Experimentation Made Easy[⋆]

Mesut Güneş, Bastian Blywis, Felix Juraschek, and Olaf Watteroth

Distributed Embedded Systems
Institute of Computer Science
Freie Universität Berlin, Germany
{guenes,blywis,jurasch,watterot}@inf.fu-berlin.de

Abstract. Scientifically sound network studies require the execution of large series of experiments. Researchers usually have to execute experiments manually, a labor-intensive and error-prone task, since there is no automation of the overall experimentation process. This task becomes especially hard on a distributed testbed and the researcher has to deal with additional challenges.

In this paper we introduce the *Distributed Embedded Systems Testbed Management System* (DES-TBMS) for the automation of experimentation in wireless mesh networks and wireless sensor networks. DES-TBMS consists of several components including a domain specific language to describe and define experiments, a scheduler to manage and control experiments, a distributed monitoring system to gather system data, a visualization tool, and an evaluation tool to compute performance metrics from collected experiment data. Thus, DES-TBMS supports the researcher during the design, execution, and evaluation of experiments.

1 Introduction

Experimentation is one of the pillars of scientific work. There are different ways to perform an experiment, such as mathematical constructs, simulation, emulation, and testbeds [1]. These environments provide different degrees of realism and have to be used carefully and questioned whether they are suitable for a particular study.

In this paper the research focus is on distributed wireless mesh and wireless sensor networks. In the last decade simulation environments for these networks proved useful to develop and analyze network protocols. However, results obtained from simulations can not be transferred directly to real world network deployments, since the provided realism of current simulators is restricted to a certain extent because their models contain too many simplifications [2]. Still, most of the research on wireless networks is based on simulations [3]. The setup of testbeds with real hardware proved useful to fill this gap between simulation environments and real world deployments. However, testbeds are more difficult

[⋆] This work has been partially sponsored and supported by the European Union OPNEX project.

to deploy and harder to maintain. But in exchange, they promise more accurate and reliable results.

Consequently, there is a trend in the wireless network community to construct testbeds and use them for research. Unfortunately, a testbed usually does not support the user in the experimentation workflow. Users usually access the testbed by directly connecting to the individual nodes, e.g., via `ssh`. Then, the users themselves have to take care of all steps of the execution of an experiment. In consequence, experiments in a testbed environment are difficult to repeat and results are difficult to compare. Few testbeds have addressed this issue by either introducing an experimentation framework or by providing support for experiment description files as used in simulation environments.

The contributions of this paper are twofold. On the one side, we introduce a workflow for network experiments on a testbed. On the other side, we present the *DES Testbed Management System* (DES-TBMS) for the automation of experiments based on the described workflow. The goal of the DES-TBMS framework is to enable the researcher to run experiments as comfortable as in simulation environments and scientifically sound. DES-TBMS consists of several components each designated to a particular step in the experiment workflow. It comprises a domain specific language to describe and define experiments, a scheduler to manage and control experiments, a distributed monitoring system to gather system data, a visualization tool, and an evaluation tool to compute performance metrics from collected experiment data.

To give a holistic idea how an automated experimentation framework can ease network experimentation, we describe the way how experiments are usually done on a testbed at the example of studying the performance of TCP. For this purpose tools like `ttcp` or `iperf` are available. As a first step, source and destination nodes and a correspondent route are needed. Either the researcher has to manage that a route exists or it can be assumed that it is provided by a routing daemon. Subsequently, the selected tool is started on the destination node in server mode and afterwards on the source node in client mode. For this the researcher has to log in on each node via `ssh` and start the tool manually. During the experiment run, the output of the tool has to be logged for subsequent analysis. Therefore, the log files have to be copied to a particular machine. For a sound scientific analysis, the experiment has to be repeated with the same source-destination pair of nodes and as well with different source-destination pairs of nodes. The overall process of doing the study involves many pitfalls as well as huge labor. With increasing complexity of an experiment, each manual repetition is likely to get an individual touch, since sensitive issues like command timing can only be accurate to a certain degree. In practice, researchers often fall back to shell scripts to automate repetitious parts of an experiment, such as the execution of a command sequence. But usually such a script does not automate the entire workflow.

Concluding from this example, the usual steps of an experiment are as follows: i) Design the experiment, ii) Execute the experiment for a particular number of times and collect data, iii) Evaluate the resulting data to obtain performance metrics. The DES-TBMS framework for automated experimentation supports the researcher in all of these steps without imposing restrictions.

The remainder of the paper is structured as follows. In Section 2 we present the DES-Testbed at Freie Universität Berlin. DES-TMBS is introduced in Section 3 and evaluated in Section 4. In Section 5 we discuss related work. The paper finishes in Section 6 with some conclusions and future work.

2 DES-Testbed

The *Distributed Embedded Systems* work group testbed (DES-Testbed) is a hybrid wireless network located on the campus of the Freie Universität Berlin. Currently it consists of 70 *DES-Nodes* with future upgrade plans to a total of 100. It is hybrid in a way, that all DES-Nodes consist of a mesh router equipped with three IEEE 802.11a/b/g transceivers each and a MSB-A2 sensor node [4] in the same enclosure, thus forming an overlapping wireless mesh and wireless sensor network. The MSB-A2 sensor nodes are attached to the mesh routers via USB 2.0, delivering several advantages in maintaining and accessing the sensor network. Custom firmware images can be distributed on the mesh routers and automatically flashed on the sensor nodes. Sensor nodes in an unresponsive state can be reset via the mesh router. Current sensor values such as the temperature and humidity provided by the on-board SHT11 sensor can be retrieved on demand. The mesh routers can also act as environment simulators for the sensor nodes by setting the sensor values to trigger events on the sensor node. The DES-Nodes are deployed in an unplanned manner across several buildings on the campus, the network architecture is depicted in Figure 1.

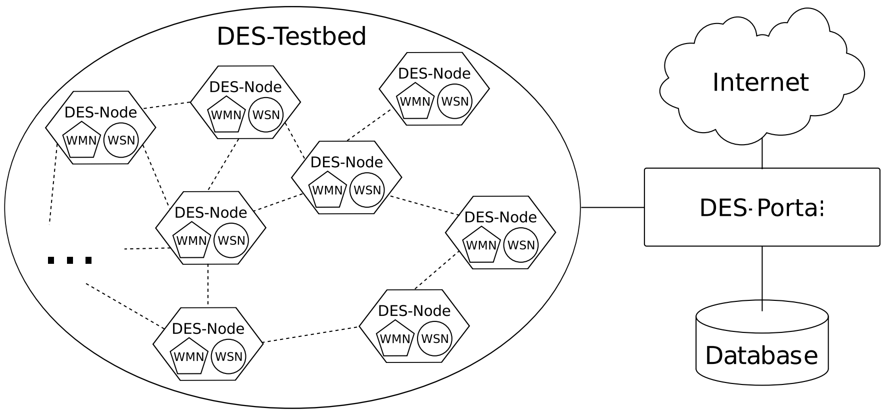


Fig. 1. Architecture of the DES-Testbed. Each DES-Node consists of a wireless mesh router and a wireless sensor node. Links are established between mesh routers and sensor nodes, the topology of both networks differs due to the different radio technology. The DES-Nodes are located in different buildings and are connected to the DES-Portal via Ethernet.

For accessing the DES-Nodes, a testbed server functions as the central control instance in the DES-Testbed. It is connected to all DES-Nodes via Ethernet and provides the databases used by the DES-TBMS components. Research on the DES-Testbed is focused on experimental comparisons of studies in simulation and real world network deployments. In order to increase the reliability of results, long-term experiments with a run-time of several months are supported. For a detailed technical description of the architecture of the DES-Testbed, we refer to [5] and [6].

3 DES-TBMS

The goal of DES-TBMS is to support the researcher to conduct studies of a large series of individual experiments on the DES-Testbed. It is not restricted to this particular testbed and has only few requirements, such as the Linux operating system and Java 6. The software of DES-TBMS is available under the terms of the GNU General Public License at the website of the DES-Testbed `http://www.des-testbed.net`.

3.1 Architecture of DES-TBMS

The architecture of DES-TBMS is shown in Figure 2. It consists of five components and four data bases, which have a tight relationship. *DES-Cript* is the domain specific language to define experiments, *DES-Exp* is the experiment manager which maintains all experiments stored in the *Experiment Description* data base and is responsible for scheduling, executing, and collecting the log files of an experiment. The raw results of experiments are stored in the *Result Data* data base. The testbed is pre- and post-configured by *DES-Mon*, which also monitors the testbed during experimentation. The monitoring data is stored in the *Monitoring Data* data base. Based on this data the status of the testbed is visualized by *DES-Vis*. The *Result Data* and *Monitoring Data* are used by *DES-Eval* in order to compute *Performance Metrics* for a particular experiment as defined in the experiment description.

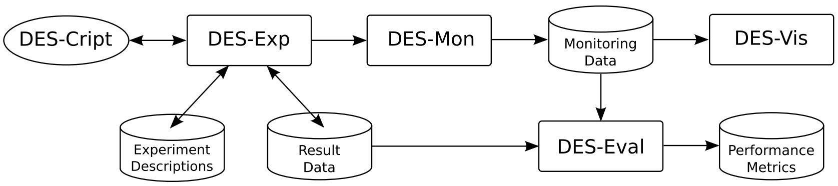


Fig. 2. Architecture of DES-TBMS comprising all components and their relationships

3.2 DES-Cript

DES-Cript refers to the developed domain specific language for network experiment descriptions. It is based on XML inheriting its easy-to-read property. DES-Cript features a clear and logical structure resembling the workflow of an experiment broken down into sequential tasks. By abstracting from the structure of the underlying network, DES-Cript is not limited to the DES-Testbed. The DES-Cript file merely specifies at what time and under what circumstances actions are carried out. The actual execution of the commands depends on the underlying network and is implemented in the DES-Exp software component.

Listing 1.1 shows a sample DES-Script file for a simple experiment with the purpose to measure the TCP performance according to the example given in the introduction.

Each DES-Cript file contains general information about an experiment, such as its name, the starting time, and number of iterations (see lines 3–5 in Listing 1.1). Next, available network nodes are assigned into groups with particular roles (lines 7–14). Currently, the defined roles are *client*, *server*, and *servent*. Thus, the user can design experiments for the classical client/server communication as well as for peer-to-peer communication.

In DES-Cript, actions are assigned to nodes in order to create an experiment, for example generating traffic flows. Basically, an action is just a command executed on the node. Therefore, actions are fully customizable by the user allowing a wide range of possible experiments. Actions can be assigned to a group or to individual nodes (lines 17–37). They can be scheduled sequentially, parallel, or at exact points of time relative to the starting time of the experiment. This enables the researcher to precisely determine and control the action schedule. Each of these actions can be related to an evaluation script which processes the generated output and produces performance metrics (lines 24 and 33).

Existing DES-Cript experiment descriptions can be edited and reused in order to modify or reschedule previously executed experiments. This can be very beneficial if experiment results suggest further refining of the experiment design to isolate critical parameters. Also, experiments can be exchanged and run on different testbeds supporting DES-Cript without adjusting the description.

In order to understand and interpret the results of manually executed experiments, every action has to be documented precisely. With DES-Cript this documentation is already created before the experiment is executed. The researcher relies on DES-Exp for carrying out all actions as specified. The experiment description is accessible at all times to investigate the results according to the initial experiment idea.

The DES-Cript experiment description format is applicable for the mesh and sensor network of the DES-Testbed. For instance, a long-term environment monitoring of temperature and humidity is implemented by retreiving these values periodically from the sensor nodes via the USB interface. The pause interval between successive iterations and the total number of iterations specify the resolution and overall runtime of the experiment.

```
<experiment>
  <general>
    <name>Simple TCP flow with iperf</name>
    <start_time>2009-03-19 20:00:00</start_time>
    <iterations>30</iterations>
     ...
    <groups>
      <group name="Receiver-TCP" role="Server">
        <members><node id="t9-150"></node></members>
      </group>
      <group name="Sender-TCP" role="Client">
        <members><node id="t9-155"></node></members>
      </group>
    </groups>
  </general>

  <actions>
    <action_block id="1" execution_mode="1">
      <action id="1">
        <group>Receiver-TCP</group>
        <command>iperf -s</command>
        <start_time>0</start_time>
        <duration>330</duration>
        <evaluation_script>iperf.py</evaluation_script>
      </action>
    </action_block>
    <action_block id="2" execution_mode="1">
      <action id="1">
        <group>Sender-TCP</group>
        <command>iperf -c t9-150-wlan0 -t300</command>
        <start_time>10</start_time>
        <duration>330</duration>
        <evaluation_script>iperf.py</evaluation_script>
      </action>
    </action_block>
     ...
  </actions>
</experiment>
```

Listing 1.1. Simplified DES-Script file for an experiment to measure the performance of TCP. It begins with general information such as the name of the experiment, its starting time, and the number of repetitions. Next, the participating nodes are assigned to roles and groups. Finally, commands comprised in action blocks are defined. An evaluation script can be assigned to each command.

3.3 DES-Exp

DES-Exp provides an experiment manager which handles the tasks to create, schedule, and execute experiments. The actual execution of an experiment is carried out automatically: The researcher specifies the preferred starting time and is notified via email after the execution when the results are available. Therefore, the execution of the experiment can not be influenced by the researchers interactions. This ensures a high degree of reproducibility in case the same experiment is repeated multiple times.

The functionality of DES-Exp is provided by two main applications. Firstly, a web application realized as a Java Servlet allows the user to create new experiments from the scratch or modify previously defined experiments. These experiments are scheduled considering the users preferred time. This enables to automatically execute experiments at night time, when fewer interferences caused by possible in-parallel operated networks are to be expected.

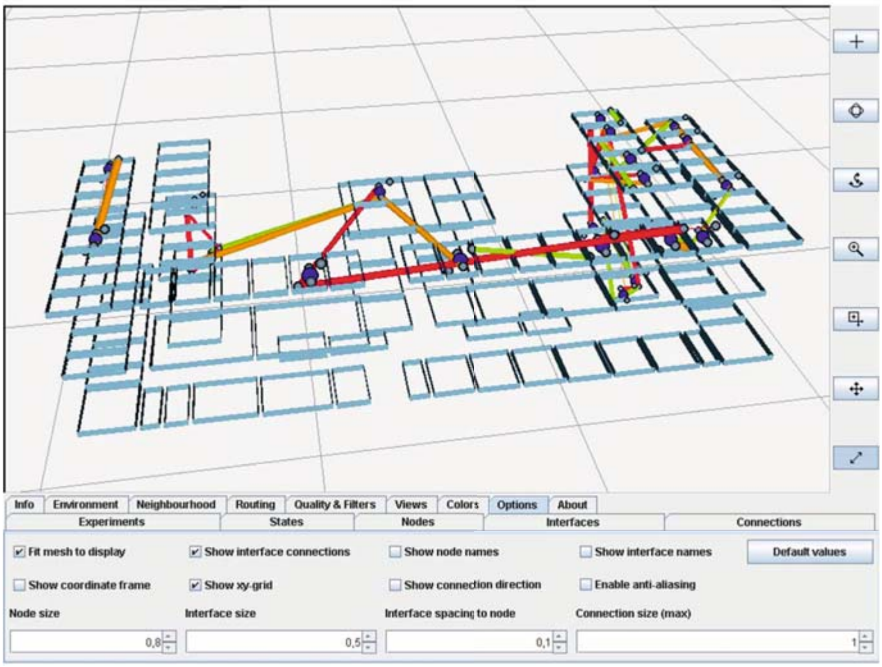
Secondly, an execution unit queries the database at regular intervals to check for any pending experiments. In case such an experiment exists, DES-Exp gathers the associated experiment description file and starts the execution. The execution unit uses an SSH-API for node communication. At the beginning of an experiment, SSH connections are established to all participating nodes, which are used throughout the experiment to configure the nodes and send commands. After the experiment run, all output files generated by action commands are gathered for further processing by *DES-Eval*. After the post-configuration of the testbed, it is released and further experiments can be carried out.

3.4 DES-Mon

The task of DES-Mon is to monitor the state of the testbed during the execution of an experiment. The state of the participating nodes in an experiment is collected by scanning these nodes in a fixed interval as specified in the experiment description file using SNMP. The gathered data of each scan consists of the mesh router state, the state of its WLAN interfaces and the Linux kernel routing table. The latter is used to visualize the connectivity during the experiment with DES-Vis. Using the SNMP-Set functionality, custom settings can be applied to the mesh routers in the configuration phase. DES-Mon is implemented as a multi-threaded Java application and uses the SNMP4J framework [7].

3.5 DES-Eval

The evaluation component DES-Eval parses the output generated by the actions of an experiment and inserts the formatted results into the *Performance Metrics* data base. Therefore, DES-Eval automates the tedious task of manually gathering all log files and extracting the important information for further analysis. It provides a suite of Python scripts for automatic evaluation of commonly used tools in networking experiments such as `ping`, `ttcp`, and `iperf`. In the experiment description, a particular evaluation script may be assigned to each action to process the generated output. DES-Eval runs the corresponding evaluation script, which processes the output file and provides the formatted results. Based on the total number of replications of an experiment statistical data like average, standard error, variance, and confidence intervals are calculated for the specified performance metrics. In a second step, further scripts can be used to generate standard graphs of the data.



[✓] Fit mesh to display [✓] Show interface connections [ ] Show node names [ ] Show interface names

[ ] Show coordinate frame [✓] Show xy-grid [ ] Show connection direction [ ] Enable anti-aliasing

Node size	Interface size	Interface spacing to node	Connection size (max)
0,8	0,5	0,1	1

Fig. 3. Screenshot of DES-Vis showing a snapshot of the connectivity of the network

For self written tools or shell scripts, we provide a template for an evaluation script, which specifies the database parameters and basic functions for processing the output. Proprietary evaluation scripts based on the template can be uploaded by the researcher and selected for processing the output of particular commands.

3.6 DES-Vis

DES-Vis is a 3D-visualization software based on the JavaView framework [8]. Its main purpose is to display the connectivity of the network during experiment runs. Figure 3 shows a screenshot of the connectivity of the wireless mesh network at a particular time. It relies on the kernel routing table information gathered during the network scans by DES-Mon. The changes in connectivity can be animated throughout an entire experiment run. For this, the corresponding network state snapshots are sorted according to their timestamp and progressed through them. This feature enables researchers to quickly discover explanations for obtained results and simplifies insight into observed phenomena.

3.7 Experiment Workflow

The interaction of the presented DES-TBMS components is depicted in Figure 4, with the workflow of an experiment broken down into sequential tasks. First,

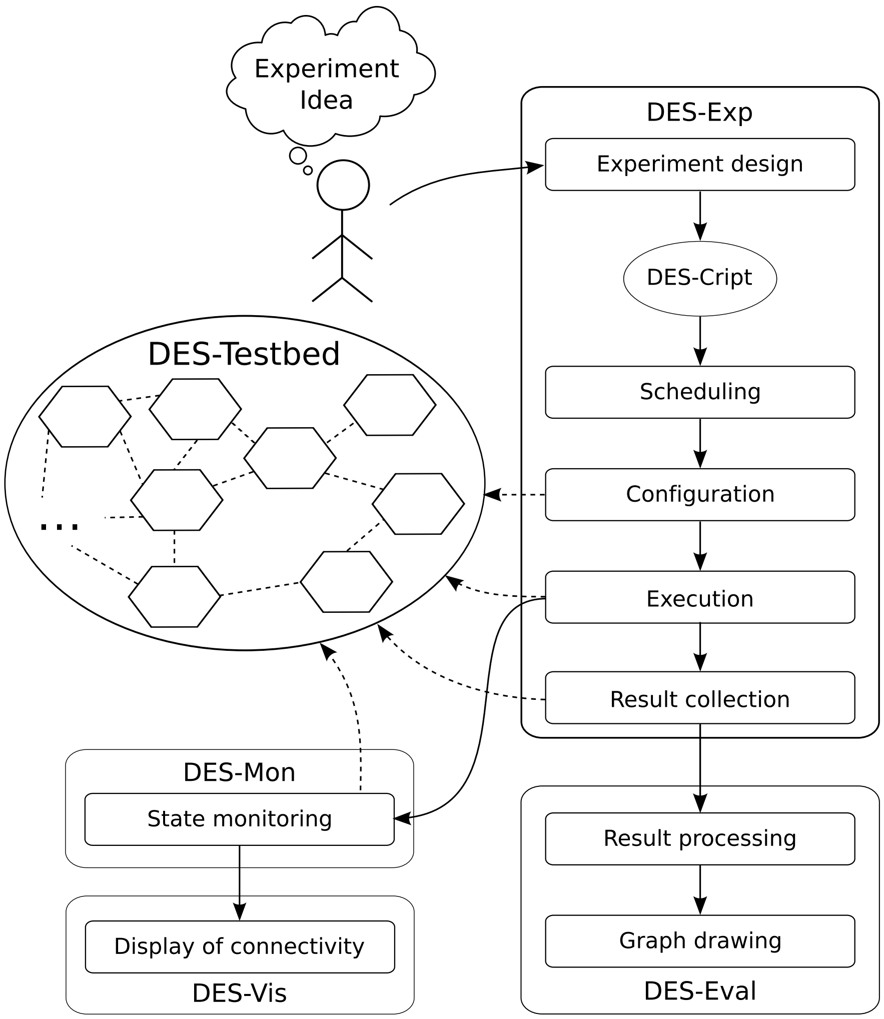


Fig. 4. The figure maps the sequential tasks of the network experiment workflow to the particular component of DES-TBMS

the user creates an experiment resulting in an DES-Cript experiment description. DES-Exp schedules the experiment at a specified date. Prior to the experiment execution, the testbed is configured according to the experiment description. Subsequently, the experiment is executed by sending the commands to the DES-Nodes at the specified times. Meanwhile the testbed state is monitored periodically by DES-Mon. The state data is preprocessed, inserted into the Monitoring Database and available afterwards for visualization. After the experiment execution, all resulting output files are copied from the DES-Nodes to the testbed server. Additionally, all data of an experiment are stored in a `tar` file and archived on the testbed server. The output files are evaluated by

DES-Eval and the results are inserted into the *Performance Metrics* database. Finally, DES-Exp notifies the user via email that the experiment has finished and results have been processed and made available.

4 Evaluation

4.1 Contributions of DES-TBMS

DES-TBMS enables the automation of experiments on testbeds to a high degree. Therefore, many pitfalls known in manual experimentation, such as using a slightly different action-timing of two consecutive runs of an experiment, are avoided. With DES-TBMS, actions can be triggered sequentially, parallel, and at fixed points of time. This allows to perform a wide range of experiments.

The scheduling service permits to run large series of an experiment. It also provides concurrent user access to the testbed. While some testbeds support multiple user access by assigning non-overlapping time slots to researchers, users can create and schedule their experiments concurrently with DES-TBMS. Conflicting experiments are automatically relayed to the next possible point of time.

With DES-Cript, a formal experiment description format is introduced. This enables researchers to conveniently communicate experiment ideas and make experiments comprehensible and reproducible for the research community. A suite of standard experiments applicable for a wide range of networks will be created with DES-Cript. Results obtained from these standard experiments can be used to compare the characteristics of different testbeds and networks.

Next to popular tools for network performance measurements, such as `iperf` and `nuttcp`, self-written scripts and tools can be used in custom experiments. For the standard tools, DES-TBMS provides evaluation scripts, which process their output, compute performance metrics and store these in the data base. A template is provided, which allows to write evaluation scripts for self-developed tools.

4.2 Limitations

In the current state of the DES-Testbed all mesh routers are connected via Ethernet to the testbed server. This reliable connection allows to call commands exactly at the specified times. It also ensures, that control messages generated by DES-TBMS are not sent over the wireless network, which otherwise might influence the experiment results. In future we plan to support mobile nodes as well. For this, either all control traffic is sent via the wireless interfaces, which will result in higher delays and may influence the experiment traffic. Another possibility is to copy the experiment descriptions to the mobile nodes prior to the execution and a *Node Manager* hosted on these nodes manages the experiment run. Higher latencies and delays for monitoring mobile nodes via the wireless interfaces have to be expected, which requires a further analysis of the scalability of DES-TBMS.

5 Related Work

In this section we review experiment environments in existing wireless mesh testbeds. We also consider simulation environments, since experimentation with simulators is more comfortable than it is in testbeds. Thus, many testbeds aim at providing a similar comfortable service as simulators do. In fact, one of the major design goals for DES-TBMS was to render network experimentation on a testbed as comfortable as it is in simulation environments.

Since each of the presented testbeds is tailored for a specific research focus, they differ vastly in network architecture, hardware, and software. Because of this, an in-depth comparison of these features is not meaningful. Therefore, we describe the motivation and set-up of each testbed and point out important and unique characteristics to provide a service needed for the particular studies. The results are summarized in Table 1.

Table 1. Overview of the discussed experimentation environments. This table shows the capabilities of the experimentation facilities and how researchers are supported in the experimentation process.

Property	APE	ORBIT	WiSEMesh	PlanetLab	Simulation	TBMS
Formal experiment description	•	•		•	•	•
Concurrent user access				•		•
Experiment repetition	•	•			•	•
Temporal command sequences	•	•			•	•
Monitoring	•	•	•		•	•
Evaluation	•	•	•			•
Network visualization	•		•		•	•
Archive						•
Real time inspection				•	•	
Mobility support	•				•	

5.1 Testbeds

The APE project [9] is one of the first ad-hoc mesh testbeds and was created to research and evaluate routing protocols for MANETs. Experiments are described in scenario files consisting of simple instructions regarding node movement, called the test choreography, and actions, e.g., traffic flows [10]. Multiple iterations of an experiment can be run using the same scenario files. A suite of tools exists for post-processing the results of an experiment by combining all local log files into one chronologically ordered one.

The ORBIT testbed [11] consists of a grid network of up to 400 wireless mesh routers. Experiments are described in a domain specific language and multiple iterations are possible by rescheduling them. An adjustable tool was created to monitor the testbed state. Applications, which can be run on the nodes, e.g., for creating data flows, are provided. Evaluation scripts post-process results and use Maple to visualize them. Experimentation methodology is addressed with the focus on repeatability and reproducibility [12]. For this, periodic calibration of the transceivers to normalize the transmitting power and receiving sensitivity is suggested.

The WiSEMesh testbed [13] consists of 56 mesh routers with the purpose to provide Internet access. A network management software called WiVi exists, which is used to configure the nodes and monitor the testbed state in an interval of one minute. A 2D-visualization tool displays the current network state. Experiments, which seriously affect the connectivity and availability are not desired and thus no experiment framework exists.

PlanetLab [14] is a wired testbed of about 1000 nodes connected via the Internet. It is dedicated to evaluate and develop Internet services and protocols. Users access and use a partition of the nodes, called a slice, either via SSH and execute experiments manually or via the PlanetLab experiment manager (PlMan). PlMan provides an XML-RPC interface, which enables experiments descriptions as Python or Perl scripts. PlanetLab declares reproducibility not as a goal, since hardware and software of nodes of a slice is not guaranteed to be the same at different access times.

5.2 Simulation Environments

The typical simulation environments used in wireless research are ns-2, ns-3, OmNeT++, OPNET/Modeler, and GlomoSim. They provide configuration means, either as a description language or via a GUI, to describe and run an experiment. Usually, this configuration is applied to only one simulation run. For a sequence of repetitions, the user has to implement a wrapper, which may be written in the same programming language as the configuration file. Some simulators offer the ability of real time inspection of ongoing experiments. This allows to inspect intermediary results and the network conditions, such as the size of packet queues on the network nodes.

6 Conclusions and Future Work

Experimentation is a fundamental part of research on networking studies and used to show proof of concept of new algorithms and to learn detailed characteristics of networks. However, carrying out large series of experiments in testbeds is labor-intensive and error-prone.

In this paper we discussed DES-TBMS for the automation of large experiment series in a distributed wireless testbed. The design and implementation of DES-TBMS is not limited to our testbed, but can be used on other testbeds, since the only requirements are Linux and Java.

We presented the architecture, components, and workflow imposed by DES-TBMS. However, we did not provide a rigorous performance evaluation of the framework. In future we will study particular types of experiments with DES-TBMS, including routing, localization, throughput, and real time applications. We will investigate the scalability of the framework components in terms of the number of DES-Nodes and the number of traffic flows transported in parallel. Furthermore, we will extend the testbed by supporting mobile nodes and increase the number of DES-Nodes to a total of 100.

References

1. Zimmermann, A., Günes, M., Wenig, M., Meis, U., Ritzerfeld, J.: How to study wireless mesh networks: A hybrid testbed approach. In: Proceedings 21st International Conference on Advanced Information Networking and Applications, pp. 853–860 (2007)
2. Cavin, D., Sasson, Y., Schiper, A.: On the accuracy of MANET simulators. In: Proceedings of the ACM international workshop of Principles of mobile computing, New York, USA, pp. 38–43 (2002)
3. Kurkowski, S., Camp, T., Colagrosso, M.: MANET simulation studies: the incredibles. SIGMOBILE Mob. Comput. Commun. Rev. 9(4), 50–61 (2005)
4. Baar, M., Will, H., Blywis, B., Liers, A., Wittenburg, G., Schiller, J.: The ScatterWeb MSB-A2 Platform for Wireless Sensor Networks. Freie Universität Berlin, Technical Report (2008)
5. Güneş, M., Blywis, B., Juraschek, F.: Concept and design of the hybrid distributed embedded systems testbed. Freie Universität Berlin, Technical Report TR-B-08-10 (2008)
6. Güneş, M., Blywis, B., Juraschek, F., Schmidt, P.: Practical issues of implementing a hybrid multi-nic wireless mesh-network. Freie Universität Berlin, Technical Report TR-B-08-11 (2008)
7. SNMP4j.org: The SNMP API for Java, `http://www.snmp4j.org/`
8. JavaView: JavaView - Interactive 3D geometry and visualization, `http://www.javaview.de/`
9. Lundgren, H., Lundberg, D., Nielsen, J., Nordström, E., Tschudin, C.: A large-scale testbed for reproducible ad hoc protocol evaluations. Technical Report (2001)
10. Nordström, E., Gunningberg, P., Lundgren, H.: A testbed and methodology for experimental evaluation of wireless mobile ad hoc networks. In: Proceedings 1st International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities, pp. 100–109 (2005)
11. Raychaudhuri, D., Seskar, I., Ott, M., Ganu, S., Ramachandran, K., Kremo, H., Siracusa, R., Liu, H., Singh, M.: Overview of the orbit radio grid testbed for evaluation of next-generation wireless network protocols. In: Proceedings IEEE Wireless Communications and Networking Conference, vol. 3, pp. 1664–1669 (2005)
12. Sachin Ganu, R.H., Kremo, H., Seskar, I.: Addressing repeatability in wireless experiments using orbit testbed. In: Proceedings of IEEE Tridentcom (2005)
13. Shrestha, S.L., Lee, J., Lee, A., Lee, K., Lee, J., Chong, S.: An open wireless mesh testbed architecture with data collection and software distribution platform. In: Proceedings 3rd International Conference on Testbeds and Research Infrastructure for the Development of Networks and Communities TridentCom, pp. 1–10 (2007)
14. Peterson, L., Bavier, A., Fiuczynski, M.E., Muir, S.: Experiences building PlanetLab. In: Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation, OSDI (2006)