This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2016.2570220, IEEE Internet of Things Journal

Pymote 2.0: Development of an Interactive Python Framework for Wireless Network Simulations

Farrukh Shahzad, *Member, IEEE*

***Abstract*— Wireless sensor networks (WSNs) are utilized in various applications and are providing the backbone for the new pervasive Internet, or Internet of Things. The development of a reliable and robust large-scale WSN system requires that the design concepts are checked and optimized before they are implemented and tested for a specific hardware platform. Simulation provides a cost effective and feasible method of examining the correctness and scalability of the system before deployment. In this work, we study the performance of Pymote, a high level Python library for event based simulation of distributed algorithms in wireless ad-hoc networks. We extended the Pymote framework allowing it to simulate packet level performance. The extension includes radio propagation, energy consumption, mobility and other models. The extended framework also provides interactive plotting, data collection and logging facilities for improved analysis and evaluation of the simulated system.**

Keywords—Wireless sensor network, Simulation, Python, Distributed event modeling.

I. INTRODUCTION

Wireless Sensor Networks (WSN) have been employed in many applications such as intrusion detection, object tracking, industrial/home automation, smart structure and several others. The development of WSN system requires that the design concepts are checked and optimized using simulation [1]. The simulation environment for WSN can either be an adaptive development or a new development. The adaptive development includes simulation environments that already existed before the idea of WSNs emerged. These simulation environments were then extended to support wireless functionality and adapted for the use with WSNs. In contrast, new developments cover new simulators, which were created solely for simulating WSNs, considering sensor specific characteristics from the beginning [2].

Recently, several simulation tools have appeared to specifically address WSNs, varying from extensions of existing tools to application specific simulators. Although these tools have some collective objectives, they obviously differ in design goals, architecture, and applications abstraction level [1] [2][3][4].

F. Shahzad is with Information and Computer Science department, King Fahd University of Petroleum and Minerals, Dhahran 31261, Saudi Arabia (e-mail: farrukhshahzad@kfupm.edu.sa)

Simulators can be divided into three major categories based on the level of complexity: algorithm level, packet level, and instruction level. Our work is based on Pymote [5] which an algorithm level simulator. Some other algorithm level simulators are NetTopo[6], Shawn[7], AlgoSensim[8] and Sinalgo[9].

In this work, our main contribution is the design and implementation of packet level modules for propagation, energy consumption and mobility models to extend the Pymote framework. Secondly, we add graphing and data collection modules to enhance the Pymote base functionality and modify the existing modules for node, network, algorithm and logging.

We believe our work is different from other commonly used more extensive tools like NS-3[4] because of ease of use (faster learning), portable operating system independent code and built-in support for plots, graphs and JavaScript based interactive charts. We feel that Pymote is better suited for faster prototyping and validation of high-level algorithms.

The rest of the paper is organized as follows: In section II, we provide some background related to existing simulation tools and we discuss the python based simulation tool Pymote. We present our extension in section III and provide simulation examples and analysis in section IV. We conclude in section V.

II. BACKGROUND AND RELATED WORK

Simulation has always been very popular among network-related research. A large number of simulators have been proposed in literature in which algorithms for wireless ad hoc networks can be implemented and studied. These simulators have different design goals and largely vary in the level of complexity and included features. They support different hardware and communication layers assumptions, focus on different distributed networks implementations and environments, and come with a different set of tools for modeling, analysis, and visualization. Classical simulation tools include NS-2/NS-3, OMNeT++, J-Sim, OPNET, TOSSIM, and others [2][3][4].

Pymote is a high level Python library for event based simulation of distributed algorithms in wireless networks [5]. The library allows the user to make implementation of their ideas using Python—a popular, easy to learn, full featured, object oriented programming language. Functionalities provided by the library are implemented without additional layer of abstraction, thus harnessing full power of Python's native highly expressive syntax. Using the library, users can quickly and accurately define and simulate their algorithms. The library particularly focuses on fast and easy implementation of ideas and approaches

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2016.2570220, IEEE Internet of Things Journal

at algorithm level without any specification overhead using formally defined distributed computing environment.

III. Extending The Functionality

One of the primary reason to extend Pymote is to add support for packet-level simulations for protocols such as MAC. The prime role of the MAC is to coordinate access to and transmission over a medium common to several nodes. A multitude of MAC protocols for WSNs have been proposed in literature, and an exhaustive list can be found in [10][11][12].

A. Propagation Model

We implemented two basic radio propagation models and the commonly used shadowing model for WSN in the Pymote framework. These models are used to predict the received signal power of each packet. At the physical layer of each wireless node, there is a receiving threshold (P_RX_THRESHOLD). When a packet is received, if its signal power is below the receiving threshold, it is marked as error and dropped by the medium access control (MAC) layer. The free space propagation model assumes the ideal propagation condition that there is only one clear line-of-sight path between the transmitter and receiver, while the two-ray ground reflection model considers both the direct path and a ground reflection path which gives more accurate prediction at a long distance than the free space model. However, in reality, the received power is a random variable due to multipath propagation or fading (shadowing) effects. The shadowing model consists of two parts: path loss component and a Gaussian random variable with zero mean and standard deviation σ_{DB}, which represent the variation of the received power at certain distance. Table I lists parameters available for propagation module. The propagation model type (free space, two-ray ground or shadowing) is a network level attribute, which is selected before starting the simulation.

B. Energy consumption Model

In our extended framework, the energy model object is implemented as a node attribute, which represents the level of energy in a node. Each node can be configured to be powered by external source (unlimited power), Battery (default) or energy harvesting (EH) sources. The energy in a node has an initial value which is the level of energy the node has at the beginning of the simulation. It also has a given energy consumption for every packet it transmits and receives which is a function of packet size, transmission rate and transmit (receive) power. The model also supports idle or constant energy discharge due to hardware/ microcontroller consumption and energy charge for energy harvesting based WSN. During simulation, each node's available energy is recomputed every second based on the charging and/or discharging rate. If it drops below minimum energy required to operate (E_{min}) then that node assumed to be dead (not available for communication) until energy reaches above E_{min} again later in simulation (for EH nodes). Table II lists parameters available for energy module which can be set differently for each node. The energy object keeps track of the energy available (for battery-operated or energy harvested nodes) and total energy consumption.

C. Mobility Model

Our extended framework allows nodes to be mobile during simulation. Each node can be configured as fixed or mobile. The mobility module support three types of motion as summarized in Table III [19][20]. During simulation, each mobile node location is recomputed every second.

D. Plotting and Data collection

These modules allow real-time plotting and data collection during and after simulation for interactive analysis and comparisons of useful information. Strictly speaking, these modules are not really extension of Pymote, but implements generic helper methods by utilizing the python Matplotlib package [21]. The simulation script is responsible for utilizing these methods to plot/chart and collect/log appropriate information as required by the simulated algorithm and application scenario. The output files are managed by utilizing separate folder for each type of files within the current working path (Table IV). Also for each simulation run, a separate folder, prefixed with the current date time is used for all files created during that simulation run. The output format includes CSV, PNG, and high quality SVG and PDF which can directly by inserted into Latex and other publishing applications. HTML files are also created with embedded JavaScript for interactive plotting needed for presentation and online content. It utilizes the innovative charting library provided by Highsoft [22], which is free to use for personal and academic purposes.

E. Modified Node module

Enhanced framework requires significant modification in the Node module. The Node object now contains node type, energy model object and mobility object. The modified send and receive methods check before transmission or reception whether node has enough energy to perform the operation. Also the propagation model dictates whether a packet is received without errors (i.e. when received signal power is greater than the threshold based on the distance between the sender and receiver nodes). The object also keeps track of number of messages transmitted, received, or lost.

F. Open Source

The extended framework is available as open source at https://github.com/farrukhshahzad/pymote2.0, which is based on the original repository (https://github.com/darbula/pymote). However, we realized that the modification in terms of adding new modules and changes in existing modules will make it backward incompatible. So we decided to call it 'Pymote 2.0' and have a separate repository. This way, user have the option to stay with original code base (latest release 0.0.2) or use the extended 2.0 release. Apart from standard Python libraries, some scientific computing libraries such as NumPy, SciPy and Networkx are required.

G. Topology Generator

As part of the extension, we have developed an interactive topology generation module. Listing 1 shows a script to generate a simple grid topology. Notice that only 81 nodes are created instead of desired 90 because 9 nodes were created outside the desired 100 m x 100 m area due to randomness factor of 0.5. Fig. 1 shows the topology generated by script of listing 1.

Listing 1: A Simple script to generate a topology

```
from pymote import *
from pymote.conf import global_settings
```

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2016.2570220, IEEE Internet of Things Journal

```
from toplogies import Topology
from pymote.utils.filing import get_path, \
   TOPOLOGY_DIR
from numpy.random import seed
from networkx import *

seed(100)  # to get same sequence for each run
# Network/Environment setup (100 m x 100 m)
global_settings.ENVIRONMENT2D_SHAPE = (100, 100)
global_settings.COMM_RANGE = 10  # comm. radius

net = Network()
h, w = net.environment.im.shape
n = 90  # total no of nodes (desired)
x_radius = 0.9*global_settings.COMM_RANGE
y_radius = 0.9*global_settings.COMM_RANGE
# Topology setup
# Generate a randomized grid network with
# 10% anchors (default)
net_gen = Topology(n_count=n, connected=True)
net = net_gen.generate_grid_network(name="Grid",
                    randomness=0.5)
# Get actual of number of nodes and Avg. degree
avg_deg = net.avg_degree()
na = net.__len__()
print "No. of Nodes: %s" % na
print "Avg. Degree: %s" % round(avg_deg, 3)
filename = "Simple Grid - %s nodes" %na
net.name = filename
# saving topology as PNG image
net.savefig(fname=get_path(TOPOLOGY_DIR, filename),
        title=net.name,
        xlabel="X-coordinate (m)",
        ylabel="Y-coordinate (m)",
        show_labels=True, format="png")

# display info about the network
print networkx.info(net)
# Save the network as json
net.save_json(get_path(TOPOLOGY_DIR,
        filename+".json"), scale=(6, 5))
net.reset()
```

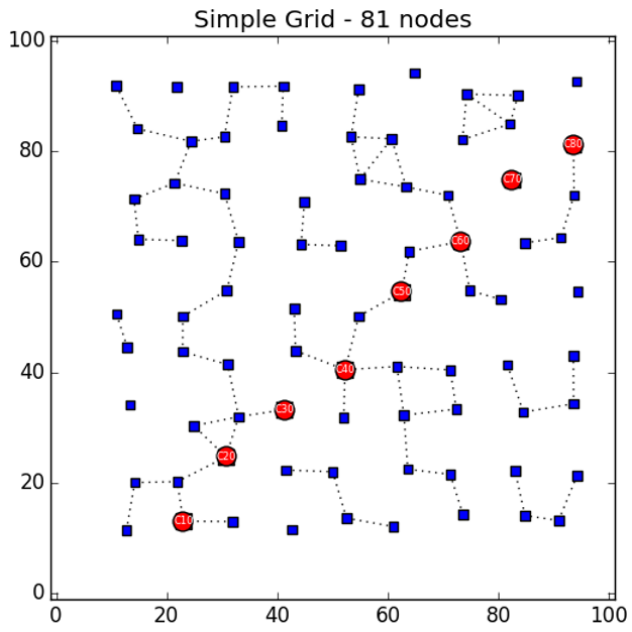

Fig. 1. A simple randmized grid output of Listing 1.

TABLE I. PROPAGATION MODEL PARAMETERS

Description	Parameter	Default
Transmit Antenna gain	G_TX	1
Receive Antenna gain	G_RX	1
System Loss (>=1.0)	L	1
Min. Received signal power threshold	P_RX_THRESHOLD	-70 dbm
Frequency	FREQ	2.4 Ghz
Path loss exponent	BETA	2.5
Gaussian noise standard deviation	SIGMA_DB	4 dbm

TABLE II. ENERGY MODEL PARAMETERS

Description	Parameter	Default
Transmit power	P_TX	84 mW
Receive power	P_RX	73 mW
Initial Energy	E_INIT	2.0 Joules
Min. Energy required for operation	E_MIN	0.5 Joules
Charging rate (EH nodes)	P_CHARGING	2 mW/sec
Discharging rate	P_IDLE	0.1 mW/sec
Transmission rate	TR_RATE	250 kbps

TABLE III. MOBILITY PARAMETERS

Type	Parameters	Default
0: Fixed	None	
1: Mobile (uniform velocity)	VELOCITY HEADING	20 m/s 45 deg
2: Mobile (uniform velocity, random heading)	VELOCITY	20 m/s
3: Mobile (random motion)	MAX_RANDOM_ MOVEMENT	30 m

TABLE IV. FILE MANAGEMENT

Type	Folder Name	Examples
Data files	/data	CSV files containing energy consumption for each node, Message received/lost counts, etc.
Charts/plots	/charts	Line plots and/or bar charts of energy levels.
Topology	/topology	Topology map of all nodes used for simulation (before/after simulation)
Logging	/logs	Simulation run and module level logging
Combined	/yyyy-mm-ddThh-mm-ss	All files generated during a specific run

IV. SIMULATION EXAMPLE

We consider Internet of Things (IoT) application scenario where an energy harvesting WSN (EHWSN) node is installed/embedded within the 'Thing' (object that need to be monitored) [13][14]. Several of such objects with EHWSN nodes form a cluster (in virtual star topology) around a high power coordinator node (or cluster head). EHWSN nodes can only communicate to its own coordinator (when they have

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2016.2570220, IEEE Internet of Things Journal

enough energy). Coordinators are special wireless nodes which have sufficient power available and can send data to base station directly or via other coordinators (multi-hop) in a typical converge-cast application as illustrated in Fig. 2. These objects are mobile and can move around its neighborhood or move to another neighborhood (within the range of a different coordinator). The coordinators are installed at strategic fixed locations throughout the facility [15][16][17].

Fig. 3 illustrates the scheme on time scale and can be summarized as follows:

- The coordinator periodically (period = $T_c = t_4 - t_0$) broadcasts a beacon pulse with 10% duty cycle. The pulse contains the MAC address (48 or 64 bytes) of the radio, which is universally unique, and the number of registered nodes. After transmitting the beacon, it goes in the listening mode to receive messages from any EHWSN mode which have any packets to send.
- The neighboring EHWSN nodes (which have enough power to operate) periodically wake up (period = $T_n > T_b$) with a certain duty cycle to receive the beacon pulse from the coordinator (t_1 in Fig. 3). If the received coordinator's MAC address is different than the last communicated coordinator or the node has not communicated recently, then the node will send a registration message containing its MAC address and the power status (t_2 in Fig. 3); otherwise node will go back to sleep or send the data packet if any. The destination address will be set to the coordinator address so that any other neighboring nodes which are listening will ignore it.
- On receiving the node' registration message, the coordinator record the registration information and increment the number of registered nodes (t_3 in Fig. 3). If coordinator receives a data message then it will buffer it for future transmission to base station or for aggregation. The coordinator will acknowledge the received packet in both cases.
- The case of multiple child nodes transmitting in response to the same beacon pulse is also shown in Fig. 2 during the second beacon period. The contention is avoided by using a random back off time before transmission. The wining node will transmit first (node A transmits at t_7 in Fig. 3) and other nodes will wait for the channel (e.g. node B transmits at t_9 in Fig. 3).
- If this deployment is in a remote location then satellite technology can be utilized at base station to route data to a back office system [18].

A. Simulation setup

We only need to simulate the communication performance of one cluster formed by a coordinator and its children EHWSN nodes. The coordinator is placed in middle of *n* randomly deployed EHWSN nodes over a 600 m by 600 m area. We assumed that these nodes are constantly being charged during simulation and nodes are mobile (type=2, see Table III). We consider beacon, registration and data packet sizes of 100 bytes while the acknowledgment packet size of 15 bytes. We used default parameters for different modules as listed in Tables I-III. Some other parameters are shown in Table V. The simulation script utilizes the plotting and data collection modules to generate image and data files for easy visualization and analysis of simulation results (Fig. 4).

B. Simulation Results

Fig. 5 shows a simple topology generated for simulation using the Pymote. The center node (#1) acts as the coordinator for the EHWSN nodes (numbered 2 to 26). The node in lighter color means that its available energy is below E_MIN (=0.5 J).

We arbitrarily selected node 5 and node 10 as borderline in terms of energy available (i.e. the initial energy at start of simulation). Node 5 doesn't have enough energy to transmit in the beginning but charged up above E_MIN (Table II) during the simulation and start communicating. On the other hand, Node 10 just has enough energy to send few messages before its energy level dropped below E_MIN. We set the charging rate to 0 for Node 10. The energy level change during the simulation run is shown in Fig. 6.

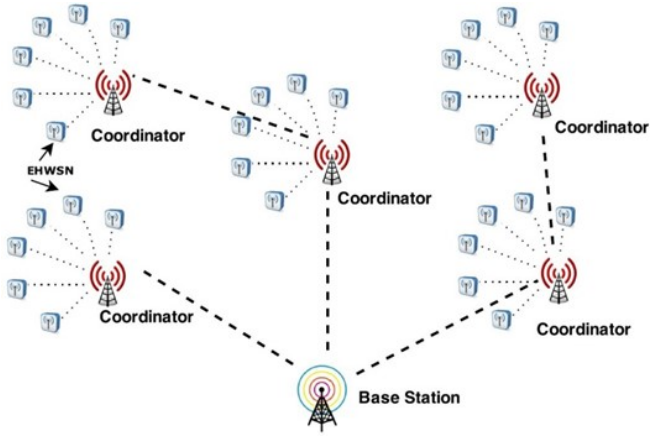

Fig. 2. Proposed deployment scheme for EHWSN system

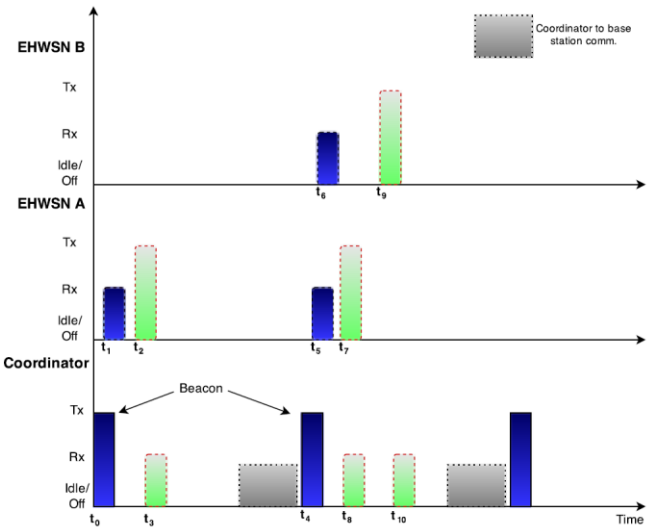

Fig. 3. Timing diagram of coordinator's beaconing and node's transmission

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2016.2570220, IEEE Internet of Things Journal

TABLE V. SIMULATION PARAMETERS

Parameter	Name	Value
T_b	Beacon period	1 sec
Δ_b	Beacon duty-cycle	10%
n	No. of nodes	5 - 100
S_d	Data packet size	100 bytes

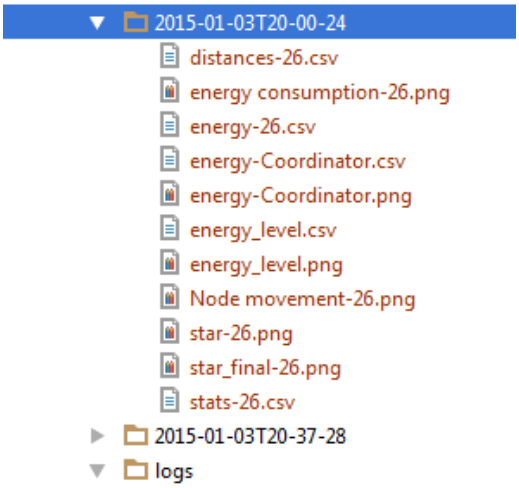

Fig. 4. Simulation output files

Fig. 7 shows the net node displacement during the simulation as they move around with constant speed but in random direction. Fig. 8 illustrates the energy consumption of all EHWSN nodes. We can notice that some nodes never communicated due to low energy (like node 2, 4, 11, 15, 20, 21 and 26) whereas node 5 and 10 were only active during some part of the simulation as we discussed earlier. Finally Fig. 9 shows the location of nodes at the end of simulation.

Secondly, we vary the number of EHWSN nodes in the network from 10 to 100 in the increment of 5. The extended framework generates simulation output files for each iteration. The output files also include the overall summary. Fig. 10 shows the generated topology for 100 EHWSN nodes (2 to 101). Fig. 11 shows energy consumption plots for coordinator and other nodes combined (sum of energy consumption for all EHWSN nodes). The chart also shows number of messages (packets) received and lost at coordinator for each iteration.

Table VI presented the overall simulation summary for all iterations. Node displacement and energy level change during the simulation for 100 nodes are shown in Fig. 12 and Fig. 13 respectively.

V. CONCLUSIONS AND FUTURE WORK

In this work, we utilized and extended the Python based *Pymote* framework to allow packet level simulation. We implemented modules for propagation, energy consumption and mobility models. We also added graphing and data collection modules to enhance the Pymote base functionality and modified existing modules for node, network, algorithm and logging to support the extended framework. Finally, we performed an example simulation for a scheme to efficiently utilize EHWSN in an IoT application. The simulation results presented include topology maps, plots for available energy, bar charts for node displacement and energy consumption and comparison of received and lost packets at the coordinator node.

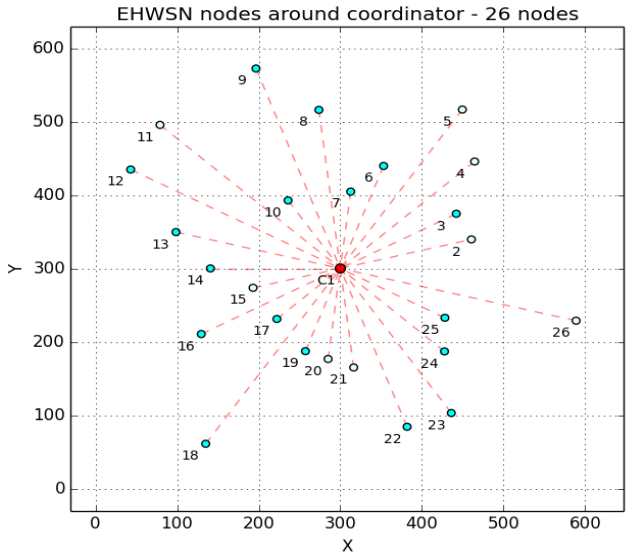

Fig. 5. 25 EHWSN nodes around a coordinator

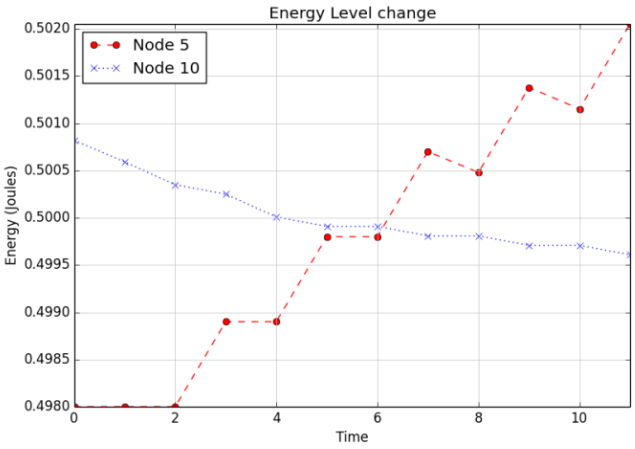

Fig. 6. Energy level change for Node 5 and 10

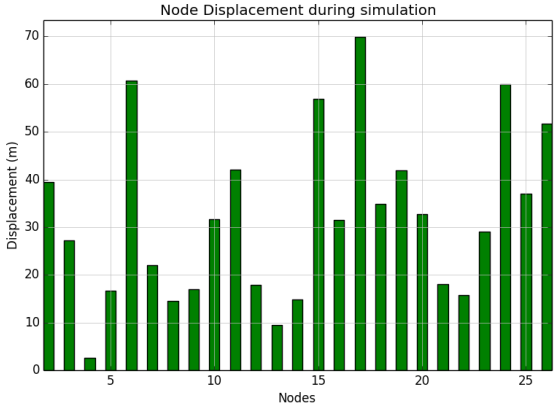

Fig. 7. Node displacement during the simulation

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2016.2570220, IEEE Internet of Things Journal

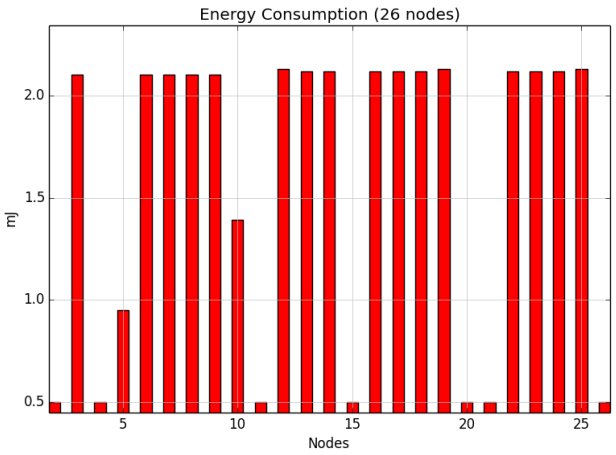

Fig. 8. Energy consumtion during the simulation

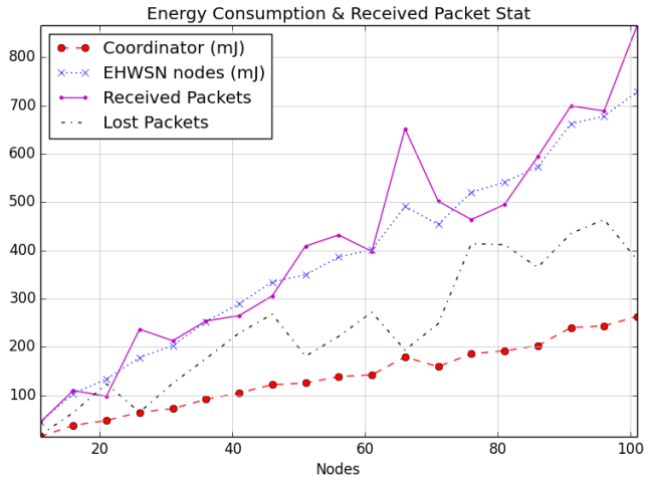

Fig. 11. Overall summary

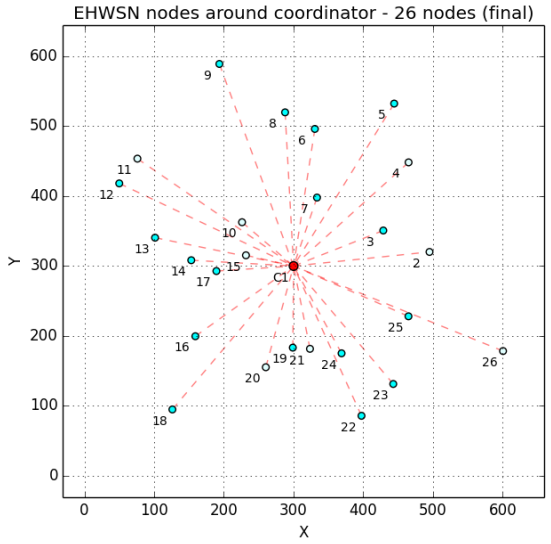

Fig. 9. Final position of nodes at the end of simulation

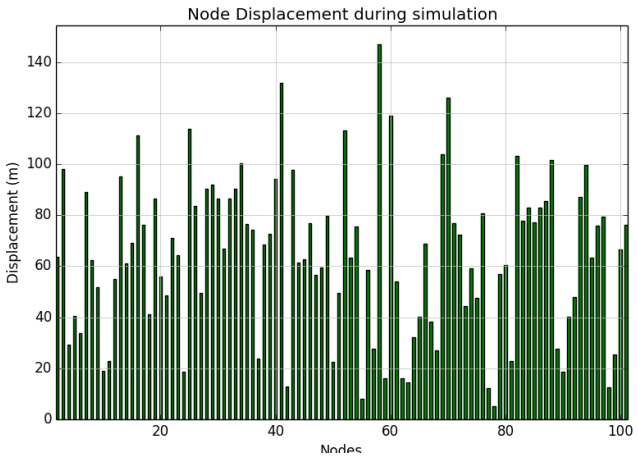

Fig. 12. Node displacement during the simulation for 100 nodes

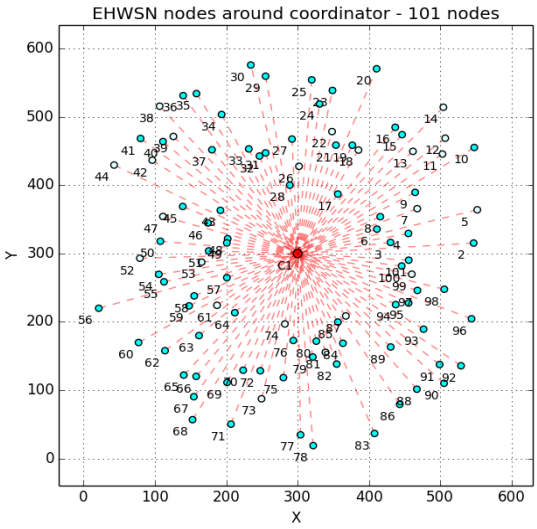

Fig. 10. 100 EHWSN nodes around a coordinator

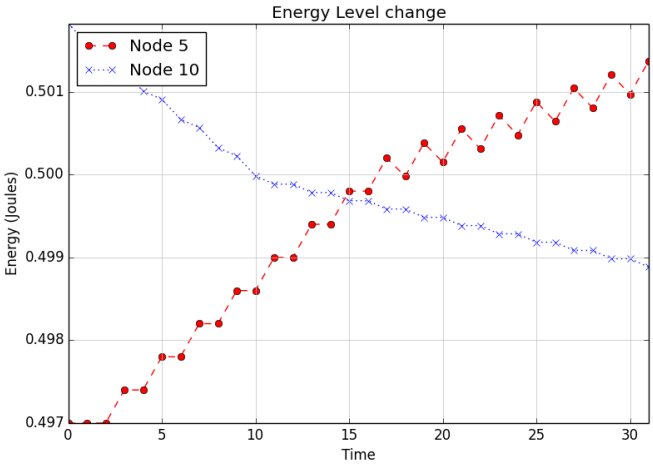

Fig. 13. Energy level change for Node 5 and 10 with network size = 100

TABLE VI. SIMULATION STATISTICS

Nodes	Coordinator Energy consumption (mJ)	Total EHWSN consumption (mJ)	Received Packets at Coordinator	Lost Packets at Coordinator
11	15	45	44	18
16	38	104	110	64
21	48	134	98	124
26	65	178	237	65
31	73	203	213	125
36	92	253	254	176
41	105	290	265	229
46	122	335	306	268
51	126	349	409	181
56	139	386	432	222
61	142	401	398	272
66	180	491	652	194
71	159	454	502	248
76	186	521	464	414
81	192	541	495	412
86	203	573	594	365
91	240	662	699	435
96	244	678	689	464
101	264	729	866	380

As a future work, we will continue building on our simulation framework to include analysis of other network algorithms like routing, clustering, optimization, data gathering, etc. Although the framework is extended to support low –level protocols, like MAC protocols, we or interested user still need to implement specific protocols like SMAC, BMAC, etc.

REFERENCES

[1] Abuarqoub, Abdelrahman, et al. "Simulation issues in wireless sensor networks: A survey." SENSORCOMM 2012, The Sixth International Conference on Sensor Technologies and Applications. 2012.

[2] Ali, Q., Abdulmaojod, A., Ahmed, H.," Simulation & Performance Study of Wireless Sensor Network (WSN) Using MATLAB, IJEEE Journal, 2010.

[3] Sobeih, A.; Hou, J.C.; Lu-Chuan Kung; Li, N.; Honghai Zhang; Wei-Peng Chen; Hung-ying Tyan; Hyuk Lim, "J-Sim: a simulation and emulation environment for wireless sensor networks," Wireless Communications, IEEE , vol.13, no.4, pp.104,119, Aug. 2006

[4] WNS3 '15: Proceedings of the 2015 Workshop on Ns-3, New York, NY, USA: ACM, 2015.

[5] Damir Arbula and Kristijan Lenac, "Pymote: High Level Python Library for Event-Based Simulation and Evaluation of Distributed Algorithms," International Journal of Distributed Sensor Networks, vol. 2013, Article ID 797354, 12 pages, 2013. doi:10.1155/2013/797354

[6] L. Shu, M. Hauswirth, H. C. Chao, M. Chen, and Y. Zhang, "Nettopo: a framework of simulation and visualization for wireless sensor networks," Ad Hoc Networks, vol. 9, pp. 799–820, 2011.

[7] A. Kroeller, D. Pfisterer, C. Buschmann, S. Fekete, and S. Fischer, "Shawn: a new approach to simulating wireless sensor networks," in Proceedings of theDesign, Analysis, and Simulation of Distributed Systems (DASD '05), San Diego, Calif, USA, 2005, https://github.com/itm/shawn.

[8] Algosensim, http://tcs.unige.ch/doku.php/code/algosensim/overview.

[9] Sinalgo, http://dcg.ethz.ch/projects/sinalgo/.

[10] P. Suriyachai, U. Roedig, and A. Scott. "A survey of MAC protocols for mission-critical applications in wireless sensor networks." IEEE Communications Surveys & Tutorials, vol. 14, no. 2 (2012): 240-264.Fukumoto, M. and Tonomura, Y. Body coupled Fin-geRing. In Proc of CHI 1997, 147-154.

[11] P. Huang, L. Xiao, S. Soltani, M. Mutka, and N. Xi. "The evolution of MAC protocols in wireless sensor networks: A survey." IEEE Communications Surveys & Tutorials, vol.15, no.1, pp.101-120, First Quarter 2013.

[12] A. Bachir, M. Dohler, T. Watteyne, and K. K. Leung. "MAC essentials for wireless sensor networks." IEEE Communications Surveys & Tutorials, vol. 12, no. 2 (2010): 222-248.

[13] Steve Grady and Jeff Mullin Cymbet, "The Design Secrets for Commercially Successful EH-Powered Wireless Sensors." Energy Harvesting for Powering WSN Symposia, 2014

[14] Zhi Ang Eu, Hwee-Pink Tan, Winston K.G. Seah, Design and performance analysis of MAC schemes for Wireless Sensor Networks Powered by Ambient Energy Harvesting, Ad Hoc Networks, Volume 9, Issue 3, May 2011, Pages 300-323, ISSN 1570-8705, http://dx.doi.org/10.1016/j.adhoc.2010.07.014.

[15] Nintanavongsa, P.; Naderi, M.Y.; Chowdhury, K.R., "Medium access control protocol design for sensors powered by wireless energy transfer," INFOCOM, 2013 Proceedings IEEE , vol., no., pp.150,154, 14-19 April 2013. doi: 10.1109/INFCOM.2013.6566753

[16] Basagni, Stefano, et al. "Wireless sensor networks with energy harvesting." Mobile Ad Hoc Networking: The Cutting Edge Directions (2013): 701-736.

[17] Seah, Winston KG, Y. K. Tan, and Alvin TS Chan. Research in energy harvesting wireless sensor networks and the challenges ahead. Springer Berlin Heidelberg, 2013.

[18] F. Shahzad, "Satellite monitoring of wireless sensor networks (WSNs)," Procedia Computer Science, vol. 21, no. 0, pp. 479 – 484, 2013.

[19] T. Camp, J. Boleng, and V. Davies, "A survey of mobility models for ad hoc network research," *Wirel. Commun. Mob. Comput.*, vol. 2, no. 5, pp. 483–502, Aug. 2002.

[20] G. Han, J. Chao, C. Zhang, L. Shu, and Q. Li, "The impacts of mobility models on DV-hop based localization in Mobile Wireless Sensor Networks," *J. Netw. Comput. Appl.*, vol. 42, pp. 70–79, Jun. 2014.

[21] J. D. Hunter, "Matplotlib: A 2D Graphics Environment," *Comput. Sci. Eng.*, vol. 9, no. 3, pp. 90–95, 2007.

[22] T. Høns, "Highcharts, Highstock and Highmaps documentation | Highcharts," Online, 2013. [Online]. Available: http://www.highcharts.com/docs. [Accessed: 30-Aug-2015].

Farrukh Shahzad (S`16) received the BE (Electrical Engg.) from the NED University of Engineering & Technology, Karachi, Pakistan in 1992 and the MSEE from King Fahd University of Petroleum & Minerals (KFUPM), Dhahran, Saudi Arabia in 1996. He recently completed his Ph.D.from KFUPM in 2016. In 1996, he moved to USA, where he gained 20 years field experience in product design, software development, engineering, and implementation of many M2M and satellite based remote monitoring systems. His current interests include Visualization and Simulation, Big Data analytics, Data sciences, Business intelligence, Machine learning and IoT/Wireless Sensor networks. He holds US copyright for four engineering software. His research activities resulted in publication of more than 15 technical papers in IEEE and other journals/proceedings.