Performance and Scalability Evaluation of the Castalia Wireless Sensor Network Simulator

Dimosthenis Pediaditakis
Imperial College London
180 Queen's Gate
London, SW7 2AZ, United Kingdom
dpediadi@doc.ic.ac.uk

Yuri Tselishchev
NICTA, Univ. of Sydney
13 Garden Street
Eveleigh, NSW, Australia
Yuri.Tselishchev@nicta.com.au

Athanassios Boulis
NICTA
13 Garden Street
Eveleigh, NSW, Australia
Athanassios.Boulis@nicta.com.au

ABSTRACT

Castalia is an open-source simulator for wireless sensor networks and body area networks which is widely used in the academic and research community. This paper presents a basic evaluation study of Castalia, reporting computation time and memory usage for a variety of scenarios/benchmarks. Moreover, key parameters, such as network size, simulation time, fraction of mobile nodes are varied to reveal Castalia's scalability potential. We discuss our results and explain counterintuitive findings in simulator's performance. The results and their explanation can be used by Castalia users as a guide to determine the limits they can push their simulations, as well as to make parameter choices that trade-off accuracy for performance. They also provide an indication of Castalia's performance capabilities to potential users.

Categories and Subject Descriptors

I.6 [Computing Methodologies]: Simulation and Modeling

General Terms

Wireless Sensor Network simulator, scalability performance evaluation.

Keywords

Castalia, WSN, wireless, sensor, network, simulator, performance, scalability.

1. INTRODUCTION

The wireless sensor networks (WSN) research community has relied heavily on simulation to propose and validate solutions at every stage and layer of a WSN system. The inherent ad hoc nature of WSN, their embedded and distributed programming difficulties, and their large-scale proposed deployments in some cases, have made real deployments and testing prohibitory. Researchers have used various platforms for their simulating needs, many of which are evolutions of wired network simulators, or they are modeling a small portion of the whole system. Castalia [1] was created out of the need to have a simulator specifically designed for WSN, encompassing all important aspects of the system and providing the most accurate modeling available in the research community, starting with the communication models (i.e., wireless channel and radio models) [14][15]. Due to its accurate modeling and relative ease of use, Castalia has gained wide acceptance in the WSN research community with a number of citations in the literature. Despite its acceptance though, there has not been a rigorous evaluation study on the performance capabilities and scalability of Castalia. This study can be of particular interest since the expert's "common sense" dictates that there is a significant performance cost for accurate modeling, and the effect is even more pronounced when we are trying to scale our simulations with respect to network size and other parameters of interest. Although, this claim is true in a general sense, it is worth investigating -quantitatively- the performance of the simulator to reveal specific and practical boundaries of its usage.

This is indeed the purpose of the paper. We have conducted a series of simulations under different scenarios and measured the time needed to complete these simulations as well as their memory usage. One scenario includes a complex real-life application: a WSN to monitor the structural health of bridges. An advanced MAC protocol, a routing protocol, and a physical process are used in this scenario. It is interesting to see how the performance scales with network size under different channel assumptions and under different MAC protocols. Other scenarios include a more artificial setting, specifically designed to evaluate Castalia. Nodes placed on a grid (some of them mobile) transmitting periodically packets, using a CSMA MAC protocol. We observe and report how performance scales with different settings in network size, fraction of nodes which are mobile, simulation time duration, while different channel models are used (ideal, static realistic and time-varying realistic).

It is important to note that we are not trying to compare the performance of Castalia with other WSN simulators, although Castalia would compare favorably to other platforms for *seemingly* similar tasks. We are avoiding this kind of comparison because two simulator platforms do not offer the same functionality nor do they have the same goals in their modeling, so it is not possible to compare "apples to apples". This key idea has also been conveyed by Voigh et al. [21]. We are rather showing the performance capabilities and limits of Castalia and we are exposing performance-accuracy tradeoffs in a quantitative way. Researchers can use this information together with the

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2010.8727
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2010.8727

functionality specification of Castalia to decide when and how to use the simulator.

The rest of the paper is organized as follows: Section 2 discusses related work. Section 3 provides a brief description of some of the models used in Castalia, in particular relating to communication. Section 4 presents our performance results and explains some counter-intuitive findings. Finally section 5 concludes the paper.

2. RELATED WORK

There are several simulators used in WSN research as mentioned in the introduction. The most popular ones are: 1) the network-simulator-2 (ns-2) [18], and its specific WSN modifications [3][13] 2) TOSSIM [8], a discrete event simulator, emulating nodes that run the TinyOS [9] operating system, 3) J-Sim (formerly known as JavaSim) [17], a general purpose JavaTM-based simulator developed after NS-2 with the aim to overcome the scalability issues of the latter. Comparing these and other WSN simulators with Castalia in terms of functionality and model accuracy is outside the scope if this work. The reader can refer to [6][16] for surveys that compare these platforms as well as several others.

Work that evaluates simulators is of more relevance to this paper. Nicol et al. in [12] compared the scalability of NS-2, J-Sim and SSFNet by measuring the execution-time, the number of generated events, the memory footprint, and finally the accuracy of results. Ns-2 was found to be the fastest among the other two, but also the most memory-hungry and JavaSim was the worst performer in terms of speed. Lopez et al. in [10] present several design practices for designing fast and scalable simulators for sensor networks, but unfortunately, they were limited in theoretical implementation norms, without actually performing some kind of benchmarking. Finally, in their recent work, Weingartner et al. [22], compared the performance of ns-2, ns-3, OMNeT++, Jist and SimPy. The comparison criteria they used were execution time and memory usage for various network sizes. They also made an interesting measurement of simulation execution time versus the packet drop probability in the network, which makes a lot of sense because as more and more packets are removed from the simulation, the fewer events are processed.

We are using the same metrics as the aforementioned works but we do not compare the results with other WSN simulators. As explained in the introduction this would involve comparing dissimilar things. Focusing solely on Castalia allows us to perform a well-rounded set of simulations ranging different scenarios and different models while varying different "scaling" parameters. This is the first such evaluation survey on Castalia.

3. SUMMARY OF CASTALIA'S MODELS

In this section we present a brief summary of the models that are relevant to the simulation results we present in the paper. For a more detailed description please refer to Castalia's User Manual [2]. Castalia is built on OMNeT++ [20], a framework which provides the basic machinery and tools to write discrete-event simulators, but by itself does not provide any components specifically for computer network simulations.

3.1 Channel Modeling

In Castalia, nodes are OMNeT++ modules that do not connect to each other directly but through the wireless channel module. The arrows in Figure 1 signify message passing from one module to another. When a node has a packet to send, it goes through the wireless channel which then decides which nodes should receive the packet.

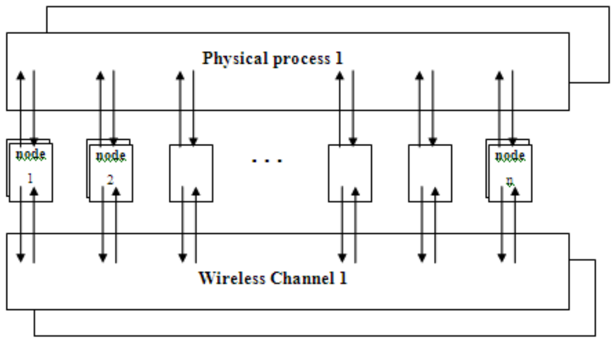

Figure 1: The top level module connection in Castalia

One important aspect of the wireless channel modeling is to estimate the average path loss between two nodes, or in general, two points in space. For WSN, where the separation of nodes is from a couple of meters to a hundred meters, the lognormal shadowing model has been shown by Zuniga [23] to give accurate estimates for average path loss *when* appropriate radio models are considered *too*. Castalia uses the lognormal shadowing model, along with radios that return a packet reception probability (PRP) given the received signal to noise ratio (SNR). These offer the realistic conditions that Zuniga describes. When we want to emulate an ideal channel model, we eliminate the shadowing (by eliminating the sigma parameter) and adopt an ideal radio modulation scheme (where the modulation function SNR→PRP is a 0-1 step function). These changes effectively make the communication model to be the unit disk model. Interference is computed dynamically and directly affects the SNR, instead of having it as a separate feature. In the ideal model all interference is discarded.

The realistic model handles average path loss, but there is another aspect to the channel: Time variability. This is particularly important for Body Area Networks, where experimental data shows a large variation in the received signal with time. The solution we gave in Castalia is generic, accurate, yet efficient.

When we have to calculate the path loss at a specific moment, we first find the average path loss from the state we have stored during channel initialization and then we have to find the component of the path loss due to temporal variation. To do this we have kept the last "observed" value (in simulation this means the last value we computed) and the time it has passed since then. These two numbers should define a probability density function (pdf) that we draw our new value from. Obviously, if little time has passed, then the bulk of the probability in this pdf should be in values close to the last observation. If the last observation is a deep fade (e.g. -40 dB) then the pdf should "boost" bigger values. We cannot produce those pdfs dynamically from a model. They have to be produced from our experimental measurements and entered as model parameters. This in turn implies that we cannot have a pdf for any combination of last observation (dB value) and time passed. Instead we have to declare for what dB values and what times we are providing the pdfs. Our algorithm then uses a series of random number draws from a combination of available pdfs to approximate the results for any dB value and time passed. We also provide a way to describe a pdf in a manner that is computationally cheap to draw a random number from it, yet it produces accurate results. We will not delve further into the effectiveness of our time-varying channel modeling, but its macroscopic performance results will be shown in the performance evaluation section.

3.2 Mobility affecting Channel Modeling

Allowing for mobile nodes complicates matters, since now it is not enough to take the average path losses between the nodes. We need to keep state about path losses between points in the space. This implies that we need to break up the space in discrete cells and calculate the path losses from each cell to each other cell. Look at figure 2 for an example of such a map only for one transmitting cell.

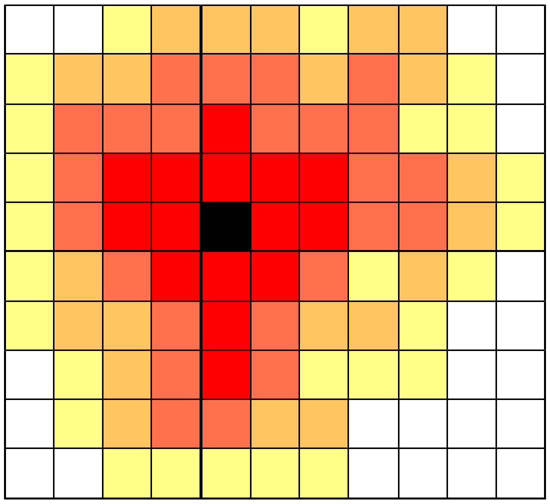

Figure 2: Path loss map in a 2D space segmented in cells (for a single transmitter cell)

The map is calculated with the lognormal shadowing model we described before. Then, we just take cell locations and cell IDs instead of specific node locations and node IDs. The smaller the cell size is, the more fine-grained and accurate is your path loss map but also the more memory is needed to store it. Imagine you just have a 2-dimentional field 100m by 100m and you would like to have a cell of 1m by 1m, this results in 10,000 cells. This in turn results in a total of $10{,}000^2 = 100{,}000{,}000$ distinct cell-to-cell path losses. With several bytes per path loss element (we store information besides the path loss) this can grow up to the order of GBytes of memory. Our algorithm is smart and does not keep all possible combinations but still the same scaling factor applies ($O(N^4)$). Consequently the state-space can quickly explode, as well as the processing time searching through these cells. Adding a 3rd dimension can aggravate matters, so the user has to be aware of these parameter values.

If we do not have any mobility in our scenario then we can declare this by setting a special parameter to be true and save a lot of memory space and computation time. In this case, the space is not broken up into cells and the exact node locations are used in path loss computations.

4. PERFORMANCE EVALUATION

A Castalia simulation scenario is fully described using an OMNeT++ specific parameter description file. For the purposes of our experiments, we had to perform a big number of simulations, and thus, we made extensive use of scripting to automatically generate the parameter description files, measure execution time, monitor the memory usage and collect/plot results. Execution times have been measured at a micro-second scale, using the Time::HiRes library of Perl. Memory profiling has been implemented with a Python script which runs at the background and polls the process's resident set size (rss) at main memory, at a customizable sample rate (set to 10Hz). The machine we ran our experiments was equipped with an Intel Core 2 DuoTM E6750 CPU running at 2.66GHz clock speed (1333MHz bus speed) and with 4GB of RAM (DDR2 @ 800MHz). Manufacturer's specifications for this particular CPU amount to 41673 MPTOPS and 21.28 GFLOPS. Moreover, an identical machine's SPEC CPU2006 benchmark results are publicly available [4][5] and the reported peak/base scores are 20.5/18.3 for the CINT2006 test, and 17.7/17.1 for the CFP2006 test.

4.1 Real-life scenarios

The first set of simulations is about an application we have developed for a real life need: Designing a WSN for structural health bridge monitoring. The nodes are deployed statically along a bridge of variable length. The network is 3 nodes wide and N nodes long, where N depends on the length of the bridge. The nodes are 20m apart as they are following the length of the bridge. A physical process creating triggering cars is part of the scenario. When nodes detect a car, they send the data to the sink sampling at 5Hz. There is one node in the middle which acts as the sink Simple tree routing is employed to achieve multi-hop communication. As for the MAC, we are using either a simple CSMA or the more complex T-MAC [19].

Figure 3 shows the ratio of real execution time over simulated time as a function of the network size (number of nodes). There are 3 curves for the 3 different scenarios we chose: A simple CSMA based MAC with 1) realistic and 2) ideal channel modeling and 3) T-MAC with a realistic channel modeling. All simulations run for a simulated time of 1800sec (30min). It must be noted that results showed very little variation against different random seeds, thus confidence levels are not shown in the figures (since they are almost 0).

We notice that the realistic CSMA scenario is the fastest of all, followed by the realistic T-MAC (at least for network sizes >150). This might seem counter-intuitive as the realistic model is expected to be more computationally intensive. However, in Castalia, the ideal model is not implemented by a computationally simple module. We are using the same code as the realistic model. The parameter change to achieve/emulate the ideal model does not have a considerable effect on the code executed in the wireless channel. Then why is the simulation much slower when using the ideal model? The answer is that because the channel characteristics change, we have more packets being successfully received and forwarded, thus more events altogether. Similar findings have been reported also by Weingartner et al. in [22]. Looking at the final received packet success rate, the ideal model reaches 99% when the realistic model reaches 59% for the biggest network. We also notice that the use of T-MAC instead of CSMA has an important impact on execution time, at least for small to medium network sizes, when most packets are delivered. For

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2010.8727
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2010.8727

network sizes smaller than 150 nodes the T-MAC scenario is almost twice as slow as its CSMA counterpart and also the slowest of the all three scenarios. This reflects the complexity of the T-MAC protocol compared to CSMA.

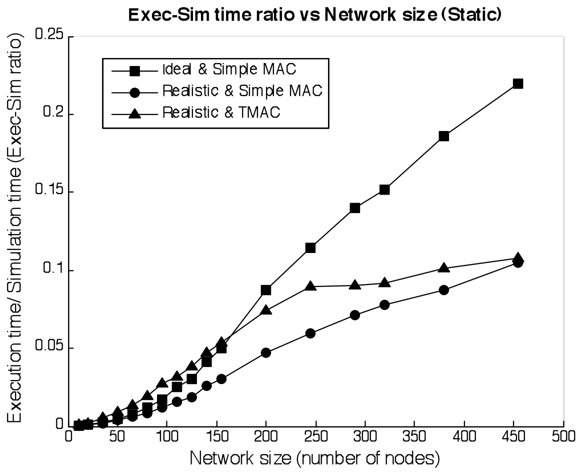

Figure 3: The Bridge Test scenarios

The memory usage follows an almost perfect linear pattern so we do not include the graph due to space restrictions. For all three different scenarios the memory increases from 4MB (for the 20 node case) to 20MB (for the 455 node case) linearly.

Another interesting trend is to see is the initialization overhead of Castalia. Figure 4 shows the “real execution” to “simulation time” ratio for different simulation times, for a 110-node network.

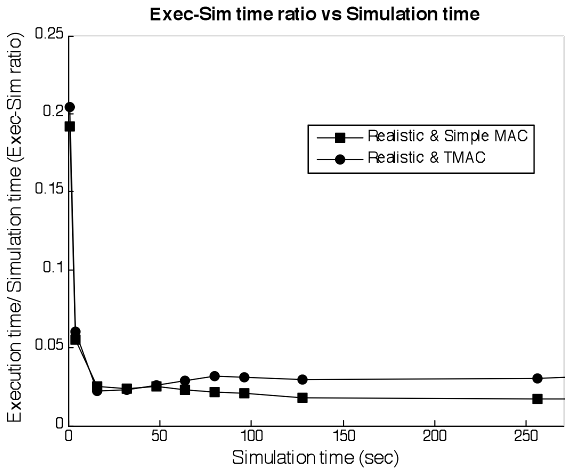

Figure 4: Showing Initialization overheads

We notice that for small simulation time values (e.g., 1sec or 5secs) CSMA and TMAC execution times seem the same, both being quite slow. This is due to the 250ms initialization delay that we need for this network.

4.2 Evaluation-specific scenarios

To better expose the scaling of performance with network size (not depending on routing protocols, or MAC, or physical process), and to delve into mobility scenarios we decided to run some “artificial” scenarios that were designed specifically for this evaluation.

The general setting is that the nodes are placed on a rectangle grid and are transmitting 1 packet/sec using CSMA. The packets are not forwarded when received and there is no routing whatsoever.

In this way we have a better view at the performance and scalability of the lower communication layers (wireless channel and radio) and less on the effect of the MAC, routing, application, or physical process. For example with the bridge scenario there was little point to go to 1000 node network since the routing of messages over so many hops would take over and furthermore we would never realistically design an application to work like this. Now with these evaluation-specific settings we can meaningfully explore these boundaries.

Figure 5 shows performance results for a static network and for three different channel models: ideal, realistic and realistic with time variability. The total simulation time duration was again 1800 sec.

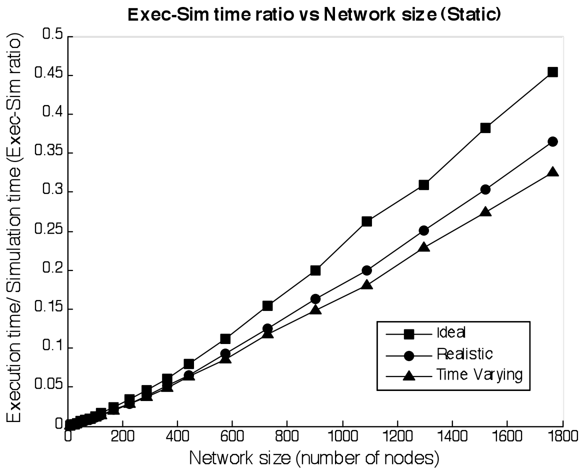

Figure 5: Static networks performance

We notice again the simulations becoming slower as networks become bigger. The increase is indeed almost linear (e.g., for the time-varying model: 441 nodes run in 115sec, 1784 nodes run in 585sec). We notice again that the supposedly simple ideal model is slower and the time-varying model is the fastest. Again, this is not an indication of the actual complexity of their computation but the indirect computation caused by extra messages generated. More specifically, the ideal model causes more receptions of transmitted packets than the realistic model and also from the time varying model (whose deep fades affect reception even worse than the time-invariant realistic model). Memory is again increasing linearly from 5MB to 64MB.

In this setting we also start exploring the effect of mobility. Figure 6 shows the performance when 0.3 of the nodes are mobile (each mobile node chooses a random point to move to from its starting grid position). We have used 5m×5m cells to break up the space.

We notice similar trends to the static network results. The absolute numbers however are quite different: networks with presence of mobile nodes are almost twice as slow as networks with only static nodes. This is because we are now breaking up the space into cells and doing channel computations with cells.

Memory usage increases significantly, reaching 635MB (time varying) and 510MB (ideal) for the biggest network.

We also carried another series of simulations where we varied the fraction of mobile nodes for a 484-node network. Figure 7 shows the results.

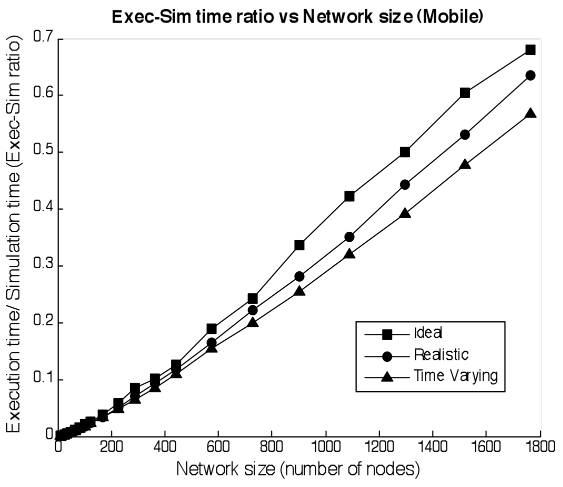

Figure 6: Mobile networks performance

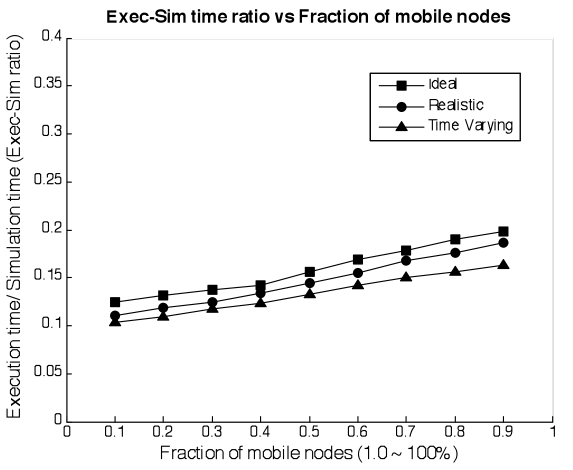

Figure 7: Performance vs the fraction of mobile nodes

It is clear that although the simulations slow down when we have more mobile nodes (as expected since we have more node movement events) the slowdown is not significant. Thus, we can claim that one important characteristic to faster simulations is to have all nodes static. From the moment we have at least 1 mobile node, the percentage of mobile nodes does not make much difference.

Finally let us look at the effect of the cell size in the performance and memory footprint. Figure 8 shows the results for varying the cell size for a 169 node network.

We notice that for small cell sizes the simulation execution time explodes. A good performance/accuracy trade-off point for our particular scenarios seems to be around 5m×5m cell sizes. Figure 9 shows the results for the memory footprint of the same simulations. The graph shows a memory footprint that exceeds 1.5GB of RAM if a cell-size of 2m x 2m is chosen. Again cell size 5m x 5m shows a reasonable compromise.

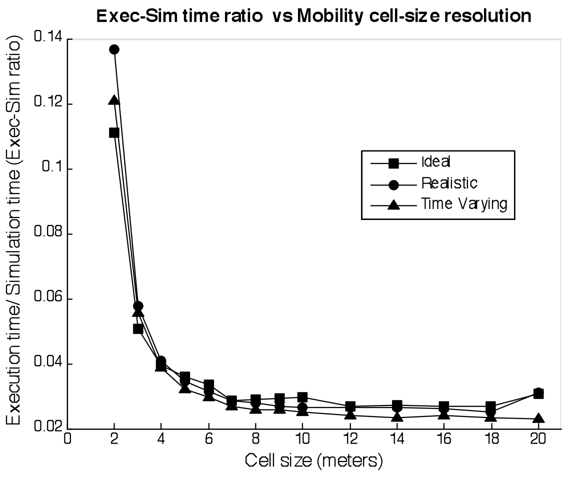

Figure 8: Performance vs cell size

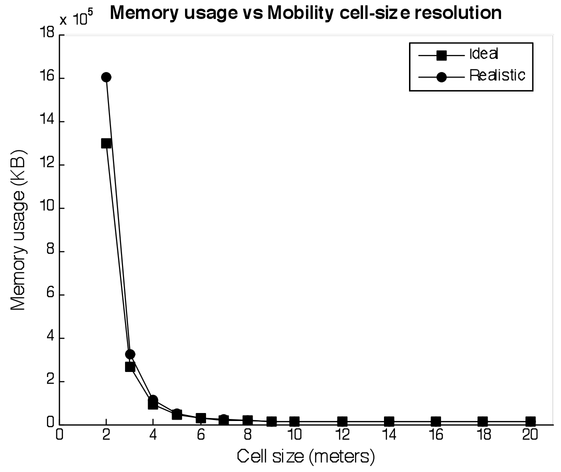

Figure 9: Memory vs cell size

5. CONCLUSION

The paper presents a performance evaluation of Castalia in terms of computation time and memory footprint for various simulation scenarios. In particular we see how Castalia's performance scales when different parameters are varied and also when different models are used. We notice that the realistic channel modeling even with a time-varying component is not prohibitory for large network sizes, while mobility can be handled efficiently if being careful with the cell size. The results of this work can act as a guide for Castalia users to tune their simulations and also to estimate their computation-time budget.

6. REFERENCES

[1] Castalia Simulator website http://castalia.npc.nicta.com.au/

[2] Castalia User's Manual http://castalia.npc.nicta.com.au/documentation.php

[3] Downard, I., Simulating Sensor Networks in NS-2. Technical Report NRL/FR/5522-04-10073, Naval Research Laboratory, Washington, D.C., U.S.A., May 2004.

[4] http://www.spec.org/cpu2006/results/res2007q3/cpu2006-20070723-01541.html (accessed Feb 1st 2010)

[5] http://www.spec.org/cpu2006/results/res2007q3/cpu2006-20070723-01543.html (accessed Feb 1st 2010)

[6] Korkalainen, M., Sallinen, M., Kärkkäinen, N., and Tukeva, P. 2009. Survey of Wireless Sensor Networks Simulation Tools for Demanding Applications. In Proceedings of the 2009 Fifth international Conference on Networking and Services - Volume 00 (April 20 - 25, 2009). ICNS. IEEE Computer Society, Washington, DC, 102-106.

[7] Lessmann, J., Janacik, P., Lachev, L., and Orfanus, D. 2008. Comparative Study of Wireless Network Simulators. In Proceedings of the Seventh international Conference on Networking (April 13 - 18, 2008). ICN. IEEE Computer Society, Washington, DC, 517-523.

[8] Levis, P., Lee, N., Welsh, M., and Culler, D. 2003. TOSSIM: accurate and scalable simulation of entire TinyOS applications. In Proceedings of the 1st international Conference on Embedded Networked Sensor Systems (Los Angeles, California, USA, November 05 - 07, 2003). SenSys '03. ACM, New York, NY, 126-137.

[9] Levis, P., Madden, S., Polastre, J., Szewczyk, R., Whitehouse, K., Woo, A., Gay, D., Hill, J., Welsh, M., Brewer, E. and Culler D., TinyOS: An operating system for wireless sensor networks. In Ambient Intelligence. Springer-Verlag, 2004.

[10] Lopez, E. E., Vales-Alonso, J., Martinez-Sala, A., Pavon-Mario, A., and Garcia-Haro, J., Simulation scalability issues in wireless sensor networks. Communications Magazine, IEEE, 44(7):64–73, 2006.

[11] Naoumov, V. and Gross, T. 2003. Simulation of large ad hoc networks. In Proceedings of the 6th ACM international Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems (San Diego, CA, USA, September 19 - 19, 2003). MSWIM '03. ACM, New York, NY, 50-57.

[12] Nicol, D. M., Comparison of network simulators revisited, May 2002.

[13] Park, S., Savvides, A., and Srivastava, M. B. 2000. SensorSim: a simulation framework for sensor networks. In Proceedings of the 3rd ACM international Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems (Boston, Massachusetts, United States, August 20 - 20, 2000). MSWIM '00. ACM, New York, NY, 104-111.

[14] Pediaditakis, D., Mohajerani, S. H., Boulis, A., "Poster Abstract: Castalia: the Difference of Accurate Simulation in WSN," In the 4th European conference on Wireless Sensor Networks, (EWSN 2007), Delft, The Netherlands, January, 2007.

[15] Pham, H. N., Pediaditakis, D., Boulis A., "From Simulation to Real Deployments in WSN and Back," A World of Wireless, Mobile and Multimedia Networks, International Symposium on, pp. 1-6, 2007 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, 2007

[16] Singh, C. P., Vyas, O. P., and Tiwari, M. K. 2008. A Survey of Simulation in Sensor Networks. In Proceedings of the 2008 international Conference on Computational intelligence For Modelling Control & Automation (December 10 - 12, 2008). CIMCA. IEEE Computer Society, Washington, DC, 867-872.

[17] Sobeih, A., Chen, W. P., Hou, J. C., Kung, Lu. C., Li, N., Lim, H., Tyan, H. Y. and Zhang, H., J-sim: A simulation environment for wireless sensor networks. In Annual Simulation Symposium, pages 175–187, 2005.

[18] The Network Simulator ns-2 (http://www.isi.edu/nsnam/ns).

[19] van Dam, T. and Langendoen, K. 2003. An adaptive energy-efficient MAC protocol for wireless sensor networks. In Proceedings of the 1st international Conference on Embedded Networked Sensor Systems (Los Angeles, California, USA, November 05 - 07, 2003). SenSys '03. ACM, New York, NY, 171-180.

[20] Varga, A., "The OMNeT++ Discrete Event Simulation System", Proceedings of the European Simulation Multiconference (ESM'2001).

[21] Voigt, T., Eriksson, J., Österlind, F., Sauter, R., Nils A., Marron, P., Reynolds, V., Shu, L., Visser, O., Koubaa, A. and Köpke, A., Towards comparable simulations of cooperating objects and wireless sensor networks. In: 1st International Workshop on Performance Methodologies and Tools for Wireless Sensor Networks, 23 Oct 2009, Pisa, Italy.

[22] Weingärtner, E., vom Lehn, H. and Wehrle, K., (2009), "A performance comparison of recent network simulators", ICC 2009: IEEE International Conference on Communications.

[23] Zuniga, M. and Krishnamachari, B. 2004. Analyzing the transitional region in low power wireless links. In Proceedings of the IEEE 1st Annual Conference on Sensor and Ad Hoc Communications and Networks (SECON). 517--526.