Sustainable Computing: Informatics and Systems 9 (2016) 1–7

Contents lists available at ScienceDirect

Sustainable Computing: Informatics and Systems

journal homepage: www.elsevier.com/locate/suscom

Comparison of simulators for assessing the ability to sustain wireless sensor networks using dynamic network reconfiguration

Joel Helkey*, Lawrence Holder, Behrooz Shirazi

School of Electrical Engineering and Computer Science, Washington State University, Pullman, WA 99164-2752, USA

A R T I C L E   I N F O

Article history:
Received 11 July 2015
Received in revised form 2 January 2016
Accepted 8 January 2016
Available online 21 January 2016

Keywords:
Wireless sensor networks
Network simulation
Power consumption

A B S T R A C T

Wireless sensor networks (WSNs) can be used for a variety of tasks. The goal is to sustain such networks for as long as possible while still maintaining acceptable performance on the task. Evaluating WSNs for such diverse tasks precludes actual deployment and thus requires simulation. In this work, we compare simulators that can model power consumption and allow dynamic reconfiguration of network parameters based on feedback from the end application. The simulation results demonstrate some fundamental differences in out-of-the-box performance in terms of the way the simulators behave and calculate power consumption. With modifications to parameters and behavior, however the simulators' outputs become more closely aligned. Overall, based on our selection criteria, ns-3 was the best WSN simulator of those evaluated for assessing the sustainability of a WSN in a variety of settings.

1. Introduction

Network simulation, a commonly used approach in the design, implementation, optimization, and evaluation of network algorithms and protocols, models the behavior of a real network operating under various configurations. This allows researchers the ability to run experiments in a controlled, reproducible manner without the time and expense of setting up an actual network test bed with real nodes, links, and devices to measure network parameter. Moreover, the intended environment may be prohibitive to recreate at any budget. For example, the ultimate working destination of a sensor network may be at the bottom of an ocean [1] or on the side of an active volcano [2].

A wireless sensor network (WSN) is a network of low-power sensing devices that communicate by means of wireless transmission and in which every node is potentially a router [3]. Sensor nodes detect and report detailed data about their surrounding physical environment. Different types of sensors may be attached to the node to sample temperature, pressure, seismic activity, vehicular movement, and so on. Typically, a WSN has many sensor nodes located inside or close to the monitored area.

This comparison study is motivated by our research into ways to reconfigure a network to maximize application performance, as depicted in Fig. 1, while simultaneously sustaining the network for as long as possible. Our assumptions are: static network nodes (the sensors do not move after initial deployment), adjustable data sending rates (at a minimum on/off capability), and the application must be capable of reporting its performance in a meaningful way, such as accuracy on a scale of 0–100% for an AI application that does classification. So, we want to find the best open-source WSN simulator that can model power consumption and has the ability to evaluate algorithms for dynamically reconfiguring network parameters based on feedback from the end application.

This paper details the key features, advantages and/or disadvantages, and limitations of available WSN simulators with respect to their feasibility as a research tool. The simulators are tested in various scenarios, matching as closely as possible the parameters and behavior of each.

The rest of this paper is structured as follows. Section 2 presents related work. Section 3 discusses currently available open-source sensor network simulators. Section 4 analyzes features of the various available simulators. Section 5 details the simulator setup, and Section 6 states the results. Section 7 discusses the lessons learned. Finally, Section 8 contains some concluding remarks.

2. Related work

Energy conservation (for longer network lifetime) in a WSN is typically dealt with in five main areas [4–6]:

1. Optimal sensor node scheduling between active and sleep states.
2. Balancing transmission power for high connectivity and low energy consumption.
3. Data compression to reduce unnecessary packet transmissions.

* Corresponding author.
E-mail address: jhelkey@wsu.edu (J. Helkey).

http://dx.doi.org/10.1016/j.suscom.2016.01.003
2210-5379/© 2016 Elsevier Inc. All rights reserved.

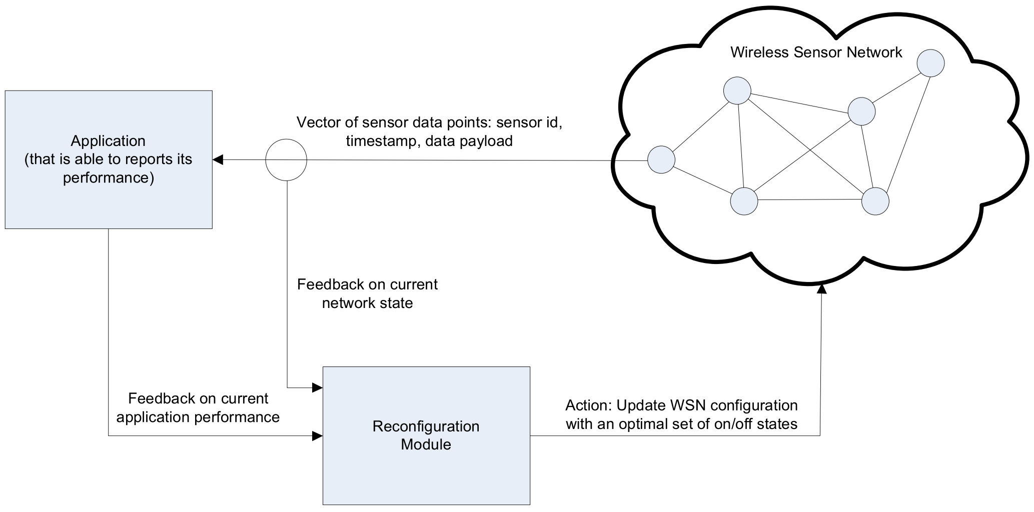
Wireless Sensor Network
Application (that is able to reports its performance)
Vector of sensor data points: sensor id, timestamp, data payload
Feedback on current network state
Feedback on current application performance
Reconfiguration Module
Action: Update WSN configuration with an optimal set of on/off states

Fig. 1. Dynamic reconfiguration.

4. Data aggregation and clustering for energy efficient routing of packets.
5. Efficient packet re-transmission and channel access protocols.

In this paper, we focus on the area of sensor node scheduling.

Papers that have addressed these same or similar issues surrounding WSN simulator selection can be roughly divided into two groups: papers that review/compare various simulators without actually using them, and papers that include a comparison of simulator results. The papers that compare simulators all fundamentally agree that there is no one best, general-purpose simulator for all situations.

2.1. *Comparison without simulator results*

Many papers just present reviews and comparisons of various simulators based on apparent advantages and disadvantages gained through a search of the literature, without actually using them. In Imran et al.'s study [7], an evaluation criteria included popularity, support for WSNs, actively maintained codebase, and the technical support available. They noted that simulators cannot promise complete accuracy of results due to their use of simplifying assumptions.

The evaluation criteria in the work by Jevtić et al. [8] were level of detail, timing, software license, popularity, platform, graphical user interface support, models and protocols, and energy consumption. They provide guidance regarding which simulator to use in different situations.

Khan et al. [9] identified limitations of simulators and investigated their suitability for large-scale WSNs. They compared simulators based on interface, accessibility and user support, availability of WSN modules, extensiblity, and scalability.

Korkalainen et al. [10] tried to estimate the suitability of simulators for high-performance network planning and verification. They accomplished this by reviewing and summarizing the simulators' capabilities, especially considering popularity, support for WSN, and active maintenance.

Musznicki and Zwierzykowski [11] classified simulators according to their features and main applications. Singh et al. [12] sought to help researchers in selecting an appropriate simulator by presenting their best and worst features.

2.2. *Comparison with simulator results*

Regarding work that compares simulator results, Sundani et al. [13] performed a comparison based on scalability and level of abstraction. In addition, they reported a case study running ns-2 and TOSSIM with an increasing number of nodes and measured computation time. TOSSIM performed better than ns-2, as it appeared to follow a linear curve with large numbers of nodes.

In Feeney's study [14], results from five OMNet++ based simulators were compared with regard to backoff and contention. Her contribution was to introduce a scenario framework for comparing results from different simulators, with the idea that common testing will lead to better quality of and confidence in simulation results.

Weingartner et al. [15] compare simulator run-time performance and memory usage, showing large differences between the various simulators. However, they just presented these differences without identifying reasons the simulators might differ.

Bergamini et al. [16] found that using simulators in their default configurations would likely produce unreliable results. They compared simulator results with a real testbed in an effort to measure accuracy, determining that simple tuning can significantly increase accuracy.

The work closest to ours, Stetsko et al. [17] obtained calibration data using two MICAz sensor nodes and used two simulated nodes in comparing simulated energy consumption. They noted that even though the simulators were setup similarly, their results differed considerably. However, they only speculated on possible reasons for the differences and did not investigate further.

In contrast, our work simulates large numbers of nodes with forced routing due to large grid size and relatively short sensor transmission range. We experiment with the most commonly used simulators, including ns-3, a popular simulator primarily targeted for research use. In addition, we were able to identify the main reasons for differences in simulation results, and we share some important lessons learned from using the simulators.

3. WSN simulators

The WSN simulators included for consideration here are Castalia, TOSSIM, and ns-3. These are open-source simulators current research papers are found to use. OPNET is a famous network simulator A couple of older simulators, J-Sim [18] and SENSE

[19,20], were briefly considered for inclusion; however, due to their dependence on obsolete libraries/compilers and lack of active development, we decided not to include them. In addition, since we only considered open-source simulators for evaluation, commercial simulators such as OPNET [21] were not considered.

3.1. Castalia

Castalia (pronounced Cast-Uh-Lee-Ah) [22,23] is an open-source, discrete event simulator for WSNs based on the OMNeT++ platform [24]. It was developed at the Networks and Pervasive Computing program at National ICT (Information and Communications Technology) Australia since 2006 and was made public in 2007 under an Academic Public License. Castalia uses C++ and OMNET++ Network Description Language (NED). The NED language allows users to declare modules, connect them, and assemble them into more complex compound modules. Castalia was specifically designed to handle modification and extension. T Its main features include: a channel model derived from empirically measured data, a radio model for low-power communication based on real radios, flexible physical process modeling, a model of clock drift in the nodes, and and the implementation of several MAC and routing protocols.

For routing, the two protocols offered are BypassRouting and MultipathRingsRouting. BypassRouting does not implement any routing. MultipathRingsRouting is a non-standard routing protocol. On setup, it assigns each node a ring number, with the sink considered ring zero. When the simulator starts up, the sink sends out a setup packet containing the number zero. A node that receives a setup packet adds 1 to the ring number and then retransmits the packet. This process continues until all connected nodes have a ring number. When a node sends a packet to the sink, it broadcasts the packet with its own ring number. Any node that receives the transmission and has a lower ring number will rebroadcast the packet. This process repeats until the packet reaches the sink, assuming there is a fully connected path to the sink and the packet does not get dropped along the way.

Sensor nodes are implemented as composite modules, consisting of submodules for application, a mobility manager, routing, MAC, a radio, and a resource manager (used to keep track of energy spent by the node). All of the modules have many parameters that can be modified to support custom behavior. The modules use the OMNeT++ NED and C++ languages to implement their functionality. Castalia was designed to provide a generic framework for validating an algorithm before implementation in a real system. The current release version of Castalia is 3.2 (released 3/30/2011).

3.2. TOSSIM

The TinyOS mote simulator (TOSSIM) [25,26] is an open-source, discrete event WSN simulator, that was developed at the University of California at Berkeley. TOSSIM uses NesC [26], which is an extension of the C programming language.

TOSSIM was built with the requirements of scalability, completeness, fidelity, and bridging. Scalability means being able to scale up to simulating large numbers of nodes. Completeness means being able to simulate as many interactions between system components as possible. Fidelity means capturing the fine-grained TinyOS behavior, down to the bit level. Bridging means making a connection between implementation and algorithm by allowing the simulation to test the actual code that will ultimately run on the real hardware.

The TOSSIM architecture comprises five parts: capability for compiling components into the simulation, an event queue, hardware abstractions, radio models, and communication services for external programs. The communication service allows a user to connect to the simulator over a TCP socket in order to start, monitor, or debug a simulation. Settings can be queried and set. The user can also obtain other information, such as application events or packet receptions and transmissions.

TOSSIM does not model the power consumption of the motes. However, PowerTossimz [27] extends the port of PowerTOSSIM v2.0 for the MICAz mote and has a battery model that simulates the nonlinear behavior (rate capacity effect, recovery capacity effect) of modern batteries.

3.3. Ns-3

Ns-3 [28] is an open-source, discrete event network simulator of a WSN. Ns-3 was built as a replacement for the well-known ns-2 network simulator, with development beginning in 2006. The project's goal was to make available an extensible network simulation platform for education and research. It has models of how networks operate and provides a simulation engine for running customized simulation experiments. The simulation scripts can be written in either C++ or Python.

Currently, ns-3 is actively developed and has an impressive number of validated models of protocols, devices, and real-world objects. Implemented modules include several routing modules (Ad Hoc On-Demand Distance Vector, Destination-Sequenced Distance Vector, Dynamic Source Routing, and Optimized Link State Routing), framework for modeling energy consumption, a module for mobile nodes, modules for propagation loss and propagation delay, modules to support wireless networks at the channel and physical levels, and modules for tracing at different levels of granularity and customized data collection. Modeling of radio sleep state is not currently implemented; however, a workaround for this will be explained later in this paper.

4. Comparison of WSN simulator features

In Table 1, we compare simulator features. The criteria used for selection concern the ability to model radio sleep state, power consumption, computational performance (the simulation run time including power-consumption calculations) and programmability (allowing for reconfiguration while the simulator is running as well as integration with end applications).

Castalia and TOSSIM can put the radio to sleep from code running on the node. With modification, Ns-3 can closely model this behavior by changing the idle current consumption based on the desired state. Note that the simulator will still route packets to the sink and change radio-state accordingly. So, Castalia and Tossim have a slight edge based on this criterion.

All three simulators can measure power consumption. Castalia uses only radio state and baseline power consumption, which is a constant power draw no matter what state the radio is in. Ns-3 by default only uses radio state to determine power consumption and includes linear along with non-linear battery models. TOSSIM does not model power consumption, but by using PowerTossimz [27], a Python script that is run as a post-processing step, it is possible to get power consumption numbers. PowerTossimz has a battery model that simulates the nonlinear behavior (e.g., rate capacity effect, recovery capacity effect) of modern batteries; it also models radio state and CPU usage.

Regarding programmability, TOSSIM can interact with an end application and can dynamically change radio state, but PowerTossimz cannot dynamically change the fixed power-consumption parameters during the simulation, which is a significant limitation. Both Castalia and ns-3 can both interact with an end application and dynamically change their power-consumption parameters during a simulation. However, of the two, ns-3 is much easier to program.

Table 1
Features of WSN simulators.

Simulator	Radio sleep state	Model power consumption	Computational performance	Programmability
Castalia	✓	✓	Good	Fair
TOSSIM	✓	✓ with PowerTOSSIMz	Poor	Poor
ns-3	✓ with modifications	✓	Good	Good

For example, ns-3 has a simple class for a device energy model where current drain can be set by the user. In Castalia, it is possible to do the same thing, but users will have to implement, integrate, and validate the class themselves.

5. Simulation setup

In order to achieve routing in the nodes, a grid-shaped topology was selected with 10-m spacing between horizontal and vertical nodes. The range of node transmission was limited such that only the immediately adjacent horizontal and vertical nodes could communicate. Simulations were run on 4×4, 8×8, 12×12, and 16×16 grids with total node counts equaling 16, 64, 144, and 256 respectively. The sink node was always node zero, located in the bottom-left corner, as shown in Fig. 2. For each simulation run, all simulators used the same communication topology.

Our experiments were limited to 256 nodes, due to the long run-times associated with PowerTOSSIMz in calculating power consumption. We hypothesize that up to 1000 nodes the behavior of the WSNs performance and the simulators' ability to simulate such large WSNs would be essentially similar to the graphs we show here.

The application on the sink node only listened for data packets and, once per minute, broadcast a reconfiguration packet that contains the on/off status for each node. It was assumed to be plugged into an electrical grid and was therefore not included in the remaining energy summation. The test application running on the remote nodes was designed as follows. If the node was on, it would send one data packet per minute and be available to forward packets. If the node was off, it would not send any data packets and would not be available to forward packets. All remote nodes listened for reconfiguration packets for 3 s out of every minute, at a synchronized time in order to coordinate delivery.

Castalia specific settings: In Castalia, the network topology was automatically generated by specifying a grid layout, the number of nodes, and the size of the grid. With this information, each node was assigned *x*, *y* coordinates. Eq. (1) defines a shadowing model called log normal that was used to calculate path loss (or path attenuation) between nodes given the distance between them and some predefined parameters:

$$PL(d) = PL(d_0) + 10 \cdot \eta \cdot \log\left(\frac{d}{d_0}\right) + X_\sigma \tag{1}$$

where $PL(d_0)$ is the path loss at reference distance d_0, η is the path loss exponent (the rate at which signal strength decreases with distance), and X_σ is a zero-mean Gaussian distributed random variable with standard deviation σ [23]. For links that are bidirectional, Castalia allows each direction to have different path loss. There is an additional Gaussian zero-mean random variable with bidirectional sigma that defines the path loss for bidirectional links.

We chose a simple unit disk model in which receiving nodes at a certain distance from a transmitter node got the same strength signal and all links were bidirectional. To achieve this, the sigmas (sigma and bidirectional sigma) were set to zero and the radio model is set to IDEAL. By setting the collision model to zero, moreover, no collisions would happen (see Section 7 for more detail).

To set the range of the node transmissions so only the immediately adjacent horizontal and vertical nodes could communicate, $PL(d_0)$ was set to 70 [23].

The default initial energy for Castalia of 18,720 J was modified slightly to equal the energy used in TOSSIM. Since TOSSIM only takes an integer value for the initial energy in milliampere-hour (mAh) and 18,720 J in Castalia is equivalent to 1733.3333 mAh in TOSSIM, the initial energy in TOSSIM is truncated to 1733 mAh. The equivalent initial energy for Castalia was 18,716.4 J.

In TOSSIM, node connectivity is modeled as a directed graph, where vertices are nodes and edges are links. Each edge specifies the gain (when a source transmits, the destination receives a packet attenuated by this value) between nodes, and node coordinates are not considered. However, TOSSIM includes a Java program called LinkLayerModel that calculates the gains between nodes based on a specified topology and a wireless channel model. Our topologies were generated using the LinkLayerModel program with the following parameters for grid topology (set to the same Castalia values to achieve similar link gains), with 16, 64, 144, and 256 nodes:

```
PATH_LOSS_EXPONENT = 2.4;
SHADOWING_STANDARD_DEVIATION = 0.0;
PL_D0 = 70.0;
D0 = 1.0;
NOISE_FLOOR = −100.0;
WHITE_GAUSSIAN_NOISE = 0;
S11 = 0.0;
S22 = 0.0;
GRID_UNIT = 10.0;
```

For ns-3, the RangePropagationLossModel was used, with maximum radio range set to 11 m. The GridPositionAllocator was used to create a grid of nodes 10 m apart. Since the nodes were statically positioned and always have a route, we used static routing. The initial node energy was set to 18716.4 J, as for the other simulators, and the default supply voltage of 3.0 V was used. Initial simulations were run using default currents of 17.4 mA (tx), 19.7 mA (rx), and 426 μA (idle).

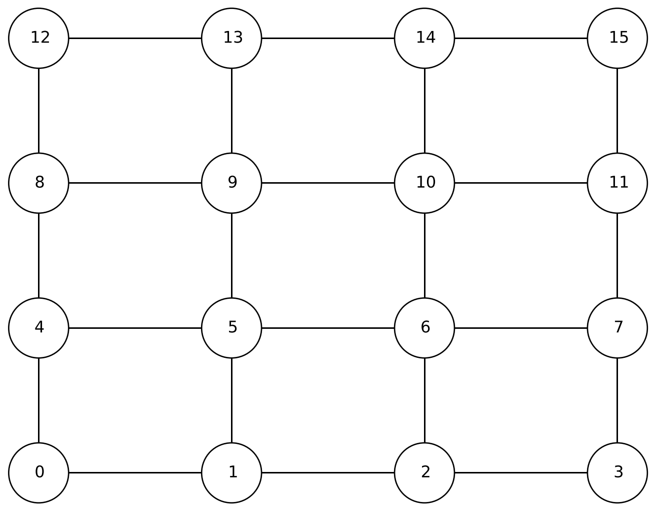

Fig. 2. Grid layout with 16 nodes.

6. Simulation results

For comparing the simulators, four realistic scenarios were selected:

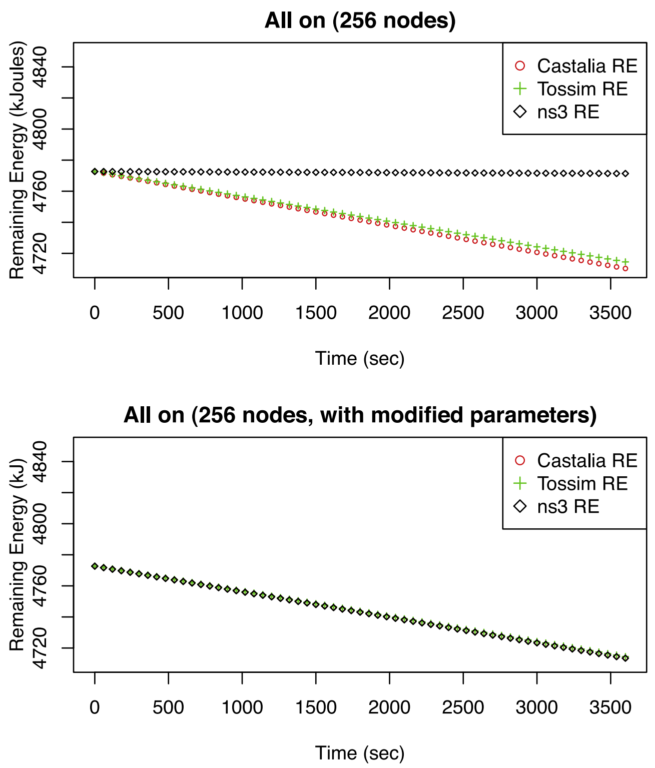

Fig. 3. All nodes on scenario with 256 nodes.

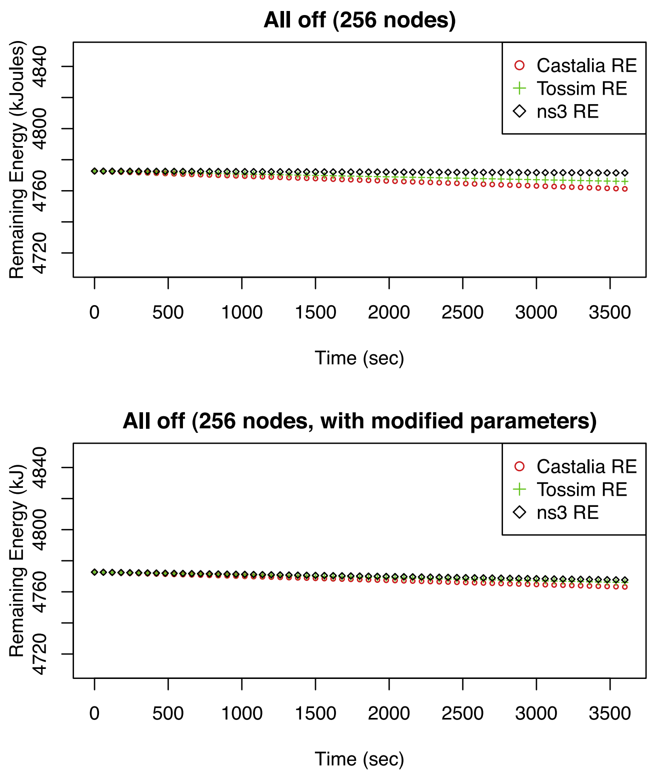

Fig. 4. All nodes off scenario with 256 nodes.

1. Turn all nodes on and leave them on the entire time.
2. Turn all nodes off and leave them off the entire time.
3. Start with all nodes on, then every minute, turn 10 nodes off and at the halfway point, switch and every minute, turn 10 nodes on. The node blocks are selected based on their node identification numbers. For example, with 20 remote nodes, node numbers 11–20 would go off first, then 1–10, leaving all remote nodes off. Then node numbers 1–10 would switch on, followed by 11–20.
4. Every minute, a random set of nodes is selected to be on, with the remainder left off. For comparison purposes, the same set of random node sequences was used to test all simulators.

The first two scenarios baselines best and worst case energy consumption, respectively. The third scenario simulates a situation where a building has 10 sensors in each room and they are used in this manner. The last, random scenario simulates a chaotic situation without any predictability in the nodes' on/off states.

The Castalia power original numbers used were Rx = 62 mW, Tx = 57.42 mW, sleep = 1.4 mW, and baselinePowerConsumption = 6 mW. Converting to 3.0 V, the modified parameters were Rx = 59.1 mW, Tx = 52.2 mW, sleep = 0 (because TOSSIM uses radio off state = 0 and, when possible, we wanted to use the same settings) and baselinePowerConsumption = 5.4 mW.

In Fig. 3, without modification and with all nodes on for the entire scenario, TOSSIM and Castalia had similar remaining energies. The ns-3 simulator had significantly different remaining energy because it uses a different radio behavior. TOSSIM and Castalia started up and switched to receive (listen) unless transmitting, and then they only slept if ordered to do so, which is correct according to the cc2420 datasheet. Ns-3 did not implement a sleep state; its radio started in an idle state, transmitted/received when necessary, and otherwise was in an idle state. To correct this, in the modified parameters run, ns-3 had the idle current set to 0.0197 A when in the on state and 0.000020 A when in the off state, essentially implementing the same radio-state behavior.

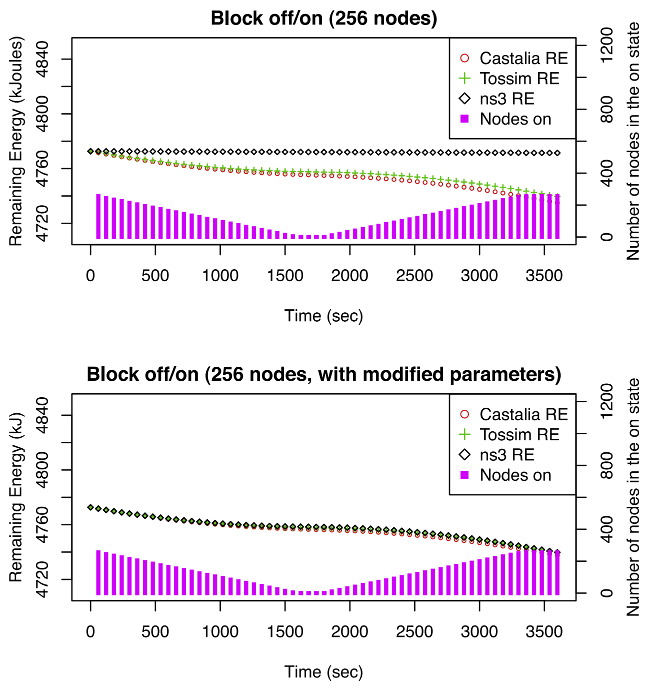

Fig. 5. Block off/on scenario with 256 nodes.

The modified parameters for ns-3 also included adding a SimpleDeviceEnergyModel set to 0.0018 A current draw in order to model baseline CPU and sensor power consumption since ns-3 only models radio power by default. 18 mA was picked to match what Castalia used for its baseline power consumption, a number which is just approximate based on the current draw from an average CPU and other electronic circuits.

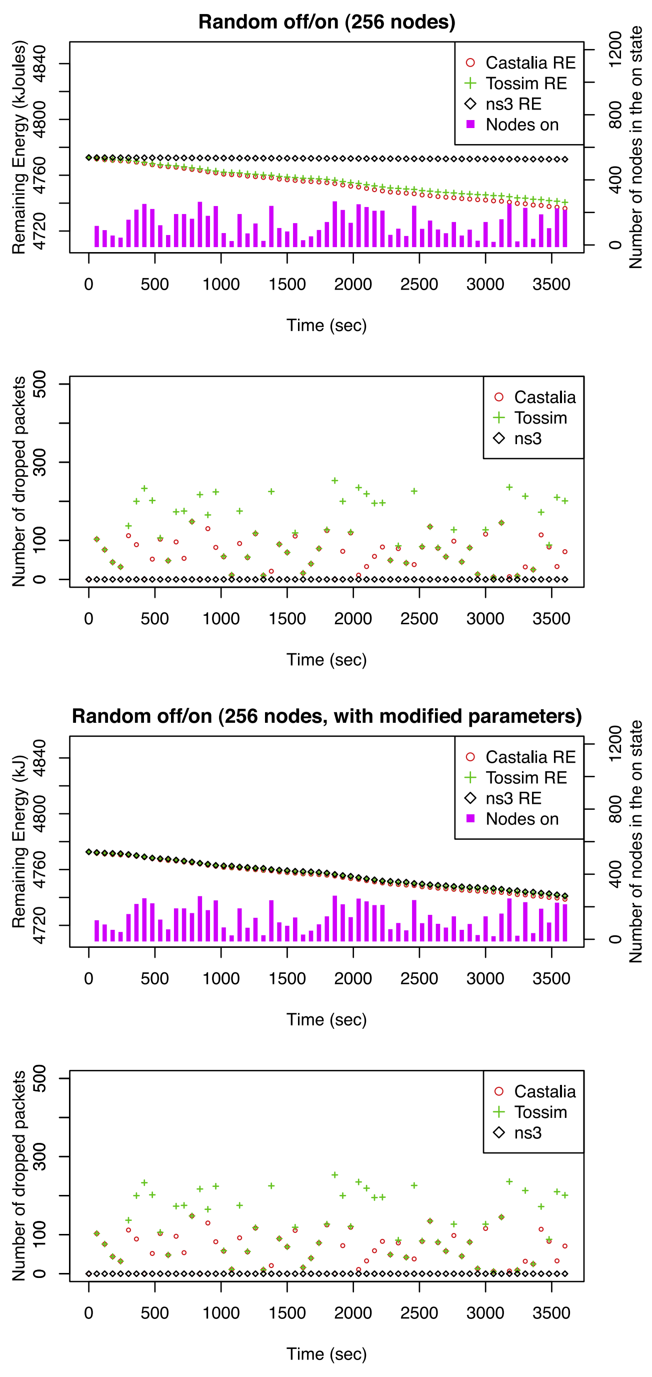

Fig. 6. Random off/on scenario with 256 nodes.

Without modification and with all nodes off, the above-described issue with the ns-3 radio idle state is not a factor. The three simulators still had somewhat different remaining energies due to the differences in how they calculate remaining energy (see Fig. 4).

Without modification and in the block off/on scenario, the ns-3 radio idle state issue again came into play (see Fig. 5). TOSSIM and Castalia report similar remaining energies, while that of ns-3 was markedly higher.

Without modification and in the random off/on scenario (see Fig. 6), once again the ns-3 radio idle state issue resulted in higher remaining energy. Also note that there were no dropped packets in the run of the ns-3 simulation. This is because all nodes participate in routing whether or not the node is on or off since ns-3 does not implement radio sleep state.

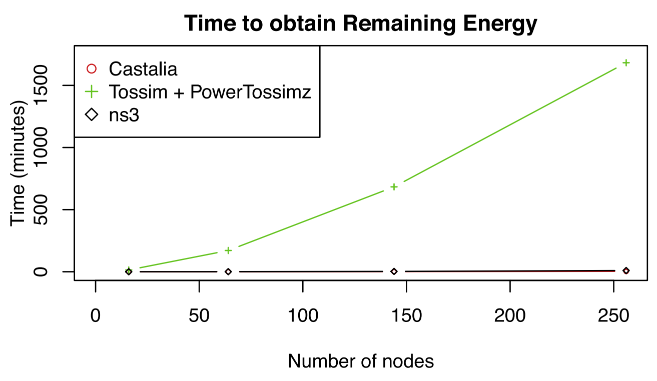

Fig. 7. Run time comparison.

As the number of nodes increased, the time to obtain the remaining energy from TOSSIM + PowerTossimz increased at a greater than linear rate (see Fig. 7). With 256 nodes and 3600 simulated seconds, it took 1682.4 min for TOSSIM + PowerTossimz (18 min for TOSSIM and 1664.4 min for PowerTossimz), 2.85 min for Castalia, and 9.77 min for ns-3.

7. Lessons learned

Packet collision: In Castalia, a simple interference/collision model was used in which no collisions happen, by assigning the collisionModel setting to zero. In TOSSIM, it was not possible to avoid collisions, but the problems caused by using a collision model in Castalia for the purposes of comparison outweighed any possible benefit. For example, one problem in Castalia was when collisions were not avoided, the setup phase of the MultipathRingsRouting functions incorrectly for a grid layout. Some nodes got the wrong ring number, and some nodes got no ring number at all.

TOSSIM timer: When using the TOSSIM timer component, the milliseconds passed into the different timer functions are not normal milliseconds (i.e., 1000 ms equals 1 s). By definition, time in TinyOS is measured in binary milliseconds: 1024 binary milliseconds per second. This nonstandard approach, which was not expressed in the timer API, was quite unexpected and only discovered after running simulations and investigating the odd timer behavior. This issue will quickly lead to problems if accurate timers are needed in your code.

PowerTossimz run time: There are at least two reasons for the long run times, both related to the Python language; it is interpreted code, which leads to longer execution times, and the Python scripts are not able to run faster when multiple cores are available.

Ns-3 radio sleep: The ns-3 simulator does not provide an out-of-the-box way to put the radio into or out of sleep from the node application code. To assist researchers experimenting with various energy saving schemes, this feature should have been built in from the first version of the simulator.

8. Conclusion

With several fundamental differences in implementation, comparing the WSN simulators was a challenging task. However, from the process of testing, many interesting lessons were learned that could not have been foreseen from the documentation accompanying these simulators alone.

Based on the selected criteria, and on extensive first-hand use of the simulators, ns-3 is clearly the best simulator to model power consumption while allowing dynamic reconfiguration of network parameters based on feedback from an end application. It has a large number of actively maintained models with which to work and is relatively easy to use.

In future work, we plan to find sustainable ways to reconfigure the WSN network to maximize end-application performance, while simultaneously being energy aware and seeking to sustain the network for as long as possible. In addition, it would be interesting to extend this comparison study to include results from a real sensor network testbed to determine which simulator performed closest to reality.

References

[1] A.K. Mohapatra, N. Gautam, R.L. Gibson, Combined routing and node replacement in energy-efficient underwater sensor networks for seismic monitoring, IEEE J. Ocean. Eng. 38 (1) (2013) 80–90.
[2] R. Huang, W.-Z. Song, M. Xu, N. Peterson, B. Shirazi, R. LaHusen, Real-world sensor network for long-term volcano monitoring: design and findings, IEEE Trans. Parallel Distrib. Syst. 23 (2) (2012) 321–329.
[3] I. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, A survey on sensor networks, IEEE Commun. Mag. 40 (8) (2002) 102–114.
[4] M. Cardei, M.T. Thai, Y. Li, W. Wu, Energy-efficient target coverage in wireless sensor networks, in: 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2005), vol. 3, 2005, pp. 1976–1984.
[5] C.-Y. Chong, S.P. Kumar, Sensor networks: evolution, opportunities, and challenges, Proc. IEEE 91 (8) (2003) 1247–1256.
[6] A. Chamam, S. Pierre, Energy-efficient state scheduling for maximizing sensor network lifetime under coverage constraint, in: Third IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMOB 2007), 2007, p. 63.
[7] M. Imran, A. Said, H. Hasbullah, A survey of simulators, emulators and testbeds for wireless sensor networks, in: Proceedings of the 2010 International Symposium in Information Technology (ITSim), vol. 2, 2010, pp. 897–902.
[8] M. Jevtić, N. Zogović, G. Dimić, Evaluation of wireless sensor network simulators, in: Proceedings of the 17th Telecommunications Forum TELFOR, 2009, pp. 1303–1306.
[9] M. Khan, B. Askwith, F. Bouhafs, M. Asim, Limitations of simulation tools for large-scale wireless sensor networks, in: IEEE Workshops of International Conference on Advanced Information Networking and Applications (WAINA), 2011, pp. 820–825.
[10] M. Korkalainen, M. Sallinen, N. Karkkainen, P. Tukeva, Survey of wireless sensor networks simulation tools for demanding applications, in: Fifth International Conference on Networking and Services (ICNS '09), 2009, pp. 102–106.
[11] B. Musznicki, P. Zwierzykowski, Survey of simulators for wireless sensor networks, Int. J. Grid Distrib. Comput. 5 (3) (2012) 23–50.
[12] C. Singh, O. Vyas, M. Tiwari, A survey of simulation in sensor networks, in: International Conference on Computational Intelligence for Modelling Control Automation, 2008, pp. 867–872.
[13] H. Sundani, H. Li, V.K. Devabhaktuni, M. Alam, P. Bhattacharya, Wireless sensor network simulators a survey and comparisons, Int. J. Comput. Netw. 2 (5) (2011) 249–265.
[14] L.M. Feeney, Towards trustworthy simulation of wireless MAC/PHY layers: a comparison framework, in: Proceedings of the 15th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems, ACM, New York, NY, USA, 2012, pp. 295–304.
[15] E. Weingartner, H. vom Lehn, K. Wehrle, A performance comparison of recent network simulators, in: IEEE International Conference on Communications (ICC '09), 2009, pp. 1–5.
[16] L. Bergamini, C. Crociani, A. Vitaletti, M. Nati, Validation of WSN simulators through a comparison with a real testbed, in: Proceedings of the 7th ACM Workshop on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks, ACM, New York, NY, USA, 2010, pp. 103–104.
[17] A. Stetsko, M. Stehlik, V. Matyas, Calibrating and comparing simulators for wireless sensor networks, in: IEEE 8th International Conference on Mobile Adhoc and Sensor Systems (MASS), 2011, pp. 733–738.
[18] H. Tyan, Design, realization and evaluation of a component-based compositional software architecture for network simulation (Ph.D. thesis), The Ohio State University, 2002.
[19] G. Chen, B. Szymanski, Cost: a component-oriented discrete event simulator, in: Proceedings of the 2002 Winter Simulation Conference, vol. 1, 2002, pp. 776–782.
[20] G. Chen, J. Branch, M. Pflug, L. Zhu, B. Szymanski, Sense: a wireless sensor network simulator, in: Advances in Pervasive Computing and Networking, Springer US, 2005, pp. 249–267.
[21] Opnet, http://www.riverbed.com/products/steelcentral/opnet.html.
[22] H.N. Pham, D. Pediaditakis, A. Boulis, From simulation to real deployments in WSN and back, in: IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM 2007), 2007, pp. 1–6.
[23] Castalia, http://castalia.research.nicta.com.au.
[24] Omnet++, http://www.omnetpp.org.
[25] P. Levis, N. Lee, M. Welsh, D. Culler, TOSSIM: accurate and scalable simulation of entire TinyOS applications, in: Proceedings of the 1st International Conference on Embedded Networked Sensor Systems, SenSys '03, ACM, New York, NY, USA, 2003, pp. 126–137.
[26] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, D. Culler, The NesC language: a holistic approach to networked embedded systems, in: Proceedings of the ACM SIGPLAN 2003 Conference on Programming Language Design and Implementation, ACM, New York, NY, USA, 2003, pp. 1–11.
[27] E. Perla, A.O. Catháin, R.S. Carbajo, M. Huggard, C. Mc Goldrick, PowerTOSSIMz: realistic energy modelling for wireless sensor network environments, in: Proceedings of the 3rd ACM Workshop on Performance Monitoring and Measurement of Heterogeneous Wireless and Wired Networks, ACM, New York, NY, USA, 2008, pp. 35–42.
[28] ns-3, https://www.nsnam.org.

Joel Helkey is a PhD student in computer science at Washington State University. He received his BS in Software Engineering from the Oregon Institute of Technology and his MS in Computer Science from Washington State University. His research interests include Artificial Intelligence, Machine Learning, and Wireless Sensor Networks: simulation/testbeds, network management, and performance optimization. He is a member of the IEEE and the IEEE Computer Society.

Lawrence Holder is a professor in the School of Electrical Engineering and Computer Science at Washington State University. His research interests include artificial intelligence, machine learning, data mining, and smart environments. He has over 150 publications in these areas, and has led several related research projects funded by DOD, DHS, NASA and NSF. Dr. Holder is director of the WSU Smart Environments Research Center and co-director of the WSU Artificial Intelligence Laboratory. He is an associate editor for the journals IEEE Transactions on Knowledge and Data Engineering, Computing, Intelligent Data Analysis, and Social Network Mining. Dr. Holder received his Ph.D. in Computer Science from the University of Illinois at Urbana-Champaign in 1991. He is a senior member of ACM and IEEE.

Behrooz A. Shirazi is currently the Huie-Rogers chair professor and director of the School of Electrical Engineering and Computer Science. His research interests include the areas of pervasive computing, software tools, distributed real-time systems, scheduling and load balancing, and parallel and distributed systems. He has received grant support totaling more than $8 million from federal agencies, including the US National Science Foundation (NSF), the US Defense Advanced Research Projects Agency (DARPA), and the AFOSR, and private sources, including Texas Instruments and Mercury Computer Systems. He has received numerous teaching and research awards. He is currently the editor-in-chief for the Special Issues for Pervasive and Mobile Computing Journal and has served on the editorial boards of the IEEE Transactions on Computers and the Journal of Parallel and Distributed Computing. He is a senior member of the IEEE.