International Journal of Engineering Inventions
e-ISSN: 2278-7461, p-ISSN: 2319-6491
Volume 4, Issue 1 (July 2014) PP: 18-24

Energy Framework Enhancements in the WSN Simulator OMNETPP

M.Thangaraj[1], S.Anuradha[2], S. Muthukala[3]
[1]*Department of Computer Science, Madurai Kamaraj University, Madurai-21,*
[2]*Department of Computer Science, Madurai Kamaraj University, Madurai-21,*
[3]*Department of Computer Science, Madurai Kamaraj University, INDIA*

Abstract: *Energy model study in the Wireless Sensor Network (WSN) simulations is the recent paradigm for the researchers. This Paper describes in details how the energy model study has been enhanced by the different frameworks and features of the simulator OMNETPP, a C++ based discrete event simulator. OMNETPP is chosen as the simulation platform because of its modular architecture and portable module libraries, Graphical IDE, Result analysis file generation capability and open source nature. The simulator's internal working model is described in detail to understand and plan for the energy model study at different levels. The frameworks MIXIM, INET, INETMANET, PAWIS, and CASTALIA of OMNETPP are explored for the study and the findings are recorded here. This Paper aims at guiding the researchers through step by step explanation of how to utilize the features of OMNETPP for the energy model study, generate outputs, create the analysis file and present them in a graphical format.*

Keywords: *Energy, Inetmanet, Mixim, OMNETPP, Pawis, WSN.*

I. INTRODUCTION

The study of energy in Wireless Sensor Network has multiple needs like configuring the nodes with initial energy level, managing the energy level at the node on every event of data packet transmission, data aggregation, routing, network layer transmissions, finding the residual energy at every node at any point of time, by introducing the energy efficiency improvement at the program level compare the energy levels. All these needs are to be programmatically addressed and recorded – displayed for the better understanding. A simulator is the platform which imitates the real environment at the level of software modules. Even multiple software are available in the market, OMNETPP outstands by its rich graphical interface and model libraries, class structures. OMNeT++ not only supports the simulator, but rather provides infrastructure and tools for writing simulations[2]. One of the essential ingredients of this infrastructure is component architecture for simulation models. Models are grouped from reusable components termed modules. Well-written modules are strictly reusable, and can be combined in various ways like LEGO blocks [1]. The frameworks like MIXIM, INETMANET, CASTALIA; PAWIS adds further features and flexibility to simulate the various aspects of OMNETPP.

II. BASIC MODULES

The **headings** The basic modules of OMNETPP simulations are represented by simple C++ classes. The classes are also particularly instrumented, allowing one to navigate objects of a running simulation and exhibit information about them such as class name, name, state variables or
Contents[4].

Module, parameter, gate, channel
Message, Packet, Battery
Transient detection and result accuracy detection classes
Data Collection classes
Container classes (queue, arrary)
Statistic and distribution estimation classes (histograms, P2 algorithm for calculating quantities etc.)

Herewith the class structures of the basic class models for reference

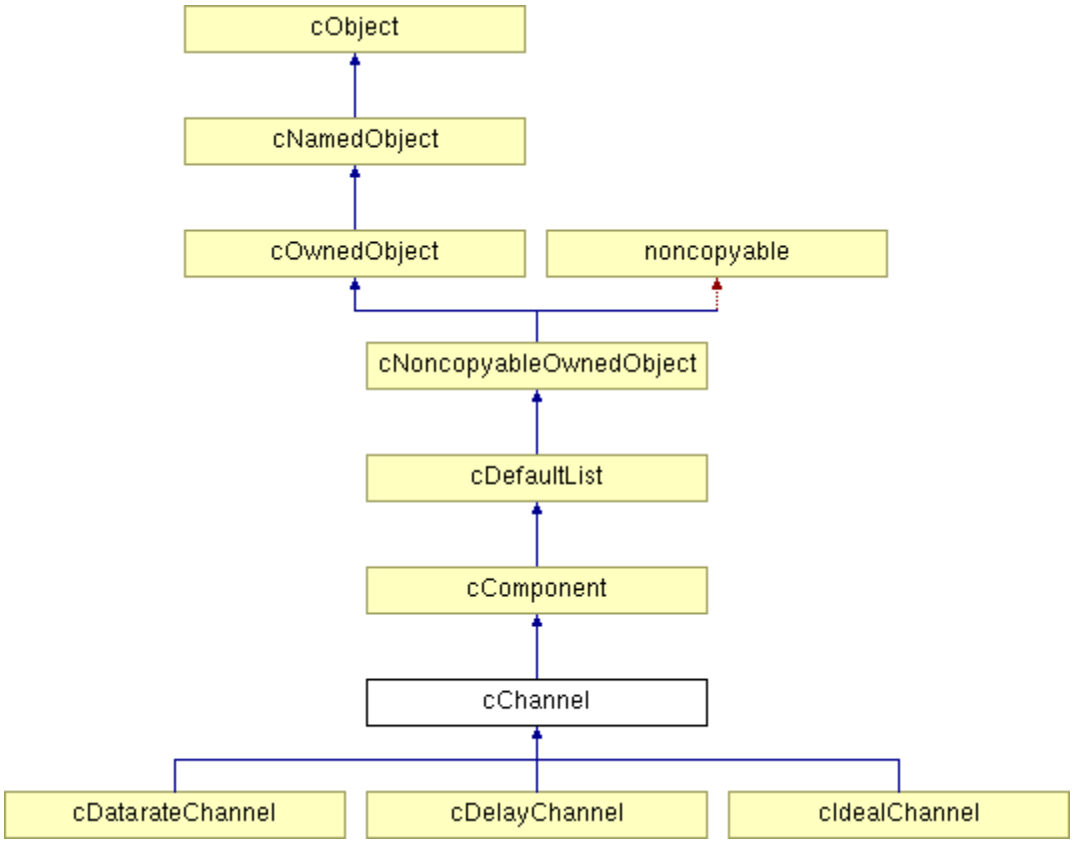


Fig.1 **Simulation Core class – Channel**

III. NED LANGUAGE

The first paragraph Every simulator has its own scripting language which has the deep learning curve of understanding and implementing. But, Omnetpp support the simple descriptive language for configuration of the WSN profile (number of nodes, frequency, lifetime, topology) . The user describes the structure of a simulation model in the NED language. **NED stands for Network Description.** NED lets the user declare simple modules, those modules are connected together them to form compound modules[4].

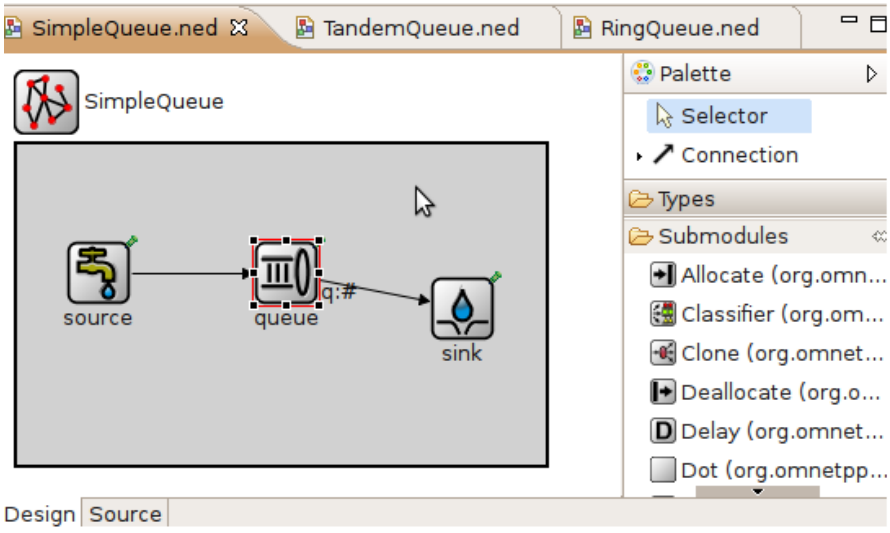


Fig.2 **NED editor for defining the network profile**

The user can label some compound modules as networks; that is, self-contained simulation models. Channels are another component type and instances can also be used in compound modules. With NED source editor the user can very easily code as it has the auto complete suggestions and property displays.

IV. ARCHITECTURE OF OMNETPP

As OMNETPP is the platform for the simulators, it supports and enhances multiple simulator framework gets embedded into the system. Loosely coupled component based architecture is the secret for the popular use of the tool by varied researchers [5].

Either it is MIXIM, CASTALIA, INET or chsim, it is the matter of porting the model library, UI library, configuring OMNETT bash with the relevant source paths, making and installing. For eg. The simple battery class of INET framework and the Battery class of MIXIM can be used along with default OMNETT functionalities like trace generation, snapshot creation, data model – chart plotting etc.

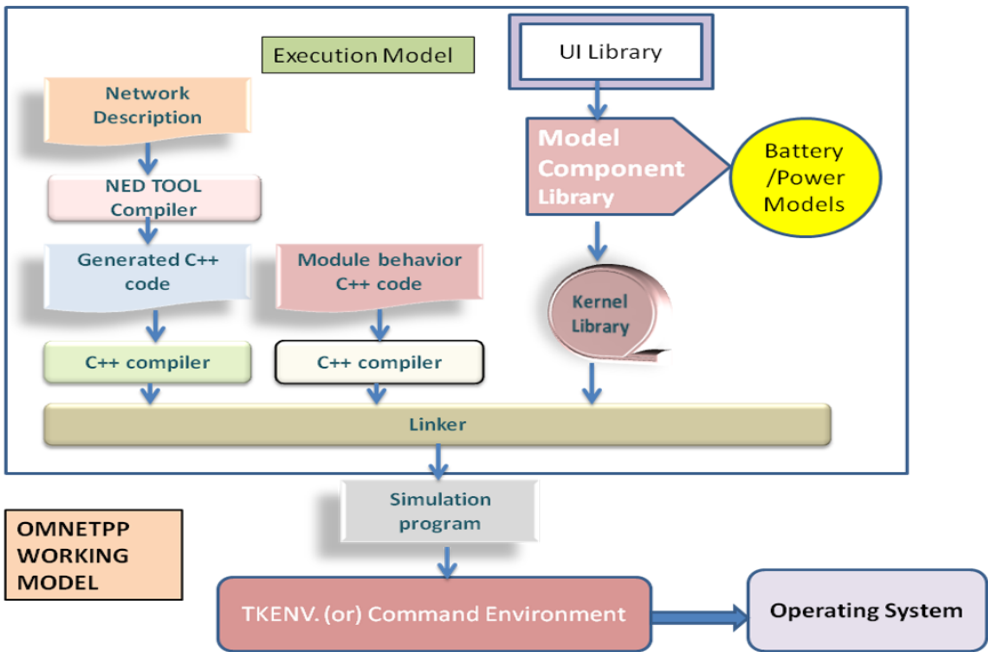


***Fig.3* OMNETPP Working Model with the power framework / Battery**

From here it is very clear that either it is NED or omnetpp.ini file, from it the C++ source file get generated and that is compiled and linked with the different library files like kernel library, model component, ui library , the created "Binary" is getting executed either on the Windows – cygwin environment or Linux operating system[6].

V. FEATURES OF ENERGY EXPLORATION

Initial WSN simulators are just extended from the conventional networks. So, most of the study didn't worry about the "Energy". But later there were initiatives on "Energy Level" optimizations as the sensor nodes are distributed at random and have limited power supply. It become mandatory to monitor the energy levels of the nodes and the energy spent on each activity. Here we discuss some of the features that support this energy study and the power spectrum details [4].

There are three major steps in any WSN simulations. One is, to setup the WSN profile, the second is to configure the simulation, the third is to record the output trace and analyze over it to produce the statistical and chart outputs. In all these steps we need the support of the energy exploration details.

5.1 Setting up The Wan Profile With Energy Parameter

When we are defining the omnetpp.ini or NED file, we should very clearly say the energy relevant parameters. SimpleBattery of iNet and BatteryState and Battery of MIXIM serve this purpose. The battery is the core of the Energy Framework, as it gives a well-defined message interface to any Device that consumes energy (i.e. any module that models energy consuming operations). It maintains an array of input Gates. In battery, size is specified by parameter numDevices, that is most easily specified in the Host.ned, where the

Device's connection to the battery is also specified [8].

```
battery.nominal = 50 mAh
battery.capacity = default (battery.nominal);
battery.voltage = 3.3 V;
battery.resolution = 10 s;
battery.publishDelta = 0.01;
battery.publishTime = 5 s;
battery.numDevices = 1;
battery.debug = true;
.....
txCurrent = 2 mW
```

Fig.4 **Battery parameters in WSN profile setting**

In the Energy Framework, a Device is any module that sends Register and DrawMsg's to the Battery and it can be a standalone module or the functionality can be integrated into another module. NIC80211 Battery and DeviceAF are two communication devices and several simple abstract test Devices have been implemented.

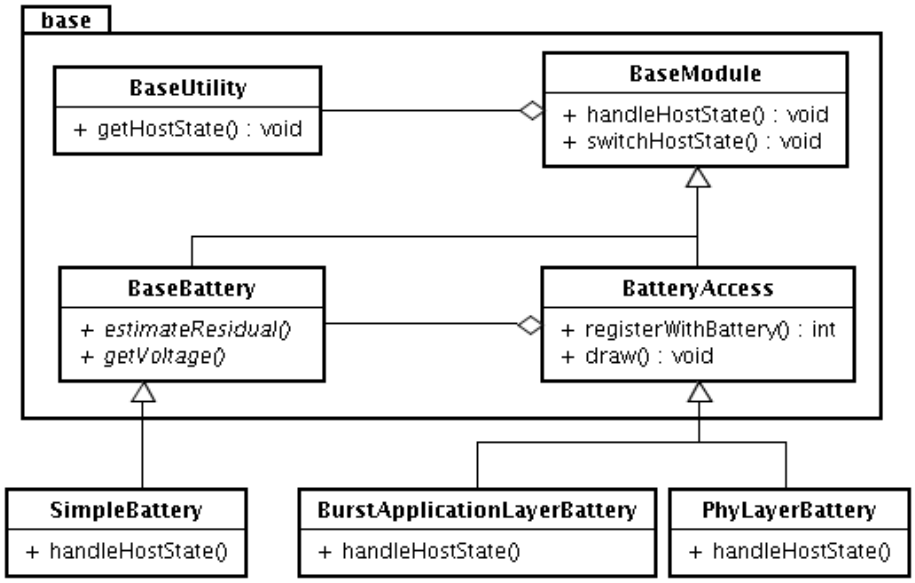
- NIC80211Battery

Nic80211Battery, which extends the mobility-fw 80211Nic. It is the best way to model battery consumption of IEEE 802.11 based systems.

- Device AF

DeviceAF is a standalone device which models battery consumption by listening to blackboard messages announcing the state of the Radio and sending DrawMsg's accordingly.

Fig.5 **Battery classes as part of base utility**



5.2 Configuring the Simulation with Energy Level

The **Battery** does not generate any statistics output by itself; this is done by the **BatteryStats** module. Two interactions are there between the Battery and the BatteryStats module [10]. The first interaction is for summary data. At finish (), the Battery passes its record of battery consumption (per-device and per-activity) to summary ()/detail () methods provided by the BatteryStats module and it allows the Battery Stats module to format data however it likes (note 4).

The second interaction is for time series data that the BatteryStats module obtains throughout the simulation. Battery publishes residual capacity (note 4) information on the blackboard, at intervals controlled by the parameters **'PublishTime'** and **'PublishDelta'.**

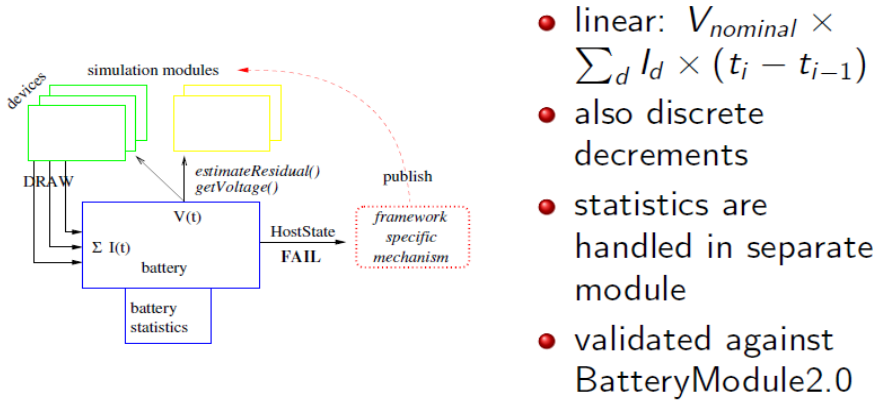


***Fig.6* Battery and Battery Statistics Classes communication**

It include battery as an active element in simulation

- Host failure due to battery depletion
- Energy-aware protocols and state-of-charge estimation
- Voltage-dependent operation
- Energy harvesting

5.3 Updating the Energy Level on Each Wsn Action

A Device can also send a 'drawMsg' that immediately deducts a certain amount of energy from the battery (ENERGY). The ENERGY message is used to represent the energy consumption of operations that are only modeled at high level of abstraction or that would be difficult to represent as a fixed current draw. Examples are state changes, sensor readings, etc. A deduction of ENERGY does not affect any ongoing CURRENT deductions [11].

```
drawMsg = new DrawMsg ("device wakeup1",
ENERGY);

drawMsg->setValue(wakeup);

            drawMsg->setActivity(WAKE);

send(drawMsg, batteryGate);

drawEnerg0079(wakeup, WAKE);

                  .....
```

***Fig.7* Draw Msg Usage in the code to configure simulation parameters**

The DrawMsg can also specify which of a Device's accounts should be "charged" for the battery consumption and the Battery keeps energy consumption statistics per-device and per-account within each Device. Each module must therefore send a Register message to the Battery, before sensing the first Draw message. Its providing a device name and number of activities to which the device will assign energy consumption [11].

5.4 Recording the Outputs of Energy in Wsn Simulation

The ultimate purpose is to record the energy level at different stages during simulation. The question is how effective and dynamic the output displays are. Is it iterated further into device level, communication layer level and is it recorded for further reference [9].

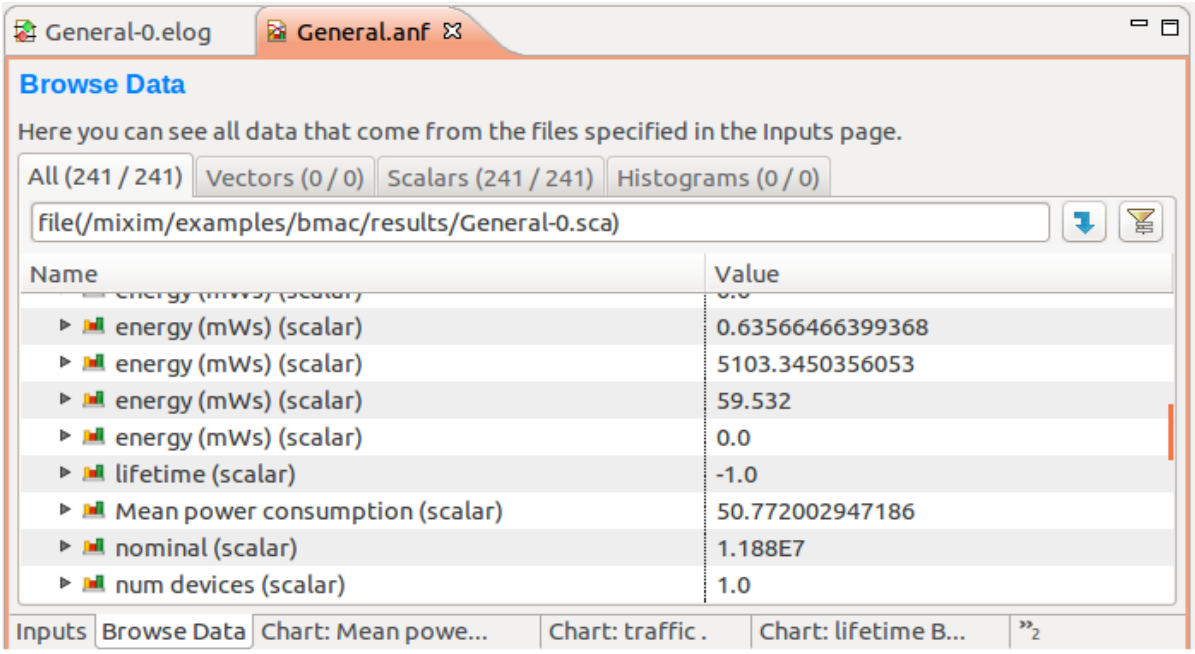


Fig.8 **Energy as lifetime, mean power consumption, device levels**

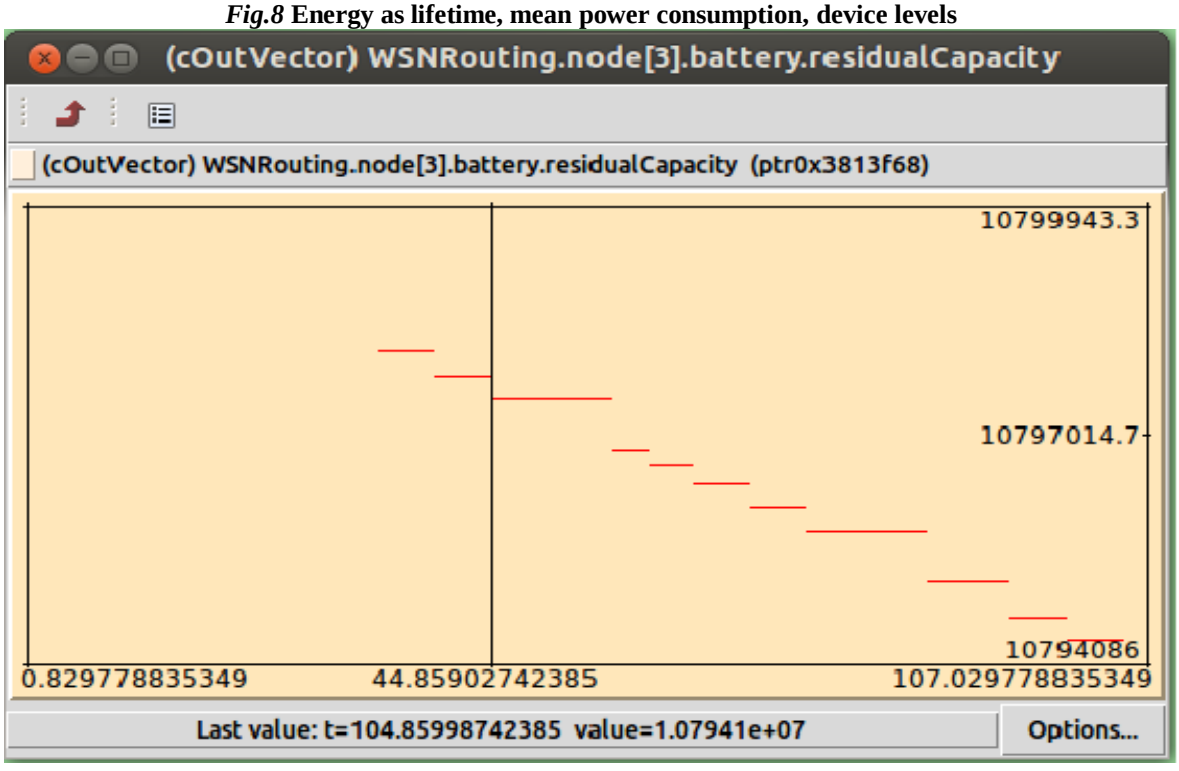


Fig.9 **Energy Level displays in the WSN simulation output of MIXIM during runtime**

The second aspect is, do we get the battery display during the simulation time to understand explicitly the node battery level.

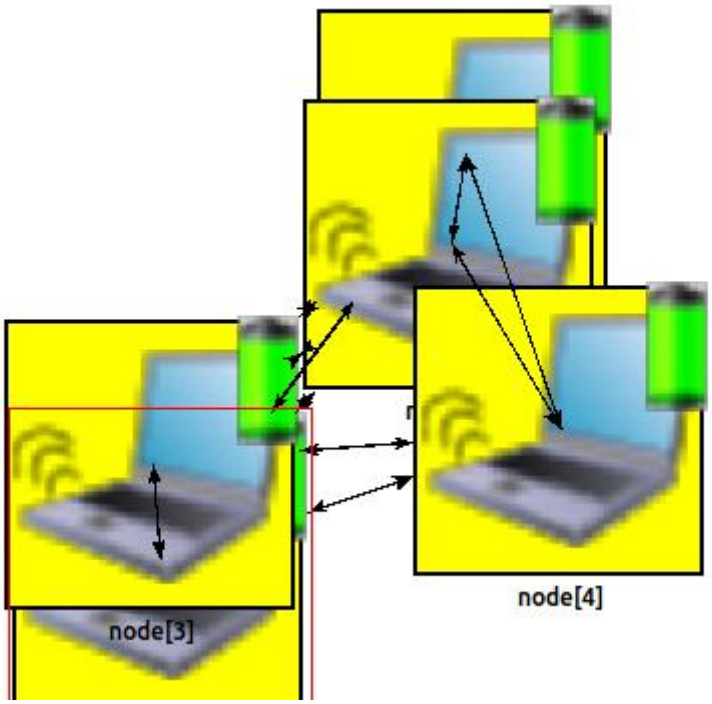


***Fig.10* Simulation run time battery display along with the node.**

VI. Conclusion And Future Work

As one of the network simulator pioneers, OMNETPP supports most of the energy relevant study in wireless sensor network. Even it is discrete C++ based simulator it runs in the Java environment mostly. There are special coding rules that obey the Java code makes it tougher for the developer to worry about the JSWING wrappers and interfaces. The battery module configuration is defined as part of the network configuration or property settings in the palettes of NED editor, where as the output generation is not through GUI completely, it needs write commands and data set definition and analysis files support. There is no well defined stack of protocols and algorithms to choose as such. The porting of the algorithms also is not as simple as the ALGOSIM. More flexible and portable application models are to be developed.

ACKNOWLEDGEMENTS

An acknowledgement section may be presented after the conclusion, if desired.

REFERENCES

[1] Aditi P Khadilkar (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 2 (3) , 2011, 1154-1159
[2] Andras Varga, Simutools '08 Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops, Article 50, ISBN: 978-963-9799-20-2
[3] H. Karl and A. Willig. Protocols and Architectures for Wireless Sensor Net-works. John Wiley and Sons Ltd, 2006.
[4] Ian F. Akyildiz and Mehmet Can Vuran. Wireless Sensor Networks. John Wiley and Sons Ltd, 2010.
[5] I. Mahgoub and M. Ilyas. Sensor Network Protocols. Taylor and Francis Group, 2006.
[6] Kazem Sohraby, Daniel Minoli, and Taieb Znati. Wireless Sensor Networks, Technology, Protocols, and Applications. Wiley Interscience Publications, 2007.
[7] Sudip Misra, IsaacWoungang, and Subhas Chandra Misra. Guide to Wireless Sensor Networks. Springer-Verlag London Limited, 2009.
[8] Waltenegus Dargie and Christian Poellabauer. Fundamentals of Wireless Sensor Networks, Theory and Practice. John Wiley and Sons Ltd, 2010.
[9] www.omnetpp.org/doc/omnetpp/UserGuide.pdf
[10] www.omnetpp.org/doc/omnetpp/Manual.pdf
[11] http://www.sics.se/~lmfeeney/software/energyframework.html