CHAPTER 35

ELEMENTS OF SENSORNET TESTBED DESIGN

Divya Sakamuri[✦], Hongwei Zhang[♦]

[✦]*Ford Motor Company, dsakamuri@gmail.com*
[♦]*Department of Computer Science, Wayne State University, hzhang@cs.wayne.edu*

Abstract

High fidelity experimental facilities play an important role in evaluating new sensornet protocols. In this chapter we will discuss the important elements of designing and implementing high fidelity sensornet testbeds, and we present in detail the elements of NetEye – a testbed with desirable features such as high-fidelity wireless environment, cost-effective hardware, web interface, and software tools to enable users to run potentially complex experiments. For instance, we will examine in detail the hardware details, software systems, and implementation (e.g., interference control) of NetEye.

1. Introduction

The research of wireless sensor networks (which we refer to as *sensornet*s hereafter) has become prosperous in recent years because of their potential applications in important domains, such as environmental monitoring, surveillance, and disaster recovery. Wireless, sensor network scenarios are expected to grow rapidly at the edge of the Internet. These systems will also be increasingly used in pervasive computing applications where the Internet enables monitoring and interaction with every aspect of the physical world. Over the past few decades, the Internet has evolved into a global network supporting a variety of computing and telecommunication applications. In the future the Internet must also respond to the many emerging requirements such as those posed by wireless sensor networks.

In order to help build the next generation Internet which will include wireless and sensor network devices, researchers need a vehicle to drive their next ideas. Researchers are investigating next generation network architecture and protocols, but they need a facility to evaluate them. The evaluation can be done using analytical modeling, simulation or high-fidelity measurements.

Analytical modeling provides insights into the effects of various parameters and their interactions. In spite of its flexibility, analytical modeling is used somewhat less often in sensornet research compared to simulation or measurements. This is partly due to the complexity and tractability of analytical modeling. Another disadvantage of analytical modeling is its often simplified mathematical assumptions which may not capture all the complex behaviors of sensornets.

Simulation is a replication of reality. When applied to the study of research design, simulations can serve as a suitable substitute for constructing and understanding field research. Simulations are easier to set up than physical experiments, are easy to repeat and modify, and are highly portable. Nonetheless, most existing simulators are unable to model many essential characteristics of the "real world." because of their simplifying assumptions. Therefore, Simulations with simplified assumptions may not produce accurate results.

In view of the limitations of simulated environments, accordingly, wireless and sensor network experimental facilities are very important to the research community. When building high-fidelity experimental facilities, the following characteristics are desirable:

- Realistic environments which are not too sterile;
- Simple programming and experimental interface;
- Flexible, comprehensive experiment control;
- Scale in terms of devices;
- Mix of devices providing heterogeneity.

In the following sections we will discuss in detail the design issues, the architecture, and the implementation of sensornet testbeds that embody the above characteristics.

2. Design issues for wireless and sensor network testbeds

To meet the goals of any deployment, proper design is the key factor. The testbed design should be done in such a way that the users can rely on the results of the facility. To obtain accurate results, the testbed should provide a controlled, realistic environment. Moreover, the testbed should be easy to use.

The deployment of a testbed with self-forming networks of wireless devices and sensor network devices requires a lot of design issues to be considered, among which are the following issues:

- Interference control
- Topology control
- Resource management
- Health monitoring necessity and requirements
- Provision of user tools
- Provision of administrative tools

2.1. *Interference control*

For testbeds to accurately generate results, an interference controlled environment is necessary. The wireless network devices are referred to by IEEE as 802.11 devices and the wireless sensor network devices as 802.15.4. One of the important concerns when building a wireless sensor network testbed is its

coexistence with other wireless networks. Coexistence can be said to be “the ability of one system to perform a certain task in a given shared environment where other systems may or may not be using same set of rules”. There will be many scenarios when different radios will operate in the same place and same time explosively growing the Industrial, Scientific and Medical (ISM) band usage. 802.15.4 radios share the 2.4 GHz ISB band with lots of other wireless devices like 802.11, Bluetooth headsets, 2.4 GHz cordless phones, microwaves, etc.

In some testbeds, deployment of both wireless and wireless sensor networks may be necessary. In such a situation 802.15.4 networks and 802.11 networks should operate simultaneously. Although the unlicensed ISB bands do not require strict coordination between the deployed devices, they do not permit all forms of behavior between the devices. The devices might interfere with each other and there could be loss of data and performance degradation.

To understand the potential problem, the available channels and RF spectrums for the 802.11 and 802.15.4 standards are shown in the following table and figure.

	IEEE 802.11		IEEE 802.15.4	
	Ch.	Freq. (GHz)	Ch.	Freq. (GHz)
2.4 GHz ISM Band	1	2.401- 2.423	11	2.405
	2	2.404- 2.428	12	2.410
	3	2.411- 2.433	13	2.415
	4	2.416- 2.438	14	2.420
	5	2.421- 2.443	15	2.425
	6	2.426- 2.448	16	2.430
	7	2.431- 2.453	17	2.435
	8	2.436- 2.458	18	2.440
	9	2.441- 2.463	19	2.445
	10	2.446- 2.468	20	2.450
	11	2.451- 2.473	21	2.455
			22	2.460
			23	2.465
			24	2.470
			25	2.475
			26	2.480

Table 2.1: 2.4GHz ISM Band (IEEE 802.15.4 and IEEE 802.11 channels)

There are 11 channels in 802.11 with only 3 non-overlapping channels. Each channel has a 22 MHz frequency range. There are 16 channels in 802.15.4 with a 3 MHz frequency range and each channel is 5 MHz apart.

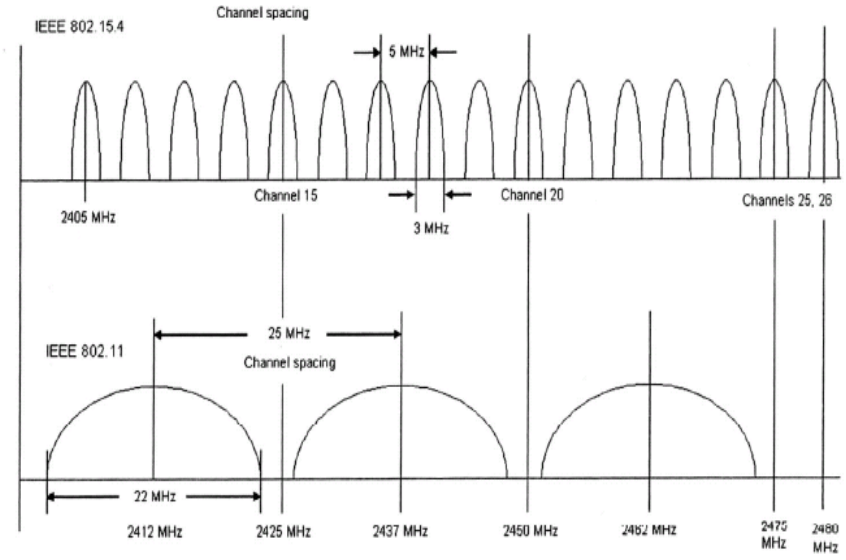


Figure 2.1: RF channel spectrum of IEEE 801.15.4/ZigBee and IEEE 802.11b / WiFi

Since the RF channels in 802.11 and 802.15.4 overlap there is a cause for concern. Even the beacon packets from the 802.11 access point cause considerable interference. To reduce the amount of interference from 802.11 devices, a channel on 802.15.4 that does not interfere with any 802.11 channel should be used.
From Figure 2.1, we can observe the RF spectrum of channels 15-24 of 802.15.4 overlap with the channels 1-11 of 802.11. This shows that channels 25 and 26 in the 802.15.4 standard are interference free from any channels of the 802.11.

2.2. *Topology control*

A few of the important topology design considerations are to choose which place to use, which equipment to use, how the devices should be placed on the chosen equipment, etc.

Once the sensor and wireless devices are chosen according to the requirements the physical and radio properties of the devices should be studied. The equipment to place these devices should be selected appropriately. For example wood can be chosen to place the devices as it won't conduct. The sensor devices will have large transmission ranges indoors. Experiments should be done appropriately to set the transmission range to form a multi-hop network. Attenuator can be used to reduce the communications over large range.

2.3. *Resource Management*

Resource allocation and management is a key factor for any shared computing and communication infrastructure. Resource allocation should provide fairness, efficiency, and reasonable performance for each scheduled task while honoring federated control. The challenge is to schedule all the tasks contending to use the nodes at the particular time. Some services using the testbed need guaranteed allocation of resources but some services may need just best-effort service model.

The shared testbed can either perform strict space sharing where individual jobs control a subset of nodes or best-effort scheduling where individual jobs receive resources inversely proportional to the number of currently competing jobs. The amount of resources that can be reserved by an application, the relative priority of best-effort applications, and the proportion of global resources subject to admission control are all matters of the resource allocation policy implemented in a particular testbed.

One of the important tasks of the resource allocation mechanism implemented is to allow efficient usage of the resources available. It should be ensured that the users only acquire only as much resources as they truly need and for only the period of time they really need them. Renewing of the resource privileges every few minutes should be implemented to ensure the resources usage efficiently.

Resource allocation algorithms implemented in the shared system, like a testbed should satisfy the following requirements:

a) Heterogeneous user base

The testbed can be used for a simple network experiment or a complex system architecture research. The resource allocation algorithm should provide some baseline guarantee of the resource availability to both kinds of users.

b) Policy Implementation

The resource allocation should implement a appropriate policy to guarantee resources to the individual users and applications. The policy can either share equal resources for all the users or allocate more resources to certain type of users. If the first type is implemented then many of the resources will be wasted if the resources are not being used by them in a given particular time. The second method lacks fairness; a normal user may or may not get the resources he wants to use.

c) Resource Isolation

Once the policy mentioned in the above section is implemented, there should be resource isolation. Virtualization is one technique that can be used to provide isolation. Resource allocation and isolation are enforced on a per node basis.

d) Security and Authentication

Security is a primary requirement for any resource allocation infrastructure. Users must have a secure mechanism for authenticating themselves to the system and for receiving the resources that they are entitled to. Users should not be allowed to consume more than their allocated share of resources denying other users from using the system.

e) Separation of Policy and Mechanism

Following two kinds of policies, best effort or guaranteed allocation, there should not be any fixed constraint on which user gets which resources, priorities of the jobs, etc. Basic mandatory classes of resource principles and the capabilities types should be defined which will take care that the resources are allocated within the authorized usage limits.

f) Providing Proper Incentives

The enabled resource allocation policies should ensure user commitment to the continued maintenance and growth of the infrastructure and promote user consumption of only as many resources as they require, especially during times of peak demand.

2.4. *Health Monitoring Necessity and Requirements*

A health monitoring service should be considered important while designing a wireless and sensor network testbed. The testbed will contain lots of hardware devices. It is natural for faults to occur in the middle of the experiment affecting the quality of the data. The fault can be due to the relatively unreliable wireless sensor network devices, software faults, or incorrect configuration of the devices. The faults can be categorized as follows:

a) Device Failure

Device failure, which the name itself suggests may be due to hardware failure or software fault that fails the device completely. There will be many hardware devices that might fail. The failure might be in the wireless sensor device, wireless device, etc. It depends on the deployed devices in the particular testbed.

b) Software Defects

Software defects and errors might corrupt a device or a network interface. This might be due to a bug in the code or using wrongly configured software.

c) Fault on the Network Interface

Communication between the devices might also fail due to driver failure, an unplugged Ethernet cable, detached cards, etc. A situation where all the network interfaces fail, resulting in a detached forest of devices, is also a possible. These faults are important to be diagnosed for both the users and the administrators of the testbed.

There are several important requirements to be considered while developing a health monitoring service since both the administrators and users use this tool. The requirements are as follows:

a) Correctness and Reliability

The monitoring results obtained should be correct. Incorrect data leads to misuse of the testbed facility. The status of each and every resource in the network should be reliably obtained. The monitoring service should be reliable in the manner that it should not be affected by the experiments scheduled on the testbed or the communication procedures. If this happens then the monitoring results might be incorrect.

b) Efficiency

The monitoring application should be efficient. This means it should not consume lots of time and lots of testbed resources. It should be efficient enough to monitor using less time, few messages, and fewer resources.

c) Independent and Adaptable

A testbed runs with lots of hardware devices. Hardware devices do not last forever; they might be replaced due to several reasons. The health monitoring code should be independent of all such changes and should adapt to any replacements on the testbed. It should be adaptable even when the network devices, interfaces, and the configuration changes with very little effort.

d) Availability

Monitored results should be available to the users and administrators in real time. To accommodate this requirement, health monitoring code should be automated to run periodically or it should be flexible to even run on demand. The results should be displayed on the GUI for the users and the administrators.

e) Handling Heterogeneity

A wireless and wireless sensor network testbeds may have many different devices with different capabilities and limitations. The health monitoring application should be capable enough to monitor the status of different hardware components like wireless devices, sensor network devices, Ethernets, embedded Linux systems, etc.

2.5. *Provision of user tools*

The kind of user tools to be provided to the user using the testbed is another important design decision to be made. The tools should be very user friendly, flexible for the use; and provide the user with full control on his experiment. A graphical user interface should be provided to each user for easy access and control of his experiments. The interface should also be remotely available. Ideally, the sensor network testbed GUI should have the necessary tools to perform the following operations:

a) Tool for registering as a new user

New users should be able to register into the testbed for doing their experiments. The users should be given a form to fill in and the administrator should check the accuracy of the information. Once the validation of the user information is done, he should be given access to the testbed with appropriate privileges. This method provides an easy way for both the users and administrators.

b) Tool for scheduling an experiment

The user, once provided authentication, should be able to schedule the nodes for his experiment. The nodes should be assigned according to the resource allocation algorithm implemented in the testbed. The algorithm can either be best-effort access or guaranteed access to the nodes. The user can select the start time for his experiment and also the duration to run his program.

c) Tool for knowing the job status

The user will always want to know the status of his scheduled experiment. The status should be communicated to the user through a GUI interface. This increases the user trust on the deployed testbed. Also implementing the necessary security mechanisms (the user should view only his jobs) is appropriate.

d) Tool for canceling the experiment

The testbed design should be done in such a way that it should save the resources whenever possible. There should be a provision to cancel the scheduled experiment at any moment. This saves lots of resources if the user has scheduled a wrong experiment, or on the wrong nodes that he does not want to use. Another advantage is if the user has specified a wrong duration, say 100 minutes instead of 10 minutes, it is not a good idea to wait for 100 minutes when the user really does not want to.

e) Tool for injection of real time data

The user might want to inject the data into the wireless or sensor devices when necessary. It might be any time during the experiment (real time). A GUI interface should allow the user to inject his data into the appropriate nodes when necessary. This injection should be abstract enough not to interfere with other nodes running different experiments and the data logging facility.

f) Tool for monitoring the health status of the nodes

To provide the user with up-to-date information about the nodes, a health monitoring service is necessary. If the user checks the health of the node often while the job is in process he can determine the quality and correctness of the output data. If the node is up and running before the job and after the job too, then if is less likely that the produced result is not correct. If the node is running before the job and if it fails in the middle and still gives some results, the users will know the result is not accurate.

Through the health monitoring tool, users can also know the job statistics on a particular testbed.

g) Provision to collect the results

The job of a testbed is not done with experiment scheduling, resource allocating, health monitoring, etc. It has to provide the results to the users for analysis. A GUI should provide the user with the link to collect his experimental data, errors if any to analyze.

With all the above facilities provided and running, a testbed provides a realistic, flexible, easy to use environment to the users. Such testbed provisioning is of high importance to the research community as it eliminates most of the disadvantages of simulated environments.

2.6. *Provision of administrative tools*

Administrative tools are necessary for maintenance of the testbed. Different kinds of administrative tools can be developed according to the requirements. The following explained administrative tools are very important in any testbed.

a) Creating New Users

Only the administrators should have privileges to create new users. Administrators should have a handy tool to add the user into the system. This helps to reduce the manual work of the administrators.

b) Health Monitoring as an Administrative Tool

The health monitor serves both as an administrator as well as a user tool.For administrators it is to help maintain the testbed. Through this service, administrators will easily know the nodes that are down and fixing the node will be easier. This way because the administrators do not have to go and check each node manually in such a large facility.

3. Survey of existing testbeds

The setting up of different wired, wireless and sensor network testbeds is growing in fast pace. A few of the important testbeds in United States are listed in the following table:

Testbed Name	Year	Establishment
Kansei	2004	The Ohio State University
Motelab	2004	Harvard University

SensorNet	2005	Intel Berkeley Research Labs
Orbit	2005	Rutgers(WINLAB), Columbia, and Princeton, Lucent Bell Labs, IBM Research and Thomson
Emulab	2002	University of Utah
Planetlab	2002	Berkeley, Intel Research, MIT, Washington, Rice, Princeton, Columbia, Duke, Utah, HP

Table 3.1: Testbeds detail chart

3.1. Kansei

The Kansei Testbed at Ohio State University was developed to provide sensing at scale. The Kansei testbed consists of stationary array with 210 nodes (XSM and Stargates) on a rectangular 15X14 rectangular bench with 3 feet spacing, 150 TMote sky nodes for mobility, 50 Trio motes in a portable array that duplicates the sensors in a stationary array for at-scale high fidelity sensing, and 5 robotic mobile nodes.

The objectives of the Kansei testbed include

- Provision of heterogeneous hardware infrastructure for wireless and sensor network experiments. Stargates are used as wireless network devices and TMote sky nodes are used as sensor nodes.
- A time accurate hybrid simulation engine for simulation that is designed to simulate large-scale experiments with an emphasis on high fidelity on timing, application, radio and sensing environments and complements the prior work on simulation.
- High fidelity sensor data generation and real-time data and event injection on the fly, which enables dynamic experiments with the mobile nodes where the user can change the input processes in real time.
- Software components to support complex multi-tier experiments utilizing real hardware resources, and data generation and simulation engines.

In the Kansei testbed a user can define the topology and schedule a job through the remote web interface. There is provision for data storage, data analysis, remote access, and high fidelity sensor data generation and injection. Easy data retrieval is another added advantage in the Kansei Testbed. The Kansei Testbed includes a cluster of PCs for running visualizations, compute-intensive analysis, high-fidelity sensor data generation, hybrid simulation, and diagnostic analysis. One of the PCs, called Director is a remotely accessible

multi-user framework provides the services. It provides services for experiment scheduling, deployment, monitoring and management for all array platforms like motes and stargates, creation and management of testbed configurations, system administrative services for managing the testbeds, and it contains tools for data injection, and logging of all array platforms and tools for hybrid simulation. "Chowkidar" which means watchman, is Kansei's health monitoring tool. Periodically the health-monitoring service runs automatically giving important data about the testbed devices.

Health monitoring serves both users and the administrators. For users monitoring this data helps in analyzing the real time data if any node is corrupted or producing error, etc. For the administrators it helps in detecting if the node or device is not working and helps in knowing the reason why it happened. The software is developed in PHP, Perl, and MySQL as the backend.

3.2. Motelab

The objective of the Motelab testbed at Harvard is to meet the challenge of deploying, developing, and debugging applications on realistic environments on a large scale. Recently, they have deployed 190 TMote Sky sensor "motes" with T1 MSP430 processors running at 8MHz, with 10KB of RAM, 1Mb Flash Memory, and a Chipcon CC2420 radio. The CC2420 radio operates at 2.4GHz and has an indoor range of approximately 100 meters. The motes are powered from wall power. The motes run the TinyOS operating system and are programmed in NesC language. The mote provides two TCP ports, one for reprogramming and one for data logging. The functionality of Motelab includes create, edit and schedule a job, programming the node, and logging the data through a central server. Users access the testbed using the web browser to set up or schedule jobs and download data.

The different software components of the motelab are

- Web interface for job creation, scheduling, and data collection.
- MySQL database backend for storing the collected data.
- DBLogger, a Java program to parse the generated data. It will be started at the beginning of every job. It connects to the data logging port of emote and its main job is to collect the data.
- Job Daemon, a Perl script job running as a cron job is responsible to set up experiments, which involves reprogramming nodes and starting other necessary system components like DBLogger and tears them down when finished, and dumping the data from the MySQL database into a format suitable for download.

Resource scheduling in motelab is done through defining the user quota, which facilitates sharing the lab between multiple uses. The quota does not control how much total access the user will have on the lab. It limits the number

of outstanding jobs the user can post to the lab at once. Motelab also provides direct access to the nodes serial port over a TCP/IP connection. This helps in monitoring and injecting the data to the motes via its serial port. An important service motelab avails is power measurement. Since the sensor network developers become aware of power as a design constraint this feature is developed. A networked Keithley Digital Multimeter is attached to a node and it will sample continuously at 250Hz, it bursts at 3000Hz, and this value is displayed on the main page of the web interface.

There are two different ways to use motelab. One is batch use where a user can schedule a batch of jobs and does not have to worry about them till he gets the data. Other is real time use, where the user can interact directly with their job by attaching to the exposed per-node serial forwarder or accessing the MySQL database. Motelab has a connectivity daemon, which is kind of health monitoring tool. It is used to collect the information about the nodes and graphically represent the connectivity on the Maps pages. The administrators can run it periodically when the lab is available. A Java program that runs during the job connects directly to active nodes and uses this connection to tell them when to send messages and when to listen. A Perl script accesses the database tables and calculates the packet losses and updates the connectivity information between the pair of nodes.

3.3. SensorNet

SensorNet is a testbed by Intel Berkeley Research. It has 150 MicaZ motes with CC2420 802.15.4 radios. A canonical testbed incorporates a collection of compute and sensing nodes that are coupled together by an out-of-band communication channel and a power supply. This channel provides for remote control, reprogramming and data collection independent of a node's wireless capabilities. The power supply eliminates the need to replace batteries, reducing the physical maintenance needs of the testbed.

The current testbed resource allocation algorithm has the following shortcomings:

- Inability of meeting real user demands on a large system.
- Either lack or have inadequate mechanisms for resolving contention among competing users during peak times of demand. This system often requires direct system administrator intervention to resolve conflicts.
- They provide limited mechanisms for expressing resource. So the user cannot express desired resources and their associated constraints.
- This affects the efficient utilization of the underlying resources by limiting the system's ability to make intelligent allocations for multiple users.

To address user contention Intel Research Berkeley has developed tool "Mirage" for allocation of resources. It applies microeconomic approach to

arbitrate among competing users. Users will be given virtual currency to request for the resources. A combinatorial auction is then run periodically to determine which bids are allocated resources based on supply and demand while aiming to maximize aggregate utility. A combinatorial auction is periodically run to determine which user bids are allocated resources based on supply and demand. Any user who ever spends more currency on particular resources is the winner. The aim is to maximize the aggregate utility of the resources on the testbed.

3.4. Orbit

Orbit is an open-access research testbed for next generation wireless networks. The Orbit testbed consists of an indoor radio grid emulator for controlled experimentation and an outdoor field trial network for end-user evaluations in real-world settings. It consists of 20X20 802.11 radio nodes with 1m spacing in the indoor testbed and a mix of 3G and 802.11 nodes in the outdoor testbed which spans over a region of 5 Km wide and 2 km long space.

The design objectives of the Orbit testbed at Winlab, Rutgers include

- Scalability in terms of the number of wireless nodes.
- Reproducibility of experiments which can be repeated in similar environments for similar results.
- Open-access flexibility giving the researcher high-level control over protocols and software used on the testbed.
- Extensive measurement capability at all layers providing cross layer experimental possibility.
- Access to the testbed remotely through a web interface.

3.5. Emulab

Emulab at the University of Utah has a wide range of environments developing, debugging, and evaluating the systems. They call it universally available "Internet in the room" to provide balance between control and realism. The main design goal of Emulab is to provide penalty free accessibility of the nodes that is hundred percent configurable. The other goals are scalability and flexibility. Emulab contains 160 end nodes and 40 core nodes (PC650/800) which are wired, a few 802.11a/b/g nodes, 25 Mica2 motes for wireless sensor networks, 6 Garcia motes for mobility.

There are different experimental environmental facilities in Emulab like the following:

- Emulation-An emulated experiment allows specifying an arbitrary topology and gives controllable, predictable and repeatable environments giving full control of the nodes.

- Live Internet experimentation- Using PlanetLab, Emulab provided a full featured environment to run the applications at hundreds of sites around the world.
- An 802.11 wireless environment for wireless applications.
- Software Defined Radio- It gives control over layer 1 of a wireless network. Everything from signal processing and up is done in software.
- A sensor network facility for sensor applications.
- A mobile sensor Network facility for sensor applications involving mobility.
- Simulation- Using the ns-2 emulation facility simulated environment can interact with real networks.

Emulab provides services through provisioning of web-based tools to remotely configure reserve and control machines, to define error models, latency, bandwidth, and packet ordering. It provides high speed network processor chips. Emulab also has non-intrusion instrumentation for security. Emulab network testbed software provides the first remotely-accessible mobile wireless and sensor testbed. This testbed is robust. Robots carry motes and single board computers through a fixed indoor field of sensor-equipped motes, all running the user's selected software. In real-time, interactively or driven by a script, remote users can position the robots, control all the computers and network interfaces, run arbitrary programs, and log data.

Emulab provides simple path planning, a vision-based tracking system accurate to 1 cm, live maps, and webcams. Precise positioning and automation allow quick and painless evaluation of location and mobility effects on wireless protocols, location algorithms, and sensor-driven applications.

3.6. PlanetLab

PlanetLab works as a global platform for designing and evaluating network services. PlanetLab is the first testbed that is distributed across the world with 834 nodes at 408 sites. Planet Lab's design evolved incrementally based on experience gained from supporting a live user community called experience-driven design, and the design decisions made with conflicting requirements called conflict-driven design.

The design objectives in PlanetLab include

- Immediate availability to researchers to do their experiments. It is achieved by spreading the nodes all across the world and implementing a fair resource allocation algorithm.
- Global platform that supports both short term experiments and long running services. The services run continuously and support real client workload.

- Autonomy and decentralized control is achieved through distribution of control to different organizations who contributed to PlanetLab and thus reducing the centralized control.
- Scalability to support many users with minimal resources by promoting efficient resource sharing.
- Distributed virtualization, which means running a service as a slice in every node which acts as an individual system. The user does not even know that he is just using a part of the resources; it seems all the resources are allocated to him through this service.
- Unbundled management services used in the PlanetLab testbed has three arguments, namely
    - To allow the system to more easily evolve;
    - To permit third-party developers to build alternative services, enabling a software bazaar, rather than rely on a single development team with limited resources and creativity; and
    - To permit decentralized control over PlanetLab resources, and ultimately, over its evolution.

3.7. Physical parameter comparison of different testbeds

Each testbed discussed so far has a different level of design detail according to the goal and requirements. Table 3.2 provides the comparison chart of different testbeds with respect to different physical parameters like the type, node type, radio type, scale and field.

Parameter	Kansei	MoteLab	SensorNet	Orbit	Emulab	Planet lab
Type	Wireless, wireless sensor	Wireless sensor	Wireless Sensor	Wireless	Wired, Wireless, Wireless sensor and Mobile	Wired Networks
Node type	XSM, Stargates TMote sky, robotic mobile nodes	TelosB, Mica2 motes, MIB-600 motes	Sensor motes	802.11 nodes, 3G cellular	Pc(600/850) nodes, 802.11X nodes, mica2 motes, robotic nodes	Wired nodes

Radio Type	802.15.4	802.15.4	802.15.4	802.11	802.11/ 802.15.4	-
Scale	1500	190	150	400	250	846
Field	Indoor/Outdoor	Indoor	Indoor	Indoor	Indoor	Distributed

Table 3.2: Physical parameter comparison table for different testbeds.

3.8. Design parameter comparison of different testbeds

.

Table 3.3 provides the comparison chart of different testbeds with respect to different design parameters like the topology, resource allocation, health monitoring, and available tools.

Design	Kansei	Motelab	SensorNet	Orbit	Emulab	PlanetLab
Topology design	3ft wood tables, 20db antenna	Multiple floors	-	Indoor 20X20 nodes with 1m space.	Multiple floors	Across the world
Resource Allocation	FCFS, best-effort	FCFS with user quota, best-effort	Auction model	FCFS, best-effort	FCFS	Best-effort and guaranteed
Health Monitoring	Chowkidar	Connectivity graphs	-	Script	Slice, sliver mechanism	Health Monitoring service
Tools	Select topology, Schedule, Upload, manage, retrieve	Select topology, Schedule, upload, manage, retrieve	No user tools for data logging and automated reprogramming	Schedule Ruby script to upload and select nodes, orbit commands	Ns2 script to select topology, schedule, run and terminate the experiment. Results through email	Scripts to schedule program

Table 3.3 Design parameter comparison of different testbed

4. NetEye

The NetEye testbed at Wayne State University is deployed to facilitate research on wireless and wireless sensing applications at scale. The NetEye testbed consists of a controlled indoor environment with a set of sensor nodes and wireless nodes deployed permanently. This topic helps the reader to understand the implementation of the design issues in greater detail.

The NetEye testbed provides a web interface to create and schedule a job on the testbed while automated reprogramming of the sensor devices and storing the experimental data on the server. NetEye has the following features:

- It motivates application deployment by providing access to a large scale, fixed sensor network.
- Through automatic storage and logging of data, it accelerates the debugging and development of an application.
- NetEye provides a web interface allowing both local and remote users to easily access the testbed.
- The resource scheduling mechanism used in NetEye ensures a fair share of resources to its users.
- NetEye consists of heterogeneous hardware for computations, data storage, and to support complex experimentation.
- It provides real-time data and event injection on the fly.
- Software components in NetEye support multi-tier complex experiments.

4.1. Interference Control in NetEye

The NetEye testbed has both wireless and wireless sensor network devices. The NetEye testbed deals with the issue of coexistence between wireless and sensor networks thus providing an interference controlled environment. The wireless network devices are referred to by IEEE as 802.11 devices and the Wireless Sensor Network devices as 802.15.4 radio devices. Both these devices share the 2.4 GHz ISB band. So there is a chance that the packets from one device can be interrupted by the other and the packets may be lost. More details about considering this as an important design issue are explained in section 2.

In NetEye this issue is taken seriously since the testbed setup itself contains both wireless and wireless sensor devices. There are many access points around the NetEye testbed set up, so definitely there is a need to find an interference free channel for 802.15.4 radio devices. There are 11 channels in 802.11 with only 3 non-overlapping channels. Each channel has a 22 MHz frequency range. There are 16 channels in 802.15.4 with 3 MHz frequency range and each channel is 5MHz apart.

To measure the environmental noise, a TinyOS application, written in the NesC language that samples RF energy at 1 KHz by reading the RSSI register of the CC2420 radio is used. The register contains the average Received Signal Strength Indicator (RSSI) over the past 8 symbol periods (125 microseconds). The application logs this data to flash for a fixed period of time (2 minute period with one sample for a millisecond). The data is collected on the PC through UART (serial port).

Noise is sampled on different radio channels in a variety of environments; including the NetEye embedded testbed lab at Wayne State University which has

six access points from the surrounding areas, Undergraduate library at Wayne State University where there is heavy usage of 802.11, Deroy apartments where there is no access point present but some peer-to-peer networks exist and an outdoor quiet area in Richmond where neither access points nor peer-to-peer networks exists.

To find the difference on how just the existence of the 802.11 access point without heavy usage and with heavy usage changes the interference levels on 802.15.4, an experiment is conducted to collect RSSI values while downloading a large HTTP file.

Table 2.1 in Chapter 2 shows the RF spectrum of available channels for 802.11 and 802.15.4. According to the table it seems like channels 25 and 26 of 802.15.4 do not interfere with any channels of 802.11.

Following experiments are done to determine the interference channel to provide the controlled environment:

- Channel 11 set in 802.15.4 when channel 1 is set in 802.11 network
- Channel 16 set in 802.15.4 when channel 6 is set in 802.11 network
- Channel 17 set in 802.15.4 when channel 6 is set in 802.11 network
- Channel 21 set in 802.15.4 when channel 11 is set in 802.11 network
- Channel 22 set in 802.15.4 when channel 11 is set in 802.11 network
- Channel 26 set in 802.15.4 when channel 1 is set in 802.11 network
- Channel 26 set in 802.15.4 when channel 6 is set in 802.11 network
- Channel 26 set in 802.15.4 when channel 11 is set in 802.11 network

The following tables show the measured RSSI values at different areas at different times. The median and confidence intervals (95%) for 5000 samples are represented in the tables and the box plots are used to represent the overlapping and non-overlapping confidence intervals.

4.1.1 NetEye Embedded Lab

NetEye Embedded Lab is at Wayne State University and is the place where the testbed is set up. The room has few access points around. The values when Channel 26 is set in 802.15.4 and channels 1, 6, 11 are set in 802.11 are measured and they are stable with standard deviation almost less than 1. Channels 11, 17, and 21 in 802.15.4 have non-overlapping confidence intervals (CI) compared to channel 26 when corresponding interfering channels are set in 802.11.

The box plot below shows the non-overlapping confidence intervals between the interfering and non-interfering channels. The red line represents the median and the notches represent the confidence intervals.

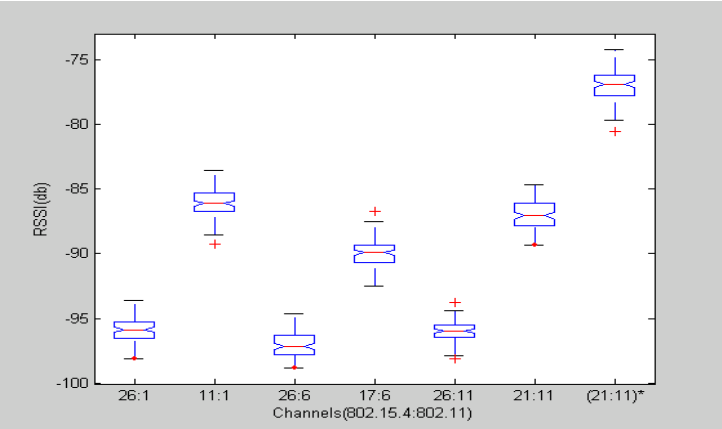


Figure 4.1: Boxplot showing median and 95% CI-NetEye Embedded lab

The first column is when channel 26 (non-interfering) is set in 802.15.4 and column 2 is when channel 11 (interfering) is set in 802.15.4 when channel 1 is set in 802.11. The RSSI values do not overlap showing the significantly different values. Likewise other values need to be compared. The last column (21:11)* shows the different non-overlapping performance compared to 21:11. This is when a large file is being downloaded and the values are significantly different.

4.1.2 Undergraduate library at Wayne State University

The Undergraduate library at Wayne State University has a heavy usage of 802.11 network. The box plot below shows the non-overlapping confidence intervals between the interfering and non-interfering channels. The red line represents the median and the notches represent the confidence intervals. The first column is when channel 26 (non-interfering) is set in 802.15.4 and column 2 is when channel 11 (interfering) is set in 802.15.4 when channel 1 is set in 802.11. The RSSI values do not overlap showing the significantly different values. Likewise column 26:6 and 17:6 show non-overlapping values. Columns 26:11 and 21:11show non-overlapping CI values

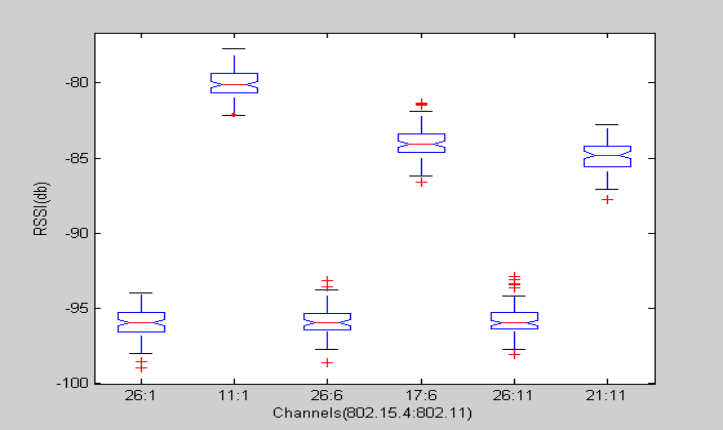
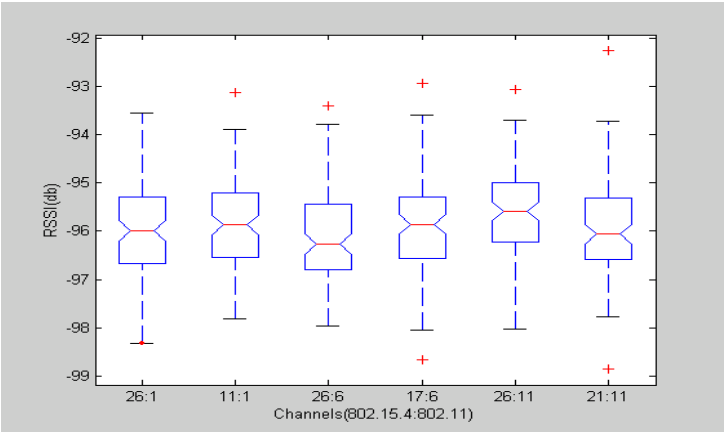


Figure 4.2: Boxplot showing median and 95% CI-Undergraduate Library

.

4.1.3 Deroy Apartments

The apartments in Wayne State University does not have any access points but have a few peer-peer networks.The box plot below shows the overlapping confidence intervals between all the channels. The red line represents the median and the notches represent the confidence intervals. The first column is when channel 26 (non-interfering) is set in 802.15.4 and column second is when channel 11 (interfering) is set in 802.15.4 when channel 1 is set in 802.11. The RSSI values do overlap showing the values are in the same range. Likewise column 26:6 and 17:6 show overlapping values. Columns 26:11 and 21:11 show non overlapping CI values because of the existence of peer-peer networks.



.Figure 4.3: Boxplot showing median and 95% CI-Deroy apartments

4.1.4 Quite Area in Richmond

There are no access points and peer-to-peer networks and the mean values are high, with much less standard deviation. The values at different confidence intervals are also almost in the same range. The box plot below shows the overlapping confidence intervals between all the channels. The red line represents the median and the notches represent the confidence intervals. The first column is when channel 26 (non-interfering) is set in 802.15.4 and column second is when channel 11 (interfering) is set in 802.15.4 when channel 1 is set in 802.11. The RSSI values do overlap showing the values are in the same range. Likewise column 26:6 and 17:6 show overlapping values. Columns 26:11 and 21:11 show overlapping CI values because of the non-existence of access points and peer-peer networks.

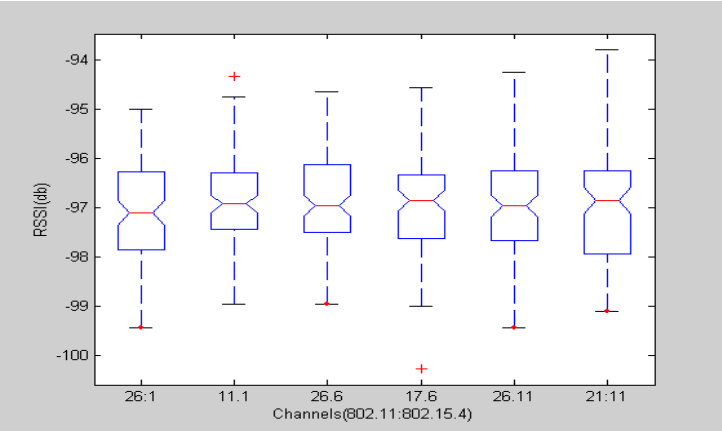


Figure 4.4: Boxplot showing median and 95% CI-Richmond

4.2. Results

With a different set of experiments done at different times and different places, it can be concluded that channel 26 set in sensor devices provides the interference free environment even in the presence of access points. It is not affected by any channel at 802.11 networks. It provides necessary abstraction even in the presence of wireless network. This is a very important study for setting up a sensor testbed since if the interfering channel is set it will be affected even by the beacon packets from the access points and there is high possibility to provide less accurate results to the user.

4.3 Elements of NetEye

The block diagram below shows all the elements of NetEye. The elements include hardware and software components.

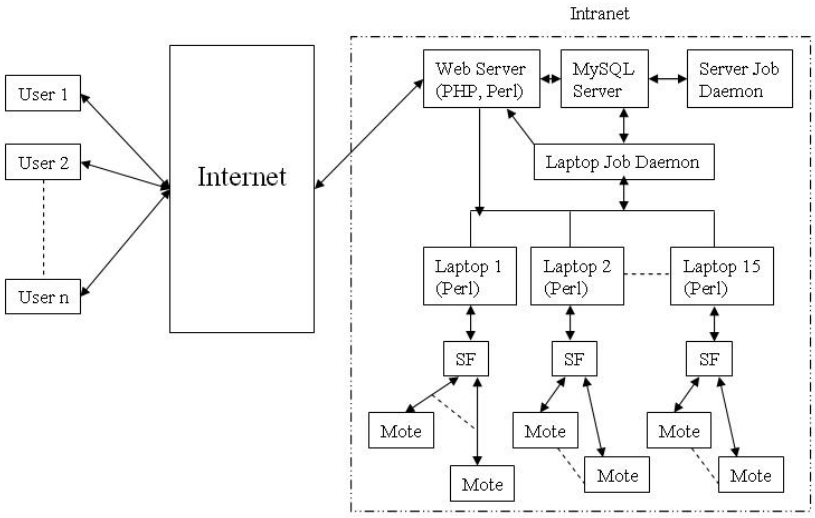


Figure 4.5: Elements of NetEye

The details of all the elements of the NetEye testbed are as follows

4.3.1 Hardware Components

NetEye consists of a fixed array of wireless and sensor nodes. The wireless nodes are Dell Vostro 1400 laptops with a Intel dual core processor, 1GB RAM, and 80GB hard disk. The operating system on the laptops is Linux Fedora core 8. There are 15 laptops arranged on 15 wooden benches. The benches are 1 foot apart column wise. Proper ventilation and air conditioning facilities are provided for the laptops as these will be used on a 24/7 basis.

The NetEye testbed has 130 "TelosB motes" connected to the laptops. The motes are connected through the USB interface. There are 6 to 12 motes connected to the laptops depending on the topology design. The design is based on providing a multi-hop network. The design is also optimized to reduce the number of USB hubs and cables used. There are 21 USB hubs connecting 130 motes to the laptops. TelosB is a 802.15.4 compliant device designed by UC Berkeley and manufactured by Crossbow Solutions. It has a T1 MSP430 microcontroller, 2.4 GHz Chipcon CC2420 radio, 4MHz 8 bit CPU, 128KB instruction memory, 4KB RAM, and a USB interface.

Motes run TinyOS 2.0.2, which is a lightweight event-based operating system that implements the networking stack and communication with the sensors, and provides the programming environment for this platform. Each mote accommodates a light sensor and has a customized onboard facility to hold a 3db RF attenuator. The spacing between the motes is 2 feet. Since the sensors are low power radio devices the TelosB motes use the power from the laptops which inturn use wall power.

Laptops in each column are connected through a Gigabit Ethernet switch to an Ethernet back-channel network that provides high-bandwidth connectivity to and from the nodes for management commands, data injection and extraction. The NetEye server connects to join the local Ethernet back channel network and it also has another Ethernet card through which it connects to the Internet. The NetEye server provides remote access to the users for creating, editing, and scheduling the job.

4.3.2 Topology Control in NetEye

Topology control is one of the important design issues for setting up a testbed. The topology setup in NetEye supports multi-hop transmission between the motes.

In the NetEye testbed wooden benches are used to hold the laptops and the motes since wood is a bad conductor. Another advantage is it does not interfere much with the signal propagation. The benches are 5 feet high in order to reduce the ground effect.

Channel 26 is set on the sensor nodes to avoid interference with any channels on wireless networks. The radio characteristics of the sensor networks are measured in the NetEye testbed. The experiments are done to select the minimum power level and to obtain reliable packet delivery in a multi-hop network. In the indoor environment the transmission ranges will be high. To deal with this a 3 db attenuator attached to each TelosB mote is used.

To achieve a multi-hop network a NesC program that broadcasts 10000 packets is scheduled on one of the nodes. All the other nodes are scheduled with a receiver program. The data is collected from all the nodes and the packets received are counted to know the Packet Delivery Rate (PDR). The experiments are done using different Power Levels (PL) and between different distant nodes. The graph below shows the Packet Delivery Rate at different power levels and at different distances.

Power levels 1 and 2 has very low delivery rate. Power level 3 can be considered since it offers reliable packet delivery rate up to 6 hops. Power levels 4, 5, and 6 have very high transmission ranges even with the attenuator.

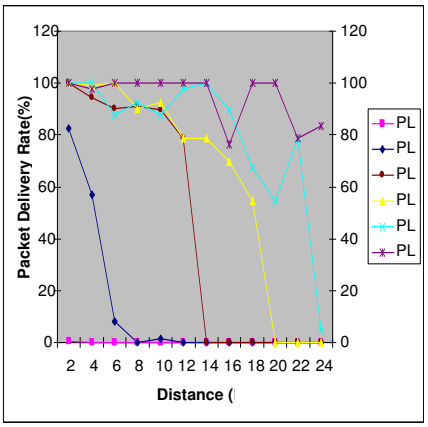


Figure 4.7: PDR vs Distance-At different power levels at channel 26

The actual network connectivity in the network depends on the power level adopted on the motes. The following box plots show the variation of packet delivery rate as a function of distance, when the power level is 2, 3, and 6.

From Figure 4.8, we see that the network is sparse at power level 2, where most nodes can talk to a limited number of nodes. Figure 4.9 shows at power 6 level the density is fairly high;almost any two nodes can talk to each other with high probability. Figure 4.8 also shows at power level 3, a node can reach many nodes in the network, and some links are reliable while some are not. Thus power level 3 gives a typical multi-hop network (about 6 hops).

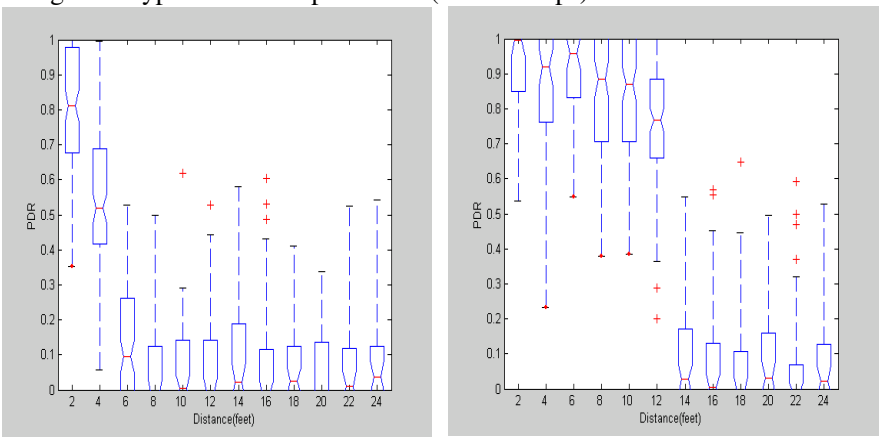


Figure 4.8: Boxplot for PDR vs Distance at power level 2and power level 3

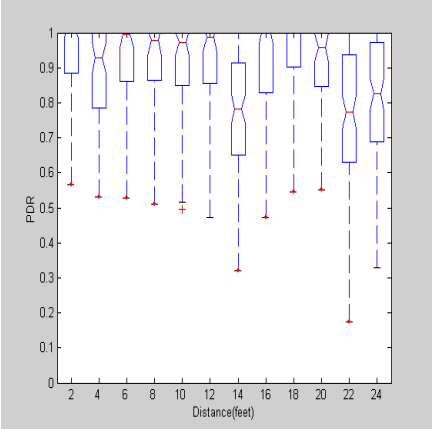


Figure 4.9: Boxplot for PDR vs Distance at power level 6

4.3.3 NetEye server

The NetEye server is a remotely accessible framework for wireless sensor network applications. A user experiment is referred to as a job. NetEye Server provides the following services:

- Web interface for experiment scheduling a job, deployment, and monitoring the fixed array platform.
- Resource scheduling for the user jobs to support multiple user jobs.

The job specified by the users consists of one or more files like sensor network executables, scripts, support files etc which should be programmed to run on the specific set of resources for a specified interval of time. The status of the jobs may be monitored during execution. On the successful completion of the jobs, output must be retrieved.

The NetEye server also handles the resource allocation mechanism according to the need and availability of the node which is FCFS scheduling. The NetEye server provides the set of services for optimized resource allocation for gathering the status of the NetEye testbed. The NetEye server collects the availability information from each of the laptop, which act as an interface between the server and the sensor nodes, considers the next job requirement, and allocates the appropriate resources for the job.

The NetEye server also implements the core set of system utilities. It includes tools for data injection, health monitoring, and logging of data. System administrative services are also part of the NetEye server. These services include user management for creation and deletion of users, assigning access privileges to the users, and platform administration like restarting a node, etc.

The NetEye server exposes these services for scheduling, managing, creating jobs etc, through a web interface using a web server. The NetEye server implementation is independent of specific hardware and platform.

4.3.3.1 . NetEye Server Implementation

The NetEye server implementation is done in a way that if the sensor node has to be deployed a component on the server will be invoked, which in turn invokes a component in the laptop to which the node is connected. All the components in the NetEye server are hierarchical and is independent. The main NetEye server runs on Linux (which also runs the Apache webserver for the web interface).

Testbed scheduling, administration, management, and experimentation are implemented in a multi threaded daemon that uses scripts and utilities written as PERL modules and which encapsulate testbed services. PHP modules implement the web-accessible testbed services, such as job-creation, storage of experiment data, and a testbed health monitoring page. A MySQL database provides persistence for storing job configurations and user reservations. Data generated by jobs are stored on the server database and may be retrieved by links through the web interface. Data injection is done by an Injector Manager component on the server and the implementation is socket programming in the C language.

4.3.4 Integration between NetEye server and motes (Laptops)

The laptops serve as integration points and interface the motes with the server. They also, in turn, acts as wireless network devices.
It provides the channels for:

- Data collection: The laptop collects the experimental data from the motes attached to it and sends it to server.
- Local data storage: The laptop stores the local information about the motes and locally stores the data from the nodes during the running of the experiment.
- Transfer of data like the executable files, injection files, and output files between the server and motes
- Local node information: The laptop collects the local node information. The health monitor service contacts each laptop to collect the health status of the nodes.
- Data injection: Each laptop has an injector for injecting events to applications running on it, as well as to serve as a bridge that injects events into the application on the mote. Each mote application has a receptor component that receives injections and takes appropriate actions.

4.3.4.1 Integration Point (Laptop) Implementation

The laptops also run on the Linux operating system. They have the local MySQL database. The laptops act as clients to the MySQL server set up on the NetEye server. In the database the local health information of the nodes, the jobs scheduled on the particular nodes attached to the laptop etc, are stored. The duration and start time of a particular job are pushed on to the laptop by the server if the node attached to it is selected for a particular job. Reprogramming the nodes and collecting the data from the nodes is done through daemons (written in shell and Perl).

Each laptop has an injector which is invoked by the Injector Manager from the server for injecting events to applications on fly. Each mote application has a receptor component that receives injections and takes appropriate actions. The injector code in the laptops is in C the language. TinyOS is installed on the each laptop to handle programming motes attached to it.

4.3.5 Software Components

NetEye consists of several different software components for providing the necessary functionality and services on the testbed.
The main components are:
a) Web interface

PHP generated pages are used for the user interface for job creation, scheduling, and data collection. Administrators also have an interface to control the testbed functionality. After logging users have the access to the following functionality:

- Home page
- Topology page
- Schedule tier-1 jobs page
- Schedule tier-2 jobs page
- Select node-group page
- Manage Tier-1 jobs page
- Manage Tier-2 jobs page
- Health Monitor Page
- Help and Instructions page

b) MySQL Database

MySQL database is used to store all the information necessary for running the testbed. There are two categories of information stored:

- Data related to the jobs: The JID, information of the user who scheduled the job, scheduled time, start time of the job, duration, information about the input files, status of the jobs, and the links for output retrieval are stored in this database.

- Testbed status data: The health status of each node, the work status of the node, etc. are stored in this database.

c) Job Daemon

Perl and Shell scripts run as cron jobs for setting up, running, and tearing down the jobs. The job daemon also does reprogramming of the nodes and starts necessary services like the serial forwarder and serial listen to get the data to the PC for a particular job. After the specified duration the job daemon tears the jobs down, kills processes which were running during the jobs, and puts the output data back to the server and updates the databases with the links.

NetEye provides the following functionality through different software components.

4.3.5.1 New user

A new user can easily register into the NetEye testbed. A user can run his 802.11 experiments as well 802.15.4 experiments on NetEye testbed. The user has to just fill in the new registration form available on the web interface of NetEye, answering a few simple questions. .

As soon as the account is available for the user, a login and password will be provided to the user through the email he has specified in the registration form. A login directory will be created for the user at the start when he registers, which will contain all the input files and output files for the jobs specified by the user. Different users can have input files with the same names. The aspect of creating directories for individual users helps in preventing the user files from being overwritten with one another.

The New User form is implemented in PHP. As the user registers an automated mail is sent to the administrator stating the user's interest in registration. An account will be created for the user through the automated administrative tools with appropriate user privileges. A directory will be created automatically (written in Perl script) on the server for the user to store user input files and output files.

4.3.5.2 Topology selection

Each user is provided with the topology of the testbed from which he can select a subset of nodes he wanted for the experiment. A user-friendly interface is provided for the user to select the nodes remotely.

The user has two options while selecting the topology:

- The user is given the option to name the selected nodes if he wants. This is stored in the database for reusing rather than selecting it every time if he wants to use the same set of nodes. This is available in the creating node group web page in the NetEye web interface.

- If he does not want the option of naming the nodes he can directly select them and use them. This is available in scheduling tier-1 job web page in the NetEye web interface.

The graphical selection tool is provided in the form of a grid of 10X13 for topology selection where a click will select the appropriate nodes for the user. This tool is implemented in Javascript and PHP.

4.3.5.3 Schedule jobs

NetEye users can schedule jobs through a web interface. The user can specify the start time of the job, the nodes selected for the experiment, and the duration time for the experiment. The user can upload the executable that needs to be programmed on the selected nodes.

The user can schedule the other jobs simultaneously and if the resources are available those will be allocated; otherwise the job will be in the pending state. In tier-2 jobs the user can upload support files along with the executable. The support files should be in zipped format.

The Schedule Job web page is in PHP. As the jobs are scheduled an entry will be made into the MySQL database about the job information like JID, start time, duration etc.

4.3.5.4 Upload files

For the users to run their experiments users can upload their executables through the web interface. Once the file is uploaded it will be stored in the user directory, which is created during user registration. These will be copied to the appropriate laptops through the secured copy protocol according to the user node selection. The selected nodes will be programmed with the uploaded file.

For the ease of the users, the files uploaded by them will be stored and the user can select from the stored files as well. The user can select from only the files he has uploaded and not from other's files. This facility serves two purposes:

- If the user has to run the same experiment at different times, he does not have to upload them again, he can just select from the existing files.
- If the user does not have his executable files available to him at a particular moment when he wishes to run his experiment, he can upload them earlier and use them when he wants to run the experiment.

For tier-2 jobs there is a possibility that he can upload support files along with his executable all in zipped format. Both the executable and support files will be uploaded in one directory, so that those can be used easily whenever needed. The schedule web page will link the upload web page and is in PHP. The user files will be uploaded into the directory through a PHP program but the secured copy

protocol is implemented in Perl to copy the files to the laptops. The uploaded files are stored in the database on the server for further usage.

4.3.5.5 Resource Management in NetEye

In any shared computing and communication infrastructure resource allocation plays an important role. The NetEye testbed will be shared by many users and each user should get a fair share of resources.

FCFS resource allocation is used in the NetEye testbed. The user is provided with the topology of the testbed from which he can select a subset of nodes he wants for the experiment. A user-friendly interface is provided for the user to select the nodes remotely. The user is also allowed to specify the start time if he wants to, otherwise the current time will be taken as the start time for his experiments. A few services running as daemons on the individual laptops collect availability information of all the nodes and send it to the server. The server stores all the information in the MySQL database and the resource allocator uses the information from the database to check the availability and allocates the resources to the user. If not the user's job will be put in the pending list and will be allocated with the resources once they are available. The allocation is based on the start time as the priority.

After the specified duration of a particular job, a module that will be running as the daemon will tear down the job and free the resources for further usage. After the duration, the daemon tears down the job and notifies the server. The resources are made available for further usage.

There are different tables in MySQL database which hold the state information about all the nodes in the facility. Job daemons (written in Perl) collect state information about the nodes from the database tables. They are responsible for setting up the jobs, tearing down the jobs after the duration, and updating the status of the nodes in the database. A local database is maintained in each laptop, which acts as a client to the NetEye server. The database holds the local information about the availability of the nodes and updates it to the server.

4.3.5.6 Manage jobs

The user can manage his jobs through the web interface remotely. There will be some sensitive information about the user and he might not want others to collect his data. For security reasons and providing confidentiality the user can view just his jobs.

The Manage jobs interface shows the user details about his job including the Job ID, the input file, start time, duration, and the status of the job. A link to the output directory is also provided in the Manage jobs interface. A cancel job option is provided on this web page if the user wishes to cancel his job. The status of the jobs will always be shown to the user through the Manage jobs

webpage. There will be a different status for each job depending on the availability of the nodes. The different statuses for a given job are as follows:

- Pending: The job has been entered in the database, but the resources requested are not yet allocated.
- Reserved: The resources for the particular job are allocated.
- Running: The motes are programmed and the job is running.
- Finished: The job is finished and the packaging module on the server is preparing for the output.
- Completed: The output link is available to the user and he can download from the provided link.

The web page for manage jobs is implemented in PHP. The job information and the status of the job is picked from the MySQL database and displayed to the user. The output link provision is also done in PHP once the job is finished.

4.3.5.7 Cancel jobs

If the user mistakenly schedules a job it should be corrected by canceling the job. It is useful for both the user as well as the resource providers. For the user it is useful because he does not have to waste time waiting for job that he does not want to complete or if he wants to schedule a different job on the same nodes he does not have to wait till resources get free. For the providers wastage of resources is reduced.

In NetEye the user is given the option to cancel a job if he does not want it or he has mistakenly scheduled it. This option is provided on the Manage Jobs webpage for the remote users. No output will be provided to the user. The job will be torn down and the resources will be freed for further usage.

The cancel job option is provided on the Manage Jobs web page which is in PHP. For canceling the job there are daemons running on the server as well on the laptops (written in Perl). Once the user cancels a job, then the information will be updated in the database by the daemon running on the server which will be picked up by the daemon running on the particular laptop to which the node is attached. This daemon will kill all the necessary processes like serial forwarder, listen tool, etc. to tear down the job and update it to the database on the server.

4.3.5.8 Programming the Motes

The user can upload the executable to program on the selected nodes. The users also have an option to select an already uploaded file for programming. For security reasons he can just use the files he has uploaded earlier. Programming the nodes is done using shell script. A daemon running on the individual laptops checks if the nodes are allocated to the particular job scheduled and it calls this shell script to program on the motes.

The users have two options while programming the motes:

- If the user does not select the injection service, just the Java listen tool will start running and collect the data from the motes through serial port. The user has to take care to write his code to send the data through the UART.
- If the user selects the injection service, a serial forwarder (one instance for each mote), an injection tool, and a listen tool listening to the serial forwarder will be opened. 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.

4.3.5.9 Killing a Job

The user processes for a particular job have to be killed after the scheduled duration or if the user cancels the job. If the jobs are not properly killed it will hamper the resource availability on the testbed.

A shell script will be called by a Perl daemon to kill the processes running by the job after the duration or on the selection of the cancel option. The script will collect the processes ID's and kill the listen tool, serial forwarder, and the injection tool to free the resources.

4.3.5.10 Data Retrieval

The user can easily retrieve his experimental data using the link provided on the web page. The output will be in zipped format. The user can just download his data with a simple click. The user can't either see or download other's data for security reasons.

The directory created for the user will be used to store his experimental data. Each output file is named with the Job ID to prevent it from being overwritten by other files. The Java listen tool and the serial forwarder tool are used to collect experimental data. Once the duration of a job is completed, a secured copy protocol is used by each laptop to push the data to the server. The data thus obtained is zipped on the server by the packaging module written in Perl and the link is provided to the user.

4.3.5.11 NetEye Injection

NetEye provides a service of real-time data injection. Users can inject traces of data to the nodes through a web interface option. If the user selects this option a tool called Injection Manager on the server will be invoked to send the traces to the selected nodes through a set of dedicated ports. The user has to upload a file that contains the injection traces. The format should be such that the first field

should contain the IP address of the laptop to which the mote is attached and the second field is the port number of the mote.

The port number can be obtained from the node ID itself. For example node "A0" means it is the mote attached to port 0 of Machine A. The IP address of the laptop to which the mote is attached can be obtained through the topology web page. The last field should be the time interval between each injection. There will be 6 bytes of data between the port number and time interval. Through this facility the user can pipe the output of one job to the other job on the fly. It is also possible for the user to send some input traces to the nodes for running a particular job if need be in real time.

The injection option is provided to the user on the Schedule Job web page. If the user selects it he is given an option to upload a file that contains the injection traces. The PHP script checks if the option is selected and updates it to the database on the server. A daemon on the server will invoke the injection manager tool (written in the C language) and send the data through the socket to the individual laptop to which the node is attached. A daemon running on the individual laptops will check the database if the injection is selected for a particular job. If it is a serial forwarder (tool provided by TinyOS), a injection tool (written in C language) and a listen tool (provided by TinyOS) will be invoked.

There will be different instances of serial forwarders and injection tools running on different ports. The ports are selected as per the USB port to which the mote is attached. The injection tools will listen to the port to which the server sends the data and inject the data to the serial forwarder. The serial forwarder will in turn send the data to the serial port. The mote application that the user specifies should have a receptor component which will read from the UART and take appropriate actions.

4.3.5.12 Health Monitoring

Health monitoring is one of the important services in any testbed for both users and administrators. Remote users can check the health status of the nodes on the testbed. The health monitoring daemon uses an executable to periodically run on all the available nodes to monitor its health status. Through this facility a user knows the following details about the testbed:

- Number of Tier-1 nodes up
- Number of Tier-1 nodes down
- Number of Tier-2 nodes up
- Number of Tier-2 nodes down
- Number of Tier-1 nodes busy
- Number of Tier-1 nodes free
- Number of Tier-2 nodes busy

- Number of Tier-2 nodes free
- Number of Tier-1 jobs running
- Number of Tier-1 jobs pending
- Number of Tier-2 jobs running
- Number of Tier-2 jobs pending
- List of nodes that are down

Through this information a user knows which nodes are not available and cannot be used for scheduling. The administrators can identify the broken nodes for fixing them easily through this service; otherwise the administrator has to check each and every node manually for its health status.

A health monitor daemon (written in Perl) runs periodically on all the available nodes to determine if the nodes are up and running. A NesC module is run on the nodes automatically and the data is collected to the laptop through the Java listen tool. The data collected to the laptop is checked for correctness if the mote has generated an AM type 99 message. If it is of type 99 then a Perl daemon script running on the laptop accesses the database table on the server and updates that the motes attached to is up and running; otherwise it updates the corresponding mote as down. The server receives the health information from all the laptops about the motes attached to it. Finally a PHP module collects the information from the MySQL database dynamically and displays the information to the user.

4.3.5.13 Administrative tool to add new users

The NetEye testbed also provides an administrative tool to add users and allow access privileges to them. Once the user registers using the registration form an automated email will be sent to the admin. After checking the authenticity of the information the administrator will add the user into the database. Appropriate access rights are given to the user by the administration. The default is a normal user who can use the testbed for scheduling, data retrieval, etc. The normal user won't have rights to view other's data. This admin tool is provided on the web interface and can only be seen if logged in as admin. The code uses PHP and MySQL database.

5. Concluding Remarks

We have discussed the important elements of sensornet testbed design, and we have presented in detail the design and implementation of the NetEye testbed. The design issues and the implementation details presented in this chapter will be useful for researchers who are interested in building their own testbeds.

Future work on NetEye includes increasing the scale and upgrading to accommodate mobile robots. Other areas of future work include integrating NetEye with the Global Environment for Network Innovations (GENI), which

will consist of a global wired network with programmable, virtualized network elements along with edge wireless access deployments intended to support experimentation with mobile computing devices, embedded sensors, etc.

As a part of this effort, NetEye will participate in the Kansei Consortium (which includes organizations such as The Ohio State University and Los Alamos National Lab) to deliver a prototype of GENI-fied infrastructure for wireless sensor network experimentation. Each testbed site will offer multiple types of sensornet platforms and operate autonomously. The prototype will also include a number of researcher client tools and researcher portal services. It will provide publicly available support for programmability, virtualization, and slice-based experimentation.

References

1. D. Raychaudhuri, M. Gerla, (eds.), "New Architectures and Disruptive Technologies for the Future Internet: The Wireless, Mobile and Sensor Network Perspective", in NSF WMPG Workshop Report, August 2005.
2. J. Evans, D. Raychaudhuri, S. Paul, "Overview of Wireless, Mobile and Sensor Networks in GENI," in GENI Design Document 06-14, Wireless Working Group, September 2006.
3. S. Paul, "Requirements Document for Management and Control of GENI Wireless Networks", GENI Design Document 06-15, Wireless Working Group, September 2006.
4. . Thomas Anderson, Amin Vahdat(Eds), "GENI Distributed Services", in GENI Design Document 06-24, Distributed Working Group, September 2006.
5. Emre Ertin, Anish Arora, Rajiv Ramnath, Mikhail 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
6. Sandip Bapat, William Leal, Taewoo Kwon, Pihui Wei, Anish Arora , "Chowkidar: A Health Monitor for Wireless Sensor Network Testbeds", in proceedings of the 3rd IEEE International Conference on Testbeds and Research Infrastructure for the Development of Networks and communities (TridentCom 2007), May 2007
7. D. Raychaudhuri, I. Seskar, M. Ott, S. Ganu, K. Ramachandran, H. Kremo, R. Siracusa, H. Liu and M. Singh, "Overview of the ORBIT radio grid testbed for evaluation of next-generation wireless network protocols", in proceedings of the first IEEE International Conference on Testbeds and Research Infrastructure for the Development of Networks and communities (TridentCom 2005), Feb 2005.
8. Geoffrey Werner-Allen, Patrick Swieskowski, and Matt Welsh, "MoteLab: A Wireless Sensor Network Testbed", in the proceedings of the 4th IEEE International Symposium on Information Processing in Sensor Networks (ISPN 2005), April 2005.
9. Brent N. Chun, Philip Buonadonna, Alvin AuYoungz, Chaki Ngy, David C. Parkesy, Jeffrey Shneidmany, Alex C. Snoerenz, Amin Vahdatz "Mirage:A Microeconomic Resource Allocation System for Sensornet Testbeds" in proceedings of the 2nd IEEE Workshop on Embedded Networked Sensors (EmNetS-II), May 2005.
10. David Johnson, Tim Stack, Russ Fish, Daniel Montrallo Flickinger, Leigh Stoller, Robert Ricci, Jay Lepreau, " Mobile Emulab: A Robotic Wireless and Sensor Network Testbed", in proceedings of 25th IEEE International Conference on Computer Communications (INFOCOM 2006), April 2006.
11. http://www.emulab.net/

12. Larry Peterson, Andy Bavier, Marc E. Fiuczynski, Steve Muir, "Experiences Building PlanetLab", in the Proceedings of the 7th conference on USENIX Symposium on Operating Systems Design and Implementation (Volume 7), 2006.
13. Andy Bavier, Mic Bowman, Brent Chun, David Culler, Scott Karlin, Steve Muir, Larry Peterson, Timothy Roscoe, Tammo Spalink, and Mike Wawrzoniak, "Operating System Support for Planetary-Scale Network Services", in Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation (Volume 1), 2004.
14. Larry Peterson "PlanetLab-A Blueprint for Introducing Disruptive Technology into the Internet", in Proceedings of ACM HotNets-1Workshop, Princeton, New Jersey, USA, October 2002.
15. http://www.xbow.com.
16. Chulho Won, Jong-Hoon Youn, Hesham Ali, Hamid Sharif, and Jitender Deogun, "Adaptive Radio Channel Allocation for Supporting Coexistence of 802.15.4 and 802.11b", in Proceedings of 2005 IEEE 62nd Vehicular Technology Conference (VTC-2005-Fall), September 2005.