2006.

[16] E. Theodoridis, I. Chatzigiannakis, and S. Dulman. Post-processing in wireless sensor networks: Benchmarking sensor trace files for in-network data aggregation. *Journal of Network and Computer Applications*, In Press, Corrected Proof:–, 2011.

[17] WISEBED WiseML data repository. `http://www.wisebed.eu/index.php/wiseml-trace-files`.