2011 Third International Conference on Communications and Mobile Computing

A Data Processing and Analysis Testbed for WSNs: Design and Implementation

Peng Cheng* Xianghui Cao* Jiming Chen* Kejie Cao* Youxian Sun* Xuemin (Sherman) Shen †
*State Key Lab. of Industrial Control Technology, Zhejiang University, Hangzhou 310027, China
† Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada

***Abstract*—In this paper, we design and implement a reliable data processing and analysis testbed (NESCTB) for wireless sensor networks (WSNs). It contains 36 wireless sensor nodes connected through Ethernet to a central server so that, with web interface, remote users are able to schedule tasks and monitor the nodes operations online. In addition, reprogramming and data logging can also be provided via the wired network. Powerful data processing and analysis tools are developed for information searching and processing, real-time topology visualization, 2-D and 3-D viewing of statistical diagrams, snapshots and timing sequences. The tools can significantly improve user-friendliness and operability of the testbed. The application scenarios of the testbed are discussed and a validation experiment for the referred application case NESCSurge demonstrates the convenience and effectiveness of NESCTB.**

***Index Terms*—WSNs; testbed; data processing and analysis; visualization**

I. Introduction

Wireless sensor networks (WSNs) are composed of a number of autonomous and compact devices, called sensor nodes, that have the capability of sensing, data processing, and communicating. The recent advances in fabrication, modern sensor and communication technologies have attracted substantial research and applications in WSNs [1], [2] and [3].

Considering the specific characteristics such as energy constraint and distributed deployment, many new network protocols and technologies have been proposed for WSNs [4]. Meanwhile, there are also various kinds of application oriented algorithms designed and developed for WSNs. It is well known that simulations are repeatable and scalable while experiments are quite effective to check the effectiveness under practical conditions. Thus both simulation and experiment are of great importance to verify the protocols and algorithms.

NS2 is a widely used simulation tool for beginners [5] while OMNET++ is more reliable and chosen by many advanced researchers [6], [7] and [8]. Additionally, TOSSIM and MATLAB are also useful software for simulation [9]. As the assumptions and models are just approximations for the practical conditions, simulations can not well handle imperfect radio communication, hardware interrupts, real PHY/MAC layer events, etc. Hence, it is necessary to employ physical experiments. Furthermore, it is possible to obtain more precise model for power consumption as well as packet timing. Meanwhile, the experiments are also useful for detecting the design limitations such as memory, bandwidth and reliability [10]. Nevertheless, it is often time-consuming for setting a specific large-scale experiment. As a result, it is of great interest to set a large-scale testbed which is applicable for different kinds of experimental requirements.

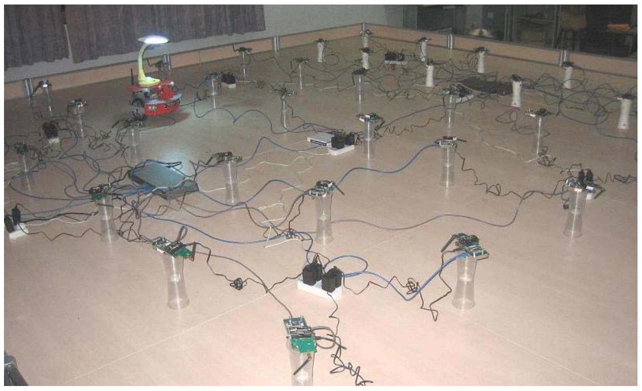


Fig. 1. Physical deployment of NESCTB in Zhejiang University

In this paper, as shown in Fig.1, we design and implement a wired testbed (NESCTB) for WSNs, which is aimed to conduct different general-purpose function tests. The testbed is deployed in the Group of Networked Sensing and Control (NESC)[11]. Currently, NESCTB has 36 nodes deployed in a 7m x 13m area in Zhejiang University, China.

Each node plugged with a sensor board and a MIB600 gateway is connected to the Ethernet for uploading/downloading the data in order to reduce the interference. And the on-line reprogramming provides great facility for experiment setting via MIB600. The testbed is also scalable due to the universal Ethernet interface. We may also deploy different nodes such as Imote2 for establishing a heterogeneous wireless sensor network testbed. Expedient software tools for data processing and analysis are also well developed for the NESCTB. The B/S 3-tier model is utilized to guarantee the safety of the system and the multiuser operation. Moreover, many statistical tools are also provided such as information searching, overall real-time linking graph, logic topology graph at a certain point of time, 2-D and 3-D statistical diagram, snapshots and timing sequences.

The remainder of the paper is organized as follows. Related work on testbeds for WSNs are presented in Section II. In Section III, the details of NESCTB including both hardware and software are depicted and analyzed. Section IV discusses the applications of NESCTB, especially for time synchronization test and routing protocol performance test. One validation experiment is shown in Section V. Section VI concludes the paper.

II. Related Works

In general, there are mainly two strategies for building wireless sensor network testbeds: domain testbeds and platform testbeds [12]. Domain testbeds focus on specific fields of WSNs applications (e.g. health monitoring), most of which are specially designed and fastened onto the particular applications. Platform testbeds pay more attention on the universality and versatility access of WSNs so that it may provide open and friendly interfaces for testing protocols or deploying WSN experiments. So far, a lot of specific contributions have been conducted for building testbed platforms, e.g., MoteLab from Harvard University and Kansei from Ohio State University.

MoteLab deployed in Maxwell Dworkin Lab [13] is the first in establishing the general concept of wireless sensor network testbed, which sets up the 3-tier architecture and open access design philosophy. It is comprised of remote user window, central network server and downstream sensor network, which aims to provide the public and reliable wireless sensor network access. The key technology of MoteLab is the integration of a set of open source tools such as Apache Web engine, MySQL database, PHP frontplane and Perl backplane. Multiple accesses are supported by MoteLab including remote web login, backend database searching and direct sensor node accessing.

Kansei is another well known testbed [14], developed for providing mid-ware for 2-tier network of ExScal program. Different from other Mote-like platforms, Kansei deploys a real heterogeneous sensor network. For scalability, Kansei integrates multiple sensor nodes in one downstream testbed network, i.e., static network composed of 210 XSMs, portable network used as a data collector in real environment, and mobile network representing moving wireless sensing objects. Another feature of Kansei is that its software platform director is combined with real and library experiment data for building a mixed simulation model of assigned objects.

There are several other testbeds, e.g., SignetLab from University of Padova, Italy [15] and WSNTB from National Tsing Hua University [16], which possess a layered architecture supporting interactive user interfaces and regular downstream sensor networks. SignetLab provides a set of management tools for testbed users to well perform management tasks, while it also guarantees the extendibility by utilizing the concept of function plugins. WSNTB constructs another kind of heterogeneous network testbed with different sensor nodes interconnected through standard interfaces and intelligent gateways equipped with an MCU.

III. Design and Implementation of NESCTB

NESCTB includes wireless sensor network among the nodes for running the testing program and wired testing network for requiring the data from nodes. Users can operate the testbed through the central server equipped with web interface. The complete architecture of the testbed system is depicted in Fig.2, where the light lines are the physical connections with cable while the bold lines are the information flow directions. It can be seen that all the sensor nodes are connected to the hub (Ethernet) via gateways and switches, meanwhile the central server is connected to the hub (Ethernet). The users can also connect to the hub with Internet.

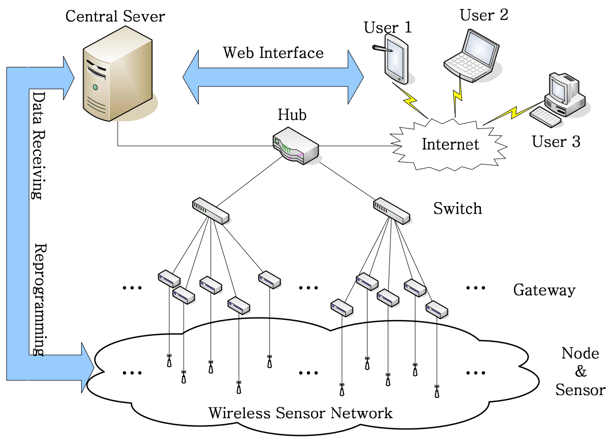


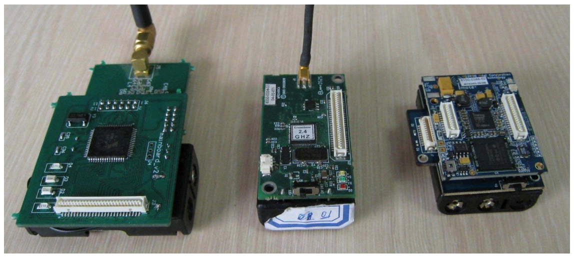
Fig. 2. Physical and information flow architecture of NESCTB

The whole framework of the testbed is shown in Fig.2. For the information flow directions, all the sensor nodes send the required data to the central server for storing, and the central server sends commands or reprogramming messages to the sensor nodes to accomplish the task. On the other hand, the user can send commands to the central server which retransmits the commands to the corresponding sensor nodes. Moreover, the user can also conduct data researching, data progressing and analysis through the web interface programmed by PHP.

A. Hardware Infrastructure

As the most important hardwares of wireless sensor networks, NESCMT and Micaz are selected as our sensor nodes, which have the same radio frequency of RF transceiver. As shown in Fig.III-A, NESCMT is a sensor node designed and developed by the Group of NESC[11], which consists of a 7.3MHz ATMega128L processor and a Chipcon CC2420 IEEE 802.15.4 compliant radios. In the future, we shall add Imote2 equipped with CC2420 so that the testbed can be extended to deal with experiments for heterogeneous sensor networks. In addition, as the software environment for Imote2 (Linux) is more powerful, it is expected the performance can also be enhanced.

As the gateway, Crossbow's product MIB600 plugged in sensor node with 51pin interface is connected to a 10 Base-T or 100 Base-T Ethernet with network interface. Due to the wired network equipped with MIB600, the online reprogramming and data processing can be conveniently realized, which reduces the time for setting up experiments and analyzing data.


(a) Sensor nodes: NESCMT(left), Micaz(middle), Imote2(right)

(b) Gateway: MIB600

Fig. 3. Hardwares.

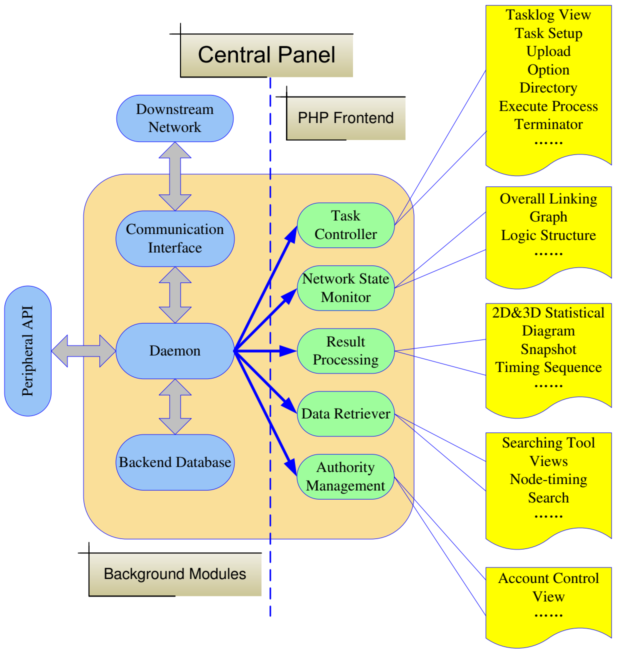
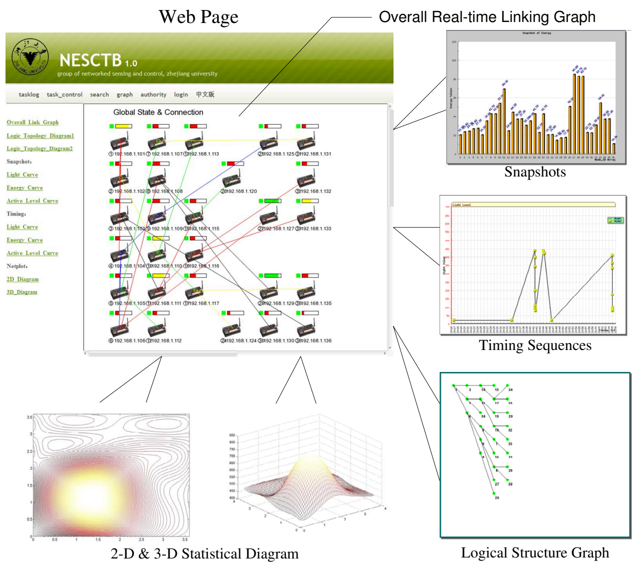


Fig. 4. The integral structure of the software



Web Page
Overall Real-time Linking Graph
NESCTB 1.0
tasklog task_control search graph authority login 中文版
Overall Link Graph
Logic Topology Diagram1
Logic Topology Diagram2
Global State & Connection
Snapshots
Timing Sequences
2-D & 3-D Statistical Diagram
Logical Structure Graph

Fig. 5. The main web and data processing tools

B. *Software Design*

The software platform of NESCTB is aimed for providing an universal user interface as well as a powerful controller that integrates separate hardware functionalities and unsystematic information scraps. In brief, the software section should possess the following features: (1) the ability to access each node embedded in downstream heterogeneous sensor networks, which performs as an overall monitor collecting feedbacks for debugging tasks; (2) an appropriate mechanism allowing for direct or indirect reprogramming process for sensor nodes; (3) a reliable data storage mechanism for guaranteeing the integrity of tasks; (4) real-time data retrieving functions under initiative process; (5) static data presented in a dynamic and organized manner or other customized ways. Furthermore, the testbed should be applicable for different environments.

The software platform with multiple original tools is well-formed and light-weighted as a whole. Due to the application of B/S 3-tier model instead of traditional 2-tier C/S model, NESCTB can be deployed on the most existing platforms, and be visited by accessing our Apache based controlling server via internet. No other client-side plugins is needed to execute the testbed tasks. We have also established 3 modules for server-side transactions: central panel as a controlling interface, MySQL database as the backend, peripheral C++ executables and system scripts as APIs, which have been shown in Fig.4.

1) Central Panel: The central panel is the main part for software platform, which is responsible for managing accessing services to databases or sensor nodes. It also provides a Web based graphic user interface. PHP scripts are used to generate dynamic Web pages, meanwhile Javascripts and Ajax techniques are adopted for constructing complex applications. After users login on the panel, they can select several modules: task controller, network state monitor, data retrieving, result processing views, and authority management, as shown in Fig.5.

The functions and relations of the modules are summarized as follows

- **Task Controller:** a module for users to select basic options and customized settings for debugging tasks, which include logging task history, customizing options, uploading executables programs or scripts, browsing the task directory, executing and ending the current object process.
- **Network State Monitor:** a module for collecting basic static information of downstream node networks. When the users select certain nodes, the interface would display a table of static information along with a map depicting locations and states of the corresponding sensor node.
- **Data Retrieving:** a module similar to a library management system which is designed for initial request for data from databases.
- **Result Processing Views:** the most featured module of software applications, dominated by piles of dynamically generated diagrams and statistic maps realized by utilizing PHP and Jpgraph graphical library.
- **Authority Management:** a module for account managements and future security transactions.

2) Data Processing and Graphical Interface: One of the most impressive features of NESCTB is the result processing module for data processing and visualization. Due to the complexity and heavy load of such data processing algorithms, previous testbeds, especially the platform strategic testbeds, seldom support these functions. There are only several application cases which have involved homologous statistic libraries.

NESCTB has established five data and network operation views in different manners, most of which are instantly generated based on the data models involved in MySQL database tables. When debugging objects are split into data pieces and rebuilt into a complete realworld-like data model in database, some of their statistics or logic features are distilled to construct a graph representing the origin object. The views are generally divided into two categories: real-time/snapshot and timing sequence. Some views listed below are also shown in Fig.5.

- **Overall Real-time Linking Graph**: When users make requests for real-time downstream state information, the corresponding view is drawn immediately in backend using Jpgraph graphical class, and then printed on screen or into PDF files reserved for further reuse if necessary. On this diagram, every static attributes (e.g., id, ip, location, status, etc.) and some requested dynamic attributes (e.g., link-nodes, parent-ids, etc.) are listed in a separate area of the canvas. A progress bar with colors is used to represent different energy levels, while a LED-like status mark, and ip strings are displayed around the corresponding node in a network map.
- **Logic Topology Graph**: When a protocol is running on NESCTB, attributes corresponding to each node and link (or route) are stored into tables named 'xx_node' and 'xx_link'. For example, when logic structures are needed for analyzing algorithms of routings, the request is accepted by pointing to a prepared Multi-tree followed with a traversal sequence. A strictly structured logic diagram is composed of common elements like ip and links. Directed lines link parent and child nodes, and the number of logic nodes involved can be as many as 36 at a time.
- **2-D and 3-D Statistical Diagram**: This is often used as a powerful result analyzer after running programs. It functions in two mechanisms: (1) the Jpgraph graphical library has an embedded statistical diagram interface (class), by which classes and objects are called; (2) peripheral API of MATLAB, which will be discussed below in detail.
- **Snapshots**: Snapshots are depicted by PHP and Jpgraph similarly as for the above graphs. They are responsible for network status at a certain point of time, which are realized by spreading node-specific information on x-axis of the diagram.
- **Timing Sequences**: They are also depicted by PHP and Jpgraph. The timing sequences focus on the variation with time during task slot, which spread timing-sequence-specific data on x-axis.

3) Daemon: NESCTB has built a C++ daemon to achieve better control of downstream networks, and the daemon is designed to support communication between server-side panel and sensor nodes via sockets and MIB600. As a background program, it should be turned on before and during task slots in order to answer any possible requests. When a message is generated by MIB600 from the nodes, the daemon responds immediately by pushing the message in a buffer, parsing its format, converting received message into a compatible format, and forwarding it to corresponding database. In reprogramming cases, when some control commands are appointed, Daemon achieves its targets by parsing, converting and forwarding, which is the reverse of the accessing process. In short, Daemon can be considered as a module of peripheral APIs, as they share the same standard of NESCTB APIs.

4) Peripheral API: NESCTB includes API which can briefly describe the interactive data and control command transmission between Central Panel and peripheral auxiliary executables or scripts. This module is designed to either make customized interface compatible or expand NESCTB's features and application scopes. In this way, we have defined a set of standards of interfaces embedded in NESCTB, and further developed some program routines utilizing such interfaces. Most of working loads are undertaken by Daemon module, which works as a converter between different data formats. As Daemon is developed as a part of API as well, the general process of manipulating such peripheral interrupts resembles that of communication between panel and downstream networks. In that case, the calling command format is *cmd_name [target_name$\cdots$] [num$\cdots$ location$\cdots$] [node_id$\cdots$] [-x$\cdots$]*, where *[num$\cdots$ location$\cdots$]* encloses number of involved nodes and their locations if needed, *[node_id$\cdots$]* encloses corresponding IDs of objects, and *[-x$\cdots$]* is an optional bracket where corresponding selected options from Central Panel are reflected. For instance, in data-collection process, the daemon is called in such format data2store *[task_name] [ip[1], ip[2], $\cdots$] [-time]*, and in 2-D and 3-D Statistical Diagram process, MATLAB library is called by *exmatpaint [nx, ny, x[1], x[2],$\cdots$, y[1], y[2],$\cdots$] [z[1][1], z[1][2],$\cdots$]*.

Although separate views are attached to Central Panel, some tentative ideas have been listed in the plan to further develop the usability and robustness of NESCTB software platform. To achieve real-time data retrieving, the process of data receiving and dispatching is still to be developed to resolve heavy time delay and data interference in the case of heavy load, which could cause significant errors for testing results especially in protocols emphasizing time synchronization. An initial 'push-and-pull' mechanism has been introduced to the controlling view of Central Panel by aiding data storing process. Under normal circumstances, downstream nodes send messages in an appropriate frequency (0.1s cycle time for example), while the frontend makes low-frequency inquiries daemon for data. In this sense, a daemon is an essential part responsible for sharing load and separating intermediate stage from PHP frontend.

IV. Applications of NESCTB

The NESCTB can be used for a number of research related to WSNs, ranging from data analysis, algorithm validating, localization and security issues, to wireless protocol evaluation. The platform is also powerful for the researchers to find more interesting problems. To be concrete, two typical testing applications are discussed as follows.

A. Time Synchronization Test

In order to reach a time synchronization within WSNs, a number of protocols have been developed, including most recent ones based on gradient clock [17] and consensus [18]. We can choose one of them as the test object to verify the protocol's mechanism and test its time-sync performance. The main obstacle of Time-sync test lies in obtaining legal time stamps of protocol, as well as computing the appropriate time synchronization indicators to assess the performance. Our nesC programs and corresponding application interface provide a convenient tool to cope with this difficulty.. Given the object of time-sync test, we can focus on the access to time attributes involved in sending and receiving time stamps, etc. Then real-time category data processing module is set to work. During the task time slot of time-sync test, NESCTB periodically calls the customized application interface, analyzes data, and obtains the variation of time deviation within runtime of TPSN, so that the time synchronization mechanism of the target protocol can be verified conveniently.

B. Routing Protocol Performance Test

The main performance indicators of WSN routing protocols include delay and energy efficiency [19]. For instance, regarding the energy efficiency, several energy-aware routing protocols can be chosen as our test object [20]. We can select the variation of energy levels as central attribute. In the following programming of application interface, Typical data attributes (such as residual energy, time stamp and path length) are prepared as the main object. After the experiment, the overall energy level curve and the diagram which depicts the relation between unbalanced and balance energy variations can be generated based on the recipient data and statistical graph library. Hence the performance test of the protocol can be achieved.

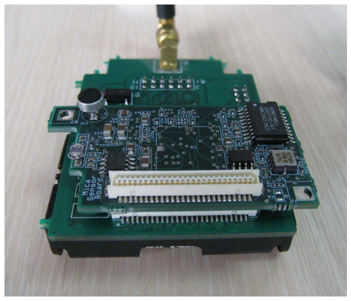
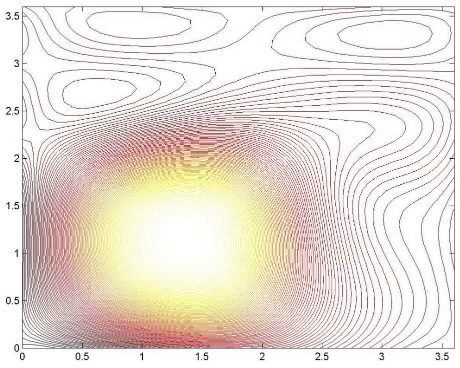


Fig. 6. The sensor board MTS300CA connected to the NESCMT

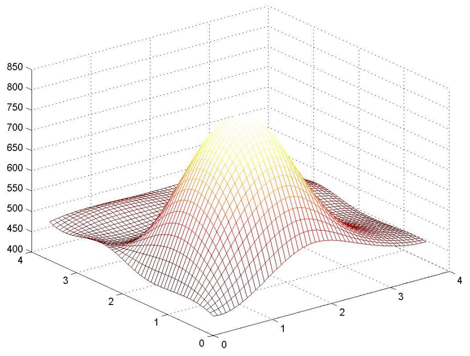
V. A Validation Experiment

Surge, a net compact framework of TinyOS[21], is a classical test program. According to the actual demands of our experiments, we revise the classical Surge to be our new NESCSurge which contains a new type of data unit and an extra kind of stable data-sending path. First, like the original Surge program, the nodes with NESCSurge would self-organize a multi-hop network, sense light intensity and send it to the server by choosing a route in the multi-hops network. Second, we add voltage message to the data unit, which means that the voltage level of each nodes could be live displayed in the server. It is without question that, when equipped with the proper sensors, we can get any kind of data requested, like humidity, magnetic field intensity, location information, etc. Third, despite the limits of the database node (actually in this primary experiment, we only use a common Micaz mote as the database node to get data units through RF way). Furthermore, we can exactly plot the topological structure of the multi-hop network with our new data unit and cooperating data gathering tools in spite of the data unit loss during the transition. In summary, most of the functions of the TinyOS are tested in our NESCSurge, including sending & receiving messages, reading the measured values, selecting route, and communicating with computers. Both the hardware and software tests have shown promising performance with NESCSurge.

In our experiments, the sensor board MTS300CA is connected to the node through the 51pin interface, as shown in Fig.6. All the sensor nodes are deployed in a $4.8m \times 4.8m$ area. They collect light intensity, send their messages to the sink node with wireless sensor network platform, and report to the MIB600 with the more reliable and precise wired Ethernet network. A lamp is placed on the laboratory table acting as the light source, as shown in Fig.1. The data of every node are gathered regularly and the values of light intensity in 2-D and 3-D maps are depicted by the data processing tools, as



(a) 2-D statistical diagram



(b) 3-D statistical diagram

Fig. 7. Illustrative diagram of the light intense

shown in Fig.7. It can be observed that the data gathered and depicted by our testbed correctly indicate the light intensity from the view of both 2-D and 3-D statistical diagrams.

VI. Conclusion and Future Work

In this paper, we have presented a new testbed, NESCTB for wireless sensor networks, deployed in Zhejiang University, China. The main contributions are summarized as follows:

- Physical sensor nodes are employed to obtain realistic and convincing data;
- Two networks are employed to enhance the performance: one is wireless sensor network for running the testing program while the other is wired testing network for querying data from nodes, which reduces the interference;
- Web interface is provided for multi-user operation and remote operation, including scheduling the job and monitoring the experiment online;
- Powerful data processing and analysis tools are developed for providing great convenience and facility for research.

Furthermore, NESCTB also provides an adaptable and practical platform for testing a number of difference experiments and algorithms. Our future work will focus on heterogeneous networks with different hardware and software systems such as Imote2. We also plan to complete a universal program library for frequently used tests such as time synchronization and protocol tests.

References

[1] I.F. Akyildiz, S. Weilian, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *IEEE Commununications Magazine*, 40:102–114, Aug. 2002.

[2] X.H. Cao, J.M. Chen, Y. Zhang, and Y.X. Sun. Development of an integrated wireless sensor network micro-environment monitoring system. *ISA Transaction (Elsevier)*, 47(3):247–255, 2008.

[3] F. Zhao and L. Guibas. Wireless sensor networks. *Communications Engineering Desk Reference*, page 247, 2009.

[4] W.L. Yeow, C.K. Tham, and W.C. Wong. Energy efficient medium access control protocols for wireless sensor networks and its state-of-art. *IEEE Trans. Vehicular Technology*, 56(2):918–928, Mar. 2007.

[5] Z.Y. Li and H.S. Shi. A data-aggregation algorithm based on adaptive ant colony system in wireless sensor networks. In *Proc. IEEE International Congress on Image and Signal Processing*, volume 4, pages 449–453, Sanya, China, May 2008.

[6] A. Varga. OMNeT++. *Modeling and Tools for Network Simulation*, pages 35–59, 2010.

[7] X.D. Xian, W.R. Shi, and H. Huang. Comparison of omnet++ and other simulator for wsn simulation. In *Proc. 3rd IEEE Conference on Industrial Electronics and Applications*, pages 1439 – 1443, Hyatt Hotel, Singapore, June 2008.

[8] J.H. Zhang, J.M. Chen, J.L. Fan, W.Q. Xu, and Y.X. Sun. Omnet++ based simulation for topology control in wireless sensor network: a case study. In *Proc. IEEE International Wireless Communications and Mobile Computing Conference*, pages 1130–1134, Crete, Greece, Aug. 2008.

[9] W.Y. Chun, E. Noel, and K.W. Tang. The tag duplication problem in an integrated wsn for rfid-based item-level inventory monitoring. In *Proc. 5th IEEE International Conference on Networked Sensing Systems*, pages 59–62, Kanazawa, Japan, June 2008.

[10] S. Bapat, W. Leal, T. Kwon, P. Wei, and A. Arora. Chowkidar: A health monitor for wireless sensor network testbeds. In *Proc. 3rd IEEE International Conference on Testbeds and Research Infrastructure for the Development of Networks and Communities*, pages 1–10, Orlando, Florida, May 2007.

[11] NESC. Homepage. http://www.sensornet.cn.

[12] A. Arora, E. Ertin, R. Ramnath, W. Leal, and M. Nesterenko. Kansei: A high-fidelity sensing testbed. *IEEE Internet Computing*, 10(2):35–47, March-April 2006.

[13] G. Werner-Allen, P. Swieskowski, and M. Welsh. Motelab: A wireless sensor network testbed. In *Proc. 4rd IEEE International Symposium on Information Processing in Sensor Networks*, pages 483–488, Los Angeles, California, USA, April 2005.

[14] E. Ertin, A. Arora, R. Ramnath, and M. Nesterenko. Kansei: A testbed for sensing at scale. In *Proc. 5th IEEE international conference on Information processing in sensor networks*, pages 399–406, Nashville, TN, USA, April 2006.

[15] R. Crepaldi, S. Friso, A. Harris, M. Mastrogiovanni, C. Petrioli, M. Rossi, A. Zanella, and M. Zorzi. The design, deployment, and analysis of signetlab: a sensor network testbed and interactive management tool. In *Proc. 3rd IEEE International Conference on Testbeds and Research Infrastructure for the Development of Networks and Communities*, pages 1–10, Orlando, Florida, May 2007.

[16] J.P. Sheu, C.J. Chang, C.Y. Sun, and W.K. Hu. Wsntb: A testbed for heterogeneous wireless sensor networks. In *Proc. 1st IEEE International Conference on Ubi-Media Computing*, pages 338–343, Lanzhou University, China, July 2008.

[17] Philipp Sommer and Roger Wattenhofer. Gradient clock synchronization in wireless sensor networks. In *Proc. International Conference on Information Processing in Sensor Networks*, IPSN '09, Washington, DC, USA, 2009.

[18] L. Schenato and F. Fiorentin. Average timesync: A consensus-based protocol for time synchronization in wireless sensor networks. In *Proc. 1st IFAC Workshop on Estimation and Control of Networked Systems (NecSys09)*, 2009.

[19] K. Akkaya and M. Younis. A survey on routing protocols for wireless sensor networks. *Ad Hoc Networks*, 3(3):325–349, 2005.

[20] M. Liu, J. Cao, G. Chen, and X. Wang. An energy-aware routing protocol in wireless sensor networks. *Sensors*, 9(1):445–462, 2009.

[21] TinyOS. An open-source operating system designed for low-power wireless devices. http://www.tinyos.net.