2011

A Dual-Network Testbed for Wireless Sensor Applications

Jedrzej Kowalczuk
University of Nebraska-Lincoln, jkowalczuk2@unl.edu

Mehmet C. Vuran
University of Nebraska-Lincoln, mcvuran@cse.unl.edu

Lance C. Pérez
University of Nebraska-Lincoln

2011 IEEE Global Telecommunications Conference (GLOBECOM 2011)
doi: 10.1109/GLOCOM.2011.6134233

A Dual-Network Testbed for Wireless Sensor Applications

Jędrzej Kowalczuk*, Mehmet C. Vuran†, Lance C. Pérez*
*Department of Electrical Engineering,
†Department of Computer Science and Engineering
University of Nebraska-Lincoln
jkowalczuk2@unl.edu, mcvuran@cse.unl.edu, lperez@unl.edu

***Abstract*—A dual-network Cyber-physical Networking (CPN) testbed developed at the University of Nebraska-Lincoln is described. The CPN testbed consists of two geographically disparate wireless sensor networks connected by a traditional TCP/IP network and enables peer-to-peer communication between *each* sensor in the network. The functionality of the testbed is enhanced by a range of software tools that support remote programming and network monitoring, and real-time visualization of sensor data. The resulting architecture supports easy deployment and evaluation of applications for both traditional and interconnected wireless sensor networks. To demonstrate the features of this testbed, a novel ping application was developed and deployed as a proof-of-concept application for peer-to-peer communication in geographically separated wireless sensor networks. Experimental evaluations of the ping application yield insight into the communications overhead that can be expected in future applications of peer-to-peer interconnected sensor networks.**

I. Introduction

The performance evaluation of sensor network applications is a well-known challenge in the area of wireless sensor networks (WSNs). The iterative nature of sensor reprogramming, the lack of easy-to-use debugging tools, and the absence of sophisticated application development tools are a few of the factors that make the deployment and testing of wireless sensor systems challenging. Due to the difficulty of reprogramming the sensor nodes in a field-deployed sensor network, most developed WSN solutions, including the majority of communication protocols, have been evaluated in software-based simulation environments. Although simulations may be sufficient for functional verification of algorithms and protocols, it is often insufficient for evaluating realistic models of resource usage and inter-node communication processes [1].

Wireless sensor network testbeds represent an alternative to the evaluation of WSN solutions by incorporating both hardware and software components in the evaluation process. A typical testbed includes a network of real sensor nodes that are controlled and accessed through a software layer. The use of real sensor nodes preserves the impacts of radio communication and mote limitations whereas the software layer improves the usability of the testing environment and allows the automation of tasks such as data collection and monitoring of nodes. Yet, most of these studies are isolated from the existing infrastructure, namely the Internet.

Recently, the availability of 6LoWPAN [2] implementations for WSNs has created significant interest in connecting WSNs *to* the Internet and eventually, *inter*connecting separate WSNs *through* the Internet. The use of IP-based messaging in the communications link between the interconnected WSNs makes it impractical to use simulation tools for evaluating their performance. Thus, interconnected wireless sensor network testbeds are needed to evaluate the performance of this integration.

In this work, the design and implementation of the Cyber-physical Networking (CPN) Testbed at the University of Nebraska-Lincoln (UNL) is described. The CPN testbed is designed to link *each individual* mote in two geographically separated WSNs over a TCP/IP tunnel. The testbed is equipped with a range of software and hardware tools which support remote programming, out-of-band monitoring, power management, and real-time visualization and interaction. The distributed physical infrastructure of the testbed, i.e. geographical separation of the two testing sites, together with the software tools, allow for the development and evaluation of applications for both traditional and interconnected wireless sensor networks.

The remainder of the paper is organized as follows: In Section II, the related work on WSN testbeds and their interconnection are discussed. A taxonomy of WSN testbeds is presented in Section III. In Section IV, the hardware architecture and supporting software of the CPN testbed are described. The utility of the testbed is demonstrated with an experiment using a ping protocol in Section V and concluding remarks are provided in Section VI.

II. Related Work

In contrast to simulation tools, which have been previously classified in [3], [4], [5], a universal taxonomy of testbeds has not been established, largely due to the diversity of features these testbeds offer. Testbeds are typically assigned to one or more of the following classes: general-purpose or application-specific, indoor or outdoor, and mobile or stationary. Additional subclasses can be derived based on the individual features of the testbed, such as integration with IP networks or support for heterogeneous nodes. In the following, we describe several examples of testbeds that have preceded and influenced our testbed design.

MoteLab [6], a comprehensive testbed developed at Harvard University, is currently equipped with 190 permanently pow-

978-1-4244-9268-8/11/$26.00 ©2011 IEEE

ered TMote Sky motes interfaced with a central testbed server via a local Ethernet network. An essential element of MoteLab is a web interface that simplifies access to the testbed's infrastructure and guarantees fair resource sharing among users. The set of operations available through the web interface include mote programming, job creation and scheduling, viewing of output and debug information generated by completed jobs and a number of administrative actions. Moreover, MoteLab features a power-profiling subsystem capable of logging energy usage measurements of selected nodes for retrieval upon completion of the experiment.

Kansei [7], a high-fildelity sensing testbed designed and built at the Ohio State University, manages three arrays of sensor nodes. The largest stationary array contains 210 nodes arranged in a regular 2-dimensional grid. One interesting characteristic of the Kansei testbed is that each node in the stationary array incorporates an Extreme Scale Mote and a standalone Linux-based Stargate computing device that provides necessary communication ports. The Stargate units not only serve as local data collection and processing centers, but they also constitute a set of endpoints in an Ethernet backbone used to deliver control messages between nodes and a cluster of computers responsible for job scheduling, monitoring, and visualization tasks. The architecture of Kansei's software platform allows two ways of accessing testbed management functions: a web interface as a standard method of interaction with users and a group of web services that allow access from applications.

The TWIST testbed is a multi-level organization of devices that utilizes mixed communication interfaces [8]. The top level of the hardware organization consists of control stations and the primary server running a database engine and core network services. These components use an Ethernet back-channel to communicate with Linux-controlled devices referred to as super nodes. Each super node is connected to a USB hub that handles a collection of motes. The state of the TWIST testbed is controlled through a set of scripts residing in the filesystems of the super nodes. These scripts are invoked remotely from the control stations and are executed in a multi-threaded manner that speeds up actions such as mote reprogramming. The architecture of TWIST was subsequently adopted by the Washington University in St. Louis (WUSTL) WSN testbed [9].

There are a few examples of distributed testbeds that integrate geographically disparate sensor networks interconnected by conventional networks. The X-Sensor [10] testbed consists of eight disparate sites consolidated into one web-accessible testing environment. The web interface of the X-Sensor testbed provides a location-transparent method of accessing experimental sensor networks. Another example is the WISEBED project [11] that connects testbeds developed at several European universities. The research group associated with the project focuses on providing guidelines and application programming interfaces for creating networks of interconnected heterogeneous testbeds. While the two aforementioned testbeds make it possible to conduct large-scale interconnected sensor network experiments, they do not allow direct peer-to-peer communication between geographically separated individual sensors. The UNL Cyber-physical Networking testbed was built specifically to enable the study of applications that require inter-WSN communications.

III. A Taxonomy for WSN Testbeds

An analysis of existing WSN testbeds reveals a hierarchy of common features that provide different experimental capabilities. In an effort to provide a taxonomy of WSN testbeds, in the following, these features are discussed in a systematic manner.

1) *Scalability and reconfigurability:* The main preferred feature of a WSN testbed is its ability to be easily expanded and adapted to meet requirements caused by the growth in the research domain. Increasing the number of nodes in a scalable and reconfigurable testbed should require a very limited hardware adjustments with minimal impact on the software components.
2) *Heterogeneity:* The plethora of different platforms emerged in WSN research demands WSN testbeds to support heterogeneity, which allows the formation of autonomous non-uniform networks with the restriction that the motes within each network be homogeneous (testbed-level heterogeneity). Network-level heterogeneity and its application-level variant discard this restriction, allowing mixed mote types within the same network. Issues regarding the design and implementation of a heterogeneous testbed are discussed in [12]. An additional level of heterogeneity can be achieved by adding programmable robots to support limited mobility as in [13]. The presence of mobile nodes increases the complexity of the inter-node communication schemes and often requires node localization services.
3) *Robustness and controllability:* Mote programing and management are made possible by exchanging control commands between a management station and motes. A robust and controllable WSN testbed is capable of establishing a bi-directional communication link for the transfer of control messages between nodes and a management station at any time instance.
4) *Debugging, data logging and monitoring:* Debugging is the process of tracking the execution of sensor programs at selected nodes in order to identify erroneous fragments of code or to preview the values of key variables. Debugging mechanisms are necessary for the implementation of data logging tools capable of capturing, time stamping, and storing sequences of readings obtained by sensor nodes. Data logging is often accompanied by a monitoring routine - a periodic diagnostic operation verifying the availability of nodes or measuring levels of quality of critical services. Reporting energy consumption is an example of a monitoring task that requires the testbed to be instrumented with additional specialized equipment [6].

5) *Ease of access*. A common practice in WSN testbeds is the implementation of a web interface that provides the user with a set of graphical interfaces for accessing the functionality of the testbed. An advantage of web interfaces is that they centralize access to the testbed's resources in terms of both programming and monitoring motes.

Among these features, support for mobility is considered optional, and was not implemented in the CPN testbed. On the other hand, scalability and reconfigurability together with controllability and robustness were chosen to be the core design principles of the testbed. As will be described in Section IV, the feature set of the CPN testbed includes debugging and data logging tools and a web interface, which provides ease of access to the testing environment.

IV. Implementation of the CPN Testbed

In this section, the hardware infrastructure and the key software components of the CPN testbed are described. The hardware infrastructure delivers the physical foundation necessary to create, power and manage the WSN, and was designed to be extensible. The software components provide tools for the implementation and debugging of sensor applications, as well as the collection and visualization of sensor data.

A. Hardware Infrastructure

The hardware infrastructure of the CPN testbed is depicted in Figure 1. In this architecture, a *mote connector* is a fundamental building block, which is defined as a USB port connected to, and managed by, a server similar to [8]. The testbed consists of 92 numbered and labeled mote connectors distributed between two different buildings on the UNL campus; the Schorr Center has 45 connectors and Scott Engineering Center has 47 connectors. Both locations are controlled by designated servers running a distribution of the Linux operating system. Currently, the testbed supports the TelosB and IRIS platforms and the family of MICA motes.

The servers communicate with mote connectors through a number of intermediate devices. Handling a multitude of mote connectors requires the use of USB hubs. However, since there exists a strict limitation on passive USB cable length, the connections between connectors and hubs are realized with UTP skeletal wiring together with appropriate UTP-to-USB converters at both ends of each UTP segment. Ethernet wires eliminate the need for active USB cables or additional hubs as signal repeaters as long as the topology remains geographically consistent. Moreover, a single UTP line can handle up to 4 mote connectors and reduces the complexity of the cabling and connections.

A key element of the CPN testbed is the *device path*, which is illustrated on Figure 2. USB devices are organized in hierarchies, using specific USB ports of the controlling PC as root nodes and mote connectors as terminal nodes or leaves. It is important that each terminal node of the USB hierarchy tree is identified by a sequence of port numbers leading from a top-level USB hub to the node. Accordingly, each hub corresponds to a port number in the device path, which uniquely identifies a node in the testbed. To allow automated analysis and improve readability, segments of a device path are separated by a dot.

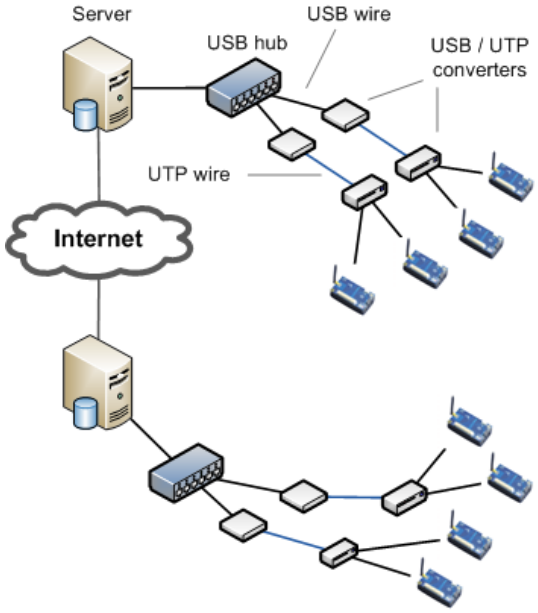

Figure 1. The hardware infrastructure of the CPN testbed.

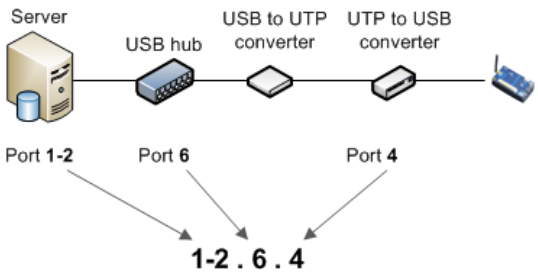

Figure 2. The concept of a device path in the CPN testbed.

B. Software Components

Based on the hardware infrastructure, several software components are developed to enable detection of physical types of motes, enable programming of groups of motes and to automate data collection. These components are described next.

1) Core Databases: The core software element at both locations is an instance of a relational database that holds information about the current state of the local testbed. The database stores a set of existing mote connectors and their attributes, i.e. associated device paths and spatial coordinates, as well as a list of devices attached to individual connectors. Lists of supported hardware platforms and lists of available motes, together with definitions of target hardware platforms for the compilation of sensor applications, are also part of the database. As described later, the database is also used to store the output collected from sensor applications.

2) Persistent Device Naming and Mote Type Detection Subsystem: Whenever a mote is attached to one of the available connectors, the `udev` device management tool recognizes this event and obtains the serial number of the connected mote together with the device path identifying the connector, and invokes a complementary script. The script then performs a database lookup to determine the type of the attached mote, and creates a record indicating that a connection has been made. Additionally, the script generates a persistent name for the device file corresponding to the connected mote, and this name is stored in the database. Analogous operations are performed upon removal of motes.

3) Application Deployment Subsystem: The build and deployment processes of nesC applications are controlled through the GNU Make tool. The default TinyOS Make system requires the programmer to execute at least one command in order to compile and upload an application to a single mote, which is impractical when the deployment needs to be performed on a large group of motes. The design of this system, however, makes it extensible and customizable, and enables end-users to define their own hardware platforms or supplemental rules for the Make tool.

We extended TinyOS's deployment tools to allow the easy and intuitive programming of groups of motes. Our development environment implements an additional command, `make group`, that handles the compilation and uploading of sensor applications as well as the assignment of node identifiers to user-specified sets of motes. These three fundamental operations (compilation, uploading, and identifier assignment) are performed regardless of the physical types of the target motes and their filesystem representations. The tool uses information stored in the database to translate the initial call to a set of generic TinyOS `make` commands and is capable of executing these commands in a multi-threaded fashion.

4) Debugging and Data Collection Subsystem: The debugging and data collection subsystem utilizes terminal printing functionality provided by the TinyOS `printf` library. Motes output debugging messages and sensor readings by simply printing to their serial ports. The user controls a set of debugger processes which can be attached to the serial ports of selected motes at any time instance in order to capture the output data. The data collected by the debugger processes is stored in the database to allow further processing.

5) Testbed Management Interface and Data Visualization Tools: Following the practice of wireless sensor network testbeds described in Section II, interaction between users and the subsystems of the testing framework occur through a web interface as shown in Figure 3. In addition to remote programming and data collection, the web interface offers a range of management tools that enable the user to monitor the status of connected devices and control the behavior of the deployment subsystem. Additionally, the web interface is integrated with a visualization API to enable the user to create application-specific interactive visualizations of the collected data.

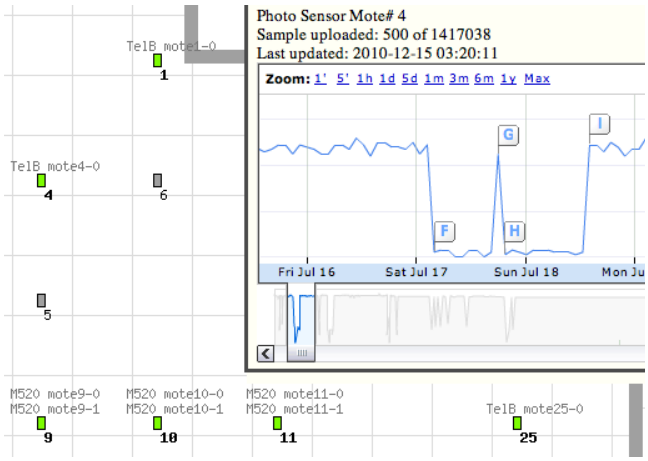

Figure 3. Interactive testbed preview with visualization of sensor data.

V. Performance Evaluation

As a proof-of-concept of peer-to-peer communication between geographically disparate sensor motes, a ping application was developed to assess the mean delay associated with communication between the two geographically disparate wireless sensor networks. The mean delay is a key performance metric for applications that require real-time event detection. An example of such an application is tracking over a very large geographic area where the network is divided into independently managed, interconnected sensor networks.

Nodes in each location were programmed to periodically test the reachability of randomly chosen nodes in the other location and to measure the latency in their responses. The source initiates the test by sending a specialized ping packet to the local base station node that forwards the request to the testbed server managing the source network. Then, the request is transferred over TCP/IP to the corresponding remote testbed server and delivered to the destination through the appropriate base station. The destination responds in a similar fashion. The experimental setup consisted of 20 motes (10 in each location plus a single base station mote) each of which served as both a source and a destination for ping packets. Each of the motes was programmed to generate one ping packet every 3000 ms.

To reflect the operation of a typical application for interconnected wireless sensor networks, the nodes in both networks were also instructed to perform intra-network messaging since local communication in such applications might still be carried out for local synchronization, protocol overhead, or data exchange. The intensity of intra-network communication is controlled by setting a desired mean inter-packet interval μ for all nodes in the network. Specifically, the time interval between two subsequent local communication events at a particular node is uniformly distributed over the interval $(0, 2\mu)$ ms. The resulting round-trip times of the ping packets and the amount of time spent on intra-network messaging, i.e., the time overhead associated with radio communication between motes and the local base stations, are shown in Figure 4. Each data point is an average of approximately 3000 observations.

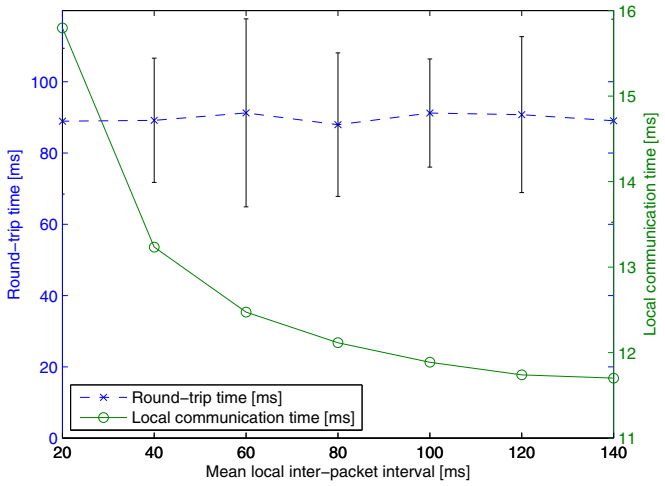

Figure 4. Round-trip times of inter-network ping packets and the delay for intra-network communication. Vertical lines show variability of round-trip times measured as the standard deviation from the mean round-trip time.

This experiment yielded an inter-network communication latency of approximately 90ms. Additionally, it can be observed that an increase in local inter-packet interval leads to an increase in latency for local communication. The overall inter-network communication time, however, is mainly determined by the latency associated with IP communication and is independent of the local network traffic. This experiment, which could not be performed via simulation due to the hybrid nature of the inter-network communications, provides insight into the communications overhead that can be expected in future applications of interconnected wireless sensor networks. The ping application and the capturing of communication latencies are possible due to the interconnected characteristic, the mote type detection capability, the ability to program groups of motes and the data collection functions of the CPN testbed. The implementation and the evaluation of the ping application serve as a proof-of-concept of peer-to-peer communication between geographically disparate sensor motes and illustrate the functionality of the CPN testbed.

VI. Conclusion

A distributed dual-network testbed for wireless sensor applications was described. The hierarchical organization of devices combining USB and Ethernet communication interfaces in the hardware infrastructure of the testbed, provides scalability and allows controlling servers to maintain robust, permanent connections with available motes. The corresponding software components installed on each of the servers automatically detect connected motes and are capable of determining their physical types. The testbed environment includes mote programming tools that allow one-click deployment of sensor applications to groups of motes. In addition, the testbed features a debugging and output collection subsystem that enables easy retrieval of sensor data. These functionalities, along with data visualization tools, are available through a web interface that simplifies and centralizes the interaction between the users and the testing environment.

The distributed architecture of the CPN testbed allowed for the development and evaluation of a ping application, through which the end-to-end delay associated with intra- and inter-network traffic was analyzed. Although delay analyses have been performed in traditional wireless sensor networks [14], the delay analysis in interconnected WSNs is much more challenging due to the presence of TCP/IP communications. Here, the problem of delay in interconnected WSNs was approached experimentally. Future work includes an analytical framework for this problem. The testbed can be used for the purposes of implementation, deployment and evaluation of applications and communication protocols for interconnected wireless sensor networks. Ultimately, such applications and protocols will contribute to the creation of the next generation Internet.

References

[1] I. Akyildiz, T. Melodia, and K. Chowdhury, "Wireless multimedia sensor networks: Applications and testbeds," *Proceedings of the IEEE*, vol. 96, pp. 1588 –1605, Oct. 2008.

[2] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks." RFC 4944, Sep. 2007.

[3] I. F. Akyildiz and M. C. Vuran, *Wireless Sensor Networks*. John Wiley & Sons Inc, Aug. 2010.

[4] M. Mekni and B. Moulin, "A survey on sensor webs simulation tools," in *Sensor Technologies and Applications, 2008. SENSORCOMM '08. Second International Conference on*, pp. 574 –579, Aug. 2008.

[5] M. Korkalainen, M. Sallinen, N. Karkkainen, and P. Tukeva, "Survey of wireless sensor networks simulation tools for demanding applications," in *Networking and Services, 2009. ICNS '09. Fifth International Conference on*, pp. 102 –106, Apr. 2009.

[6] G. Werner-Allen, P. Swieskowski, and M. Welsh, "Motelab: a wireless sensor network testbed," in *Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on*, pp. 483 – 488, Apr. 2005.

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

[8] V. Handziski, A. K. V. Handziski, A. Kopke, A. Willig, and A. Wolisz, "Twist: A scalable and reconfigurable testbed for wireless indoor experiments with sensor networks," *Proceedings of the 2nd International Workshop on Multi-hop Ad Hoc Networks: from Theory to Reality (RealMAN 2006)*, May 2006.

[9] WU WSN Research Group, "The WUSTL Wireless Sensor Network Testbed." http://www.cs.wustl.edu/wsn/index.php?title=Testbed, 2009.

[10] A. Kanzaki, T. Hara, Y. Ishi, N. Wakamiya, and S. Shimojo, "X-sensor: A sensor network testbed integrating multiple networks," in *Complex, Intelligent and Software Intensive Systems, 2009. CISIS '09. International Conference on*, pp. 1082 –1087, Mar. 2009.

[11] The WISEBED consortium, "WISEBED - Wireless Sensor Network Testbeds." http://www.wisebed.eu/, 2011.

[12] A. Reinhardt, M. Kropff, M. Hollick, and R. Steinmetz, "Designing a sensor network testbed for smart heterogeneous applications," in *Local Computer Networks, 2008. LCN 2008. 33rd IEEE Conference on*, pp. 715 –722, Oct. 2008.

[13] D. Johnson, T. Stack, R. Fish, D. M. Flickinger, L. Stoller, R. Ricci, and J. Lepreau, "Mobile emulab: A robotic wireless and sensor network testbed," in *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pp. 1 –12, Apr. 2006.

[14] Y. Wang, M. C. Vuran, and S. Goddard, "Cross-layer analysis of the end-to-end delay distribution in wireless sensor networks," *Networking, IEEE/ACM Transactions on*, vol. PP, no. 99, p. 1, 2011.