Implementation of Wireless Sensor Network Testbed - SRMSenseNet

M. Pushpalatha, SRM University
Revathi Venkataraman, SRM University
K. Sornalakshmi, SRM University
T. Ramarao, SRM University

ABSTRACT

As Wireless Sensor Networks have emerged as an exciting new area of research, testbeds have become the preferred basis for experimentation. In such testbeds, getting correct experimental result is essential. To address this need, we have developed a testbed termed as SRMSenseNet which consists of sensor nodes connected to a testbed server facilitating remote access to users. In this paper, we have proposed an easy data retrieval mechanism which helps to store the structured data results in the testbed. Additionally by providing web interfaces, our testbed allows the users to reserve their sensor motes. Users can experiment with real hardware resources and interact with our testbed in real-time.

Keywords

Wireless Sensor Networks, Testbeds, TelosB Motes, Reservation System.

1. INTRODUCTION

The research in wireless sensor networks has bloomed in recent years because of their potential applications in many areas, such as environmental monitoring, surveillance, disaster recovery and rescue. Wireless Sensor Network Testbeds (WSN-Testbeds) are the basis for experimentation with wireless sensor networks in real-world settings; and they are also used by many researchers to evaluate specific applications pertaining to specific areas. A WSN-Testbed typically consists of sensor nodes deployed in a controlled environment. WSN Testbed provides a platform for experimentation of large development projects. [1]

WSN-Testbeds are easily scalable, that is, increasing the number of motes should require only limited hardware and software adjustments. Users can test their algorithms or applications on different platforms such as TelosB, MicaZ, and Iris etc.

The setting up of different wired, wireless and wireless sensor network testbeds is growing in fast pace. Testbeds are being set up in many universities all over the world, few of them in collaboration with major organizations like Intel, Motorola etc. The testbeds studied include wireless sensor network testbeds (Kansei[2], Motelab[3], TWIST[4] and SensorNet[5], CitySense[6]), wireless testbeds (Kansei, Orbit, and Emulab[7]), and wired network testbeds (Emulab and Planetlab[8]).

This paper describes the architecture and features of the testbed framework developed in SRM University, India. Section II discusses the related testbed tools. Section III contains the proposed system and Section IV points to conclusion and future enhancements.

2. RELATED WORKS

MoteLab [1] is a very popular WSN testbed solution. It uses stargate board to provide an Ethernet back-channel to each sensor node in the network. The interaction between the users and the testbed is batch oriented and is controlled via a dynamic web interface supported by a back-end database. Motelab does not provide any facility to cancel a scheduled job. [9]

The Kansei testbed [2] is designed at The Ohio State University. It supports various wireless platforms such as Extreme Scale Motes (XSMs), TelosB, Imote2 and Stargates. The Startgates act as a local gateway for each kansei node in the network. Kansei's hardware infrastructure consists of three components: Stationary Array, Portable Array, and Mobile Array. Each node in the stationary array consists of two hardware platforms: Extreme Scale Motes (XSMs) and Stargates.

In our testbed, there is no need to have one interface board or Stargate for each "mote". Motes are connected to the microserver through USB interface making it cost effective than other testbeds in terms of hardware devices used.

SensorNet [5] is a testbed developed by Inter Berkeley Research. To solve the user disputation, they developed Mirage tool which applies microeconomic approaches to arbitrate among competing users. Users submit bids for bundle of resources using virtual currency. A combinatorial auction runs periodically to select the potential winners. The winner is the one who spend more currency on a particular resource. Mirage tool does not provide automated reprogramming of the nodes and data storage.

NetEye testbed [9] developed at Wayne State University is probably the first testbed to use the latest version of TinyOS 2.0.[10] whereas all other testbeds use TinyOS 1.1. Our Testbed uses the latest version TinyOS 2.1.1 which provides CC2420 security features and supports Iris and Shimmer platforms. The CC2420 in-line security implementations add two new interfaces to the CC2420 radio stack in TinyOS 2.1: CC2420SecurityMode and CC2420Keys. So the users can benefit from the advantages of this latest version in our testbed. New comers can also be motivated to use this testbed as TinyOS 2.1.1 is much easier to use than TinyOS 2.0 and TinyOS 1.1.

Tutornet [11] testbed is in Ronald Tutor Hall at University of Southern California. It consists of 13 clusters, with each cluster consisting of a stargate and several motes attached to it via USB cables. These stargates communicate with a central PC over 802.11b, from where any node on the testbed can be programmed. However it provides only raw packet data information as an experimental result.

SRMSenseNet addresses these challenges by providing a mechanism to get the structured experimental data results. Using MsgReader and port forwarding, the users can get a log as shown in Fig. 6.

3. SRMSenseNet ARCHITECTURE

SRMSenseNet architecture consists of three different levels.

1. Sensor nodes

2. Microservers

3. Server machine

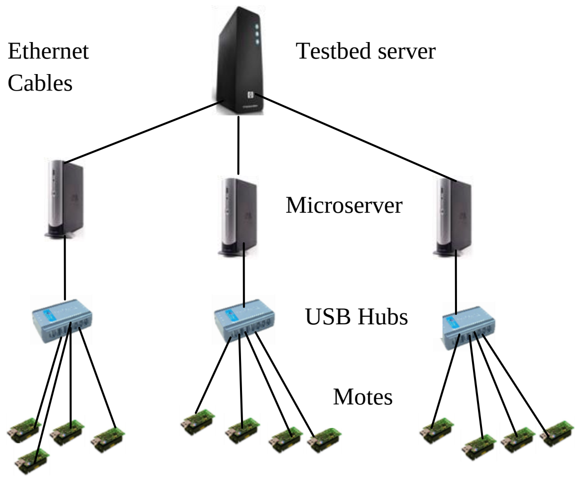


Figure 1. Testbed Architecture

At the lowest level sensor nodes (motes) are placed in order to take sensor readings or to perform certain functions based on the sensor application. These sensor nodes are connected to microservers at the second level through USB hub. Server Machine is placed at the third level which connects to all of the microservers over an Ethernet cables. The Server handles a database which contains information about the different sensor nodes and the microservers they are connected to. The server machine also provides an interface between the testbed and the end-users. Users may log onto the server to exchange messages with motes contained in the testbed.

3.1 Server Machine

Testbed server is a remotely accessible frame work for Wireless sensor network applications.

Testbed Server provides the following services.

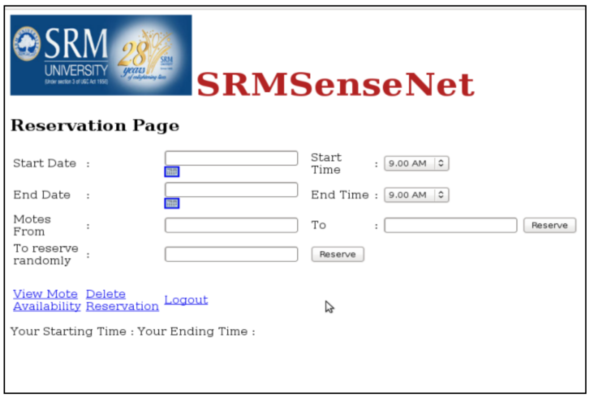
1. Web interface for registration of motes for specific time interval.

2. Resource Scheduling using MySQL database information.

3. Programming the nodes.

3.2 Web Interface

PHP [12] generated pages present a user interface for reservation of motes, checking mote availability at particular date and to delete the reservation. Users can reserve range of motes (eg.1-10) as well as random motes using space separated mote id (eg 1 5 8) as shown in Fig. 2.


SRMSenseNet

Reservation Page

Start Date :		Start Time	:	9.00 AM	
End Date :		End Time	:	9.00 AM	
Motes From :		To	:		Reserve
To reserve randomly :		Reserve			

Your Starting Time : Your Ending Time :

Figure 2. Snapshot of the Reservation System

At the end of reservation, users will get the Reservation ID. Users can delete the reservation by specifying the Reservation ID.

To find out all reservations within a time interval, click the Mote Availability Table and enter the date, then click on the "Check!" button. Users will get a table which displays the already reserved motes at the particular date. It will be helpful to the user to find out which motes are free in a time interval.

3.3 MySQL Database

MySQL database is used to store all the information necessary for running the testbed such as name of the user who reserved the nodes, start date and end date of reservations, time duration of reservations, status of each node and Reservation ID.

4. FEATURES OF SRMSENSENET

1. Remote Programming

2. Status Informations

3. Mote Informations

4. SerialForwarder

5. Execution Log

6. Alert to user

7. Easy Data Retrieval

4.1 Remote Programming

Deploying the motes into the physical environment and collect data from them is time-consuming. To avoid this, our WSN-Testbed allows users to remotely program the sensor motes and gather logs via internet. Only authorized users can access the testbed remotely.

4.2 Job Daemon

The Job Daemon is a python [13] script that runs as a cron job. The Job Daemon is responsible for changing the status of motes from reserved mode to free mode after the reserved time is completed. It also kills the processes like serial forwarder after the scheduled duration to free the resources.

4.3 Programming the motes

In order to program a mote, the files main.ihex and main.exe are needed. User has to move these two files to testbed server using scp command. After Logging into the testbed server, programming the nodes is done by shell scripting. The program running on the mote should have a receptor component to read the data from the serial port. The data is written to the serial port using the serial forwarder. The users have two options while programming the motes, that is, whether to start the SerialForwarder or not.

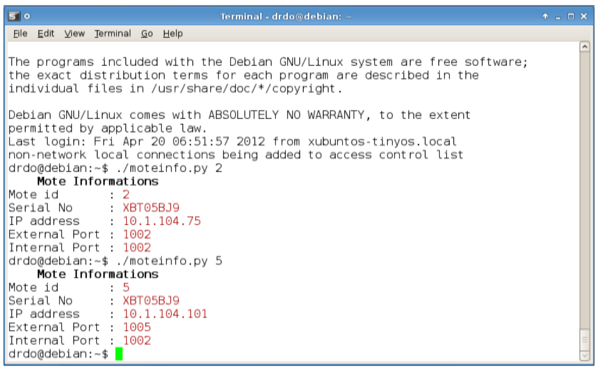
4.4 Status Information

During installation of the program, the process may be in any one of the following two states.

Done – Installation is completed.

Waiting – Installation is in progress.

4.5 Mote Information



```
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Fri Apr 20 06:51:57 2012 from xubuntos-tinyos.local
non-network local connections being added to access control list
drdo@debian:~$ ./moteinfo.py 2
    Mote Informations
Mote id       : 2
Serial No     : XBT05BJ9
IP address    : 10.1.104.75
External Port : 1002
Internal Port : 1002
drdo@debian:~$ ./moteinfo.py 5
    Mote Informations
Mote id       : 5
Serial No     : XBT05BJ9
IP address    : 10.1.104.101
External Port : 1005
Internal Port : 1002
drdo@debian:~$
```

Figure 3. Mote Information

Users can get information about each mote using its mote ID such as the micro server in which the particular mote is connected, Serial no of the mote, Internal Port number and External Port number as shown in Fig 3.

4.6 SerialForwarder

SerialForwarder program opens a packet source and lets many applications connect to it over a TCP/IP stream in order to use that source. Users can kill the specific SerialForwarder for a particular mote. The script running as a cron job will kill all the SerialForwarder after the scheduled duration.

4.7 Execution Log

Users can get their execution log file to check for possible errors during installation such as Timeout Error, Synchronization Error.

4.8 Alert to User

This is the unique feature of SRMSenseNet Testbed. While using the testbed, cron job script will give an alert message to the users, 15 minutes before the scheduled time gets over. After the specific interval cron job script will erase the entire program on motes to make it ready for further usage. So users have to save their output log during the reservation period.

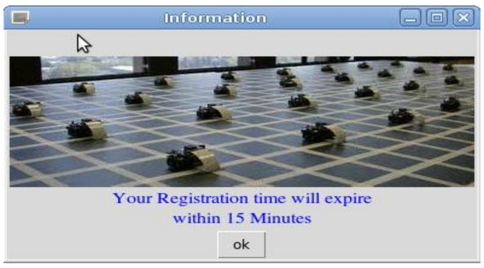
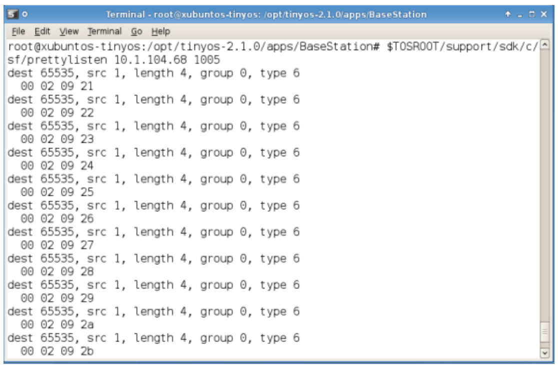


Figure 4. Alert Message

4.9 Data Retrieval

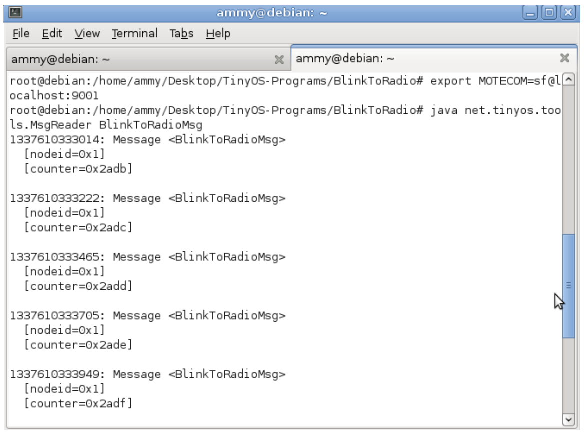
Our testbed allows users to connect directly to the SerialForwarder using its External port number. The External port number can be obtained from mote information by giving its moteid.



```
root@xubuntos-tinyos:/opt/tinyos-2.1.0/apps/BaseStation# $TOSROOT/support/sdk/c/
sf/prettylisten 10.1.104.68 1005
dest 65535, src 1, length 4, group 0, type 6
  00 02 09 21
dest 65535, src 1, length 4, group 0, type 6
  00 02 09 22
dest 65535, src 1, length 4, group 0, type 6
  00 02 09 23
dest 65535, src 1, length 4, group 0, type 6
  00 02 09 24
dest 65535, src 1, length 4, group 0, type 6
  00 02 09 25
dest 65535, src 1, length 4, group 0, type 6
  00 02 09 26
dest 65535, src 1, length 4, group 0, type 6
  00 02 09 27
dest 65535, src 1, length 4, group 0, type 6
  00 02 09 28
dest 65535, src 1, length 4, group 0, type 6
  00 02 09 29
dest 65535, src 1, length 4, group 0, type 6
  00 02 09 2a
dest 65535, src 1, length 4, group 0, type 6
  00 02 09 2b
```

Figure 5. Pretty listen output

Using prettylisten tool, users can store their output log in their system as shown in Fig 4. but it results the binary output of any packet it hears. User has to parse the packet format manually to extract the fields. Alternatively, the user can utilize the Message Interface Generator (MIG), which automatically parse each of the fields in the packet and it provides a set of standard accessors and mutators for printing out received packets.



```
root@debian:/home/ammy/Desktop/TinyOS-Programs/BlinkToRadio# export MOTECOM=sf@l
ocalhost:9001
root@debian:/home/ammy/Desktop/TinyOS-Programs/BlinkToRadio# java net.tinyos.too
ls.MsgReader BlinkToRadioMsg
1337610333014: Message <BlinkToRadioMsg>
  [nodeid=0x1]
  [counter=0x2adb]

1337610333222: Message <BlinkToRadioMsg>
  [nodeid=0x1]
  [counter=0x2adc]

1337610333465: Message <BlinkToRadioMsg>
  [nodeid=0x1]
  [counter=0x2add]

1337610333705: Message <BlinkToRadioMsg>
  [nodeid=0x1]
  [counter=0x2ade]

1337610333949: Message <BlinkToRadioMsg>
  [nodeid=0x1]
  [counter=0x2adf]
```

Figure 6. MsgReader output

Using the testbed, users can store the output log of MIG by forwarding the External port to local port of users as shown in Fig 5. SSH command to Forward SerialForwarder (SF),

```
ssh   -nNxTL localport:localport:e_port  testbed.srmuniv.ac.in
```

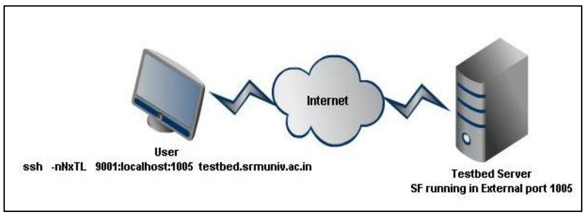

Internet
User
ssh -nNxTL 9001:localhost:1005 testbed.srmuniv.ac.in
Testbed Server
SF running in External port 1005

Figure 7. SSH Port Forwarding

Once forwarded the external port from testbed server to the local port of the user, users can access the remote SerialForwarder like a local one. Users has to make sure that it attaches to the correct port as specified in the SSH Tunnel (the above command forwards the remote port to your localport). For example, to listen packets from localport 9001, user would first need to set the MOTECOM environment variable as follows:

```
export MOTECOM=sf@localhost:9001
```

5. CONCLUSION

Using the testbed webinterface, users can reserve a set of motes for a specific period of time. Then users can install their own programs on the nodes. By using our testbed, researchers can save lot of time on building a WSN experiment environment and reduce the hardware cost, but at the same time, enhance the device utility rate, and quick verification of the experimental results.

In this paper, we have focused mainly on the software aspects of the testbed. Next hardware aspects have to be extended. If users want to run their application on a different hardware platforms or perform a mixed operation in a heterogeneous environment we need further support for writing of programs for these different hardware platforms.

6. REFERENCES

[1] Khalid El-Darymli,Mohamed H. Ahmed, "Wireless Sensor Network Testbeds: A Survey".

[2] Emre Ertin, Anish Arora, Rajiv Ramnath, Mikhali Nesterenko, Vinayak Naik, Sandip Bapat, Vinod Kulathumani, Mukundan Sridharan, Hongwei Zhang, Hui Cao, "Kansei: A Testbed for Sensing at Scale", in Proceedings of the 5th IEEE International Conference on Information Processing in Sensor Networks (ISPN 2006), April 2006.

[3] Geoffrey Werner-Allen, Matt Welsh, and Patrick Swieskowski "MoteLab: A Wireless Sensor Network Testbed", Division of Engineering and Applied Sciences, Harvard University.

[4] Vlado Handziski, Andreas K¨opke, Andreas Willig, Adam Wolisz, "TWIST: A Scalable and Reconfigurable Testbed for Wireless Indoor Experiments with Sensor Networks", Telecommunication Networks Group Technische University at Berlin, Germany, 2006.

[5] Intel Research Berkeley, "Mirage: Microeconomic Resource Allocation for SensorNet Testbeds", http://mirage.berkeley.intel-research.net

[6] Murty R.N "CitySense: An Urban-Scale Wireless Sensor Network and Test bed" , School of Eng. & Applied Sciences., Harvard University., Boston, MA,2008

[7] "Emulab – Network Emulation Testbed", http://www.emulab.net/

[8] Larry Peterson, Andy Bavier, Marc E. Fiuczynski, Steve Muir, "Experiences Building PlanetLab", in the proceedings of the 7 th conference on USENIX symposium on Operating Systems Design and Implementation (Volume 7), 2006.

[9] Divya Sakamuri, "NetEye: A Wireless Sensor Network Testbed Thesis", Wayne State University, Detroit, Michigan, 2008.

[10] "Tinyos Tutorial", http://www.tinyos.net/

[11] "Tutornet – A Tiered Wireless Sensor Network Testbed",

http://en.usc.edu/projects/tutornet/

[12] "PHP Tutorial", http://www.php.net/

[13] "Python Tutorial", http://docs.python.org/tutorial/