A Hybrid Framework for Soft Real-Time WSN Simulation

A. Lalomia, G. Lo Re, M. Ortolani

In Proceedings of the 13th IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications, 2009. DS-RT '09, pp. 201-207

A Hybrid Framework for Soft Real-Time WSN Simulation

Antonio Lalomia, Giuseppe Lo Re, Marco Ortolani
DINFO - Department of Computer Engineering
Università degli Studi di Palermo
Viale delle Scienze, ed 6. - 90128 Palermo, Italy
Email: {lalomia, lore, ortolani}@unipa.it

***Abstract*—The design of a wireless sensor network is a challenging task due to its intrinsically application-specific nature. Although a typical choice for testing such kind of networks requires devising ad-hoc testbeds, this is often impractical as it depends on expensive, and hard to maintain deployment of nodes. On the other hand, simulation is a valuable option, as long as the actual functioning conditions are reliably modeled, and carefully replicated.**

The present work describes a framework for supporting the user in early design and testing of a wireless sensor network with an augmented version of TOSSIM, the de-facto standard for simulators, that allows merging actual and virtual nodes seamlessly interacting with each other; the proposed tool does not require any special modification to the original simulation code, but it allows contemporary execution of code in actual, and virtual nodes, as well as simulation of nodes executing different application logics. The reported experimental results will also show how soft-real time constraints are guaranteed for the augmented simulation.

***Keywords*-simulation; wireless sensor networks**

I. Introduction

The design of a Wireless Sensor Network (WSN) is a very application-specific task, especially due to the peculiarity of the considered deployment environment; for instance, besides traditional monitoring tasks, recent literature also reports more advanced experimentations on such hostile environments as underground, or underwater [1]–[3]. In such cases, nodes need to be equipped with specialized sensors, thus increasing their individual cost; moreover, reliable predictive models for data correlation, or radio propagation are seldom available, so that algorithms for such functionalities as packet routing, data gathering and in-network data processing cannot be effectively designed and tuned. A thorough preliminary test phase is thus necessary, either by means of specifically crafted testbeds, or via reliable simulations.

Although a few "general-purpose" testbeds have been proposed in the past, such as MoteLab [4], WSN applications must be tested on a large scale, and under complex and varying conditions in order to capture a sufficiently wide range of interactions, both among nodes, and with the environment. However, the deployment of a large number of nodes in hostile environments could become prohibitively expensive and practically unfeasible because of maintenance costs. Simulations can address those issues by providing controlled, reproducible environments, and specialized software tools for monitoring and debugging, so that the actual deployment of nodes may be delayed till after algorithms have been thoroughly tested; however they may not deliver fully reliable results, especially due to over-simplistic assumptions about the physical channels, and the node radio models. In order to avoid these shortcomings, the authors of [5] recommend that simulation and model designers run the same code both in simulated and in real systems, so as to allow for easier comparison about experimental and simulation results; furthermore, it is advisable that the simulation framework provide a range of models for the physical and data-link layer, to be tested with a wide range of parameters, and finally that real-world characteristics are taken into account. Recent approaches toward the design of more realistic simulation environments have suggested the use of hybrid tools enabling the interaction of virtual nodes with actual ones; on one hand, the use of simulated nodes allows for the generation of easily scalable scenarios, while nodes deployed in the real sensor field are used as a complement, in order to generate realistic data models and to steer the behavior of their virtual counterpart.

Our preliminary experiments on this topic [6] showed us the feasibility of such approach, and encouraged us to pursue further research. In this paper we describe the design of a framework for hybrid simulation of wireless sensor networks that exploits the models constructed from data sensed by a minimal deployment of actual nodes in order to integrate and validate the simulation; in particular, we address the simulation of nodes of the commonly available *mote* family, and we extend the functionalities of TOSSIM [7], their *de-facto* standard simulator. Our framework does not require any modification to the simulation code, and allows for contemporary execution of code both in actual and virtual nodes, as well as for simulation of nodes executing different application logics.

One of the key challenges when trying to deliver such kind of realistic simulations is how to ensure *fidelity*, in terms of ability to reproduce the same behavior both in virtual and real nodes, and *accuracy* of timing. Unlike "pure" simulators, where simulation time does not necessarily need to be strictly correlated to actual time as long

as correct sequencing is maintained, the hybrid approach requires coordination among virtual and real nodes. The reported experimental results will show that our framework achieves acceptable performances in terms of *soft real-time* constraints, by providing probabilistic end-to-end delay guarantees.

The remainder of the paper is organized as follows. Section II summarizes the main issues to be addressed when designing realistic WSN simulators, and outlines other relevant works; Section III describes our approach, while Section IV presents an assessment scenario and the related experimental results; finally, Section V reports our conclusions, and describes our on-going work.

II. Issues in WSN Simulation

Traditional techniques for performance assessment of wired and wireless networks usually rely on analytical methods, but in the case of WSNs a growing interest has been directed toward simulation tools, due to the peculiar nature of such networks, including the well-known constraints on available energy resources on nodes, and their tight interaction with the surrounding environment, as well as the fact that many research problems in this field are still open. Scalability, extensibility, ease of simulation design, and reliability are desirable features for quick prototyping, and assessment, before actual deployment; moreover, in order to reduce the complexity of modeling the environment and its influence on the system, and to minimize the difficulties in porting simulated sensor networks to actually deployed systems, it may be advisable to use "real code" simulation tools, that run identical code in simulation and deployment.

Many different approaches to WSN simulation have already been presented in literature, starting with extensions to generic network simulators, such as SensorSim [8], an event-driven simulator built on top of ns-2, that introduces the concept of *sensor function model* and of a *power model* for a sensor node. The former implements the node's application logic, whereas the latter simulates the hardware, as regards for instance the consumed or produced energy (for radio, and batteries, respectively).

Following a different philosophy, SENSE (SEnsor Network Simulator and Emulator) [9] shows a component-oriented architecture, and is a discrete event simulator, like ShoX [10]; both are specifically designed to provide extensibility and reusability of modules, but do not allow to directly transfer simulation code into real nodes as they rely on general-purpose languages such as C++ and Java.

SENS [11] (Sensor, Environment and Network Simulator), and NetTopo [12] also have modular architectures; the former is noteworthy in that it embeds a mechanism for modeling the physical environment thus allowing to capture the influence of the environment on signal propagation, whereas the latter exploits the integration between the simulated environment, and real testbeds. Such hybrid simulation is however unidirectional since real nodes cannot receive messages from simulated ones.

Scalability issues are effectively addressed in DiSens [13] by relying on a distributed-memory parallel cluster system; its approach to simulation, however, is intrinsically different from what discussed here since simulated applications are forced to rely on the emulation of different kinds of sensor hardware.

We rather chose to follow the "real code" approach that provides the key capability to quickly switch between simulation and deployment, and ensures a close correspondence of simulation code to code executed in real nodes. Its most notable example is probably TOSSIM [7], the TinyOS simulator; TOSSIM translates hardware interrupts into discrete simulator events, which are guaranteed to be handled in the correct order; unfortunately this is not sufficient to provide timing guarantees. TimeTOSSIM [14] is an extension that aims to provide source code-level instruments in order to assess simulation timing; in particular, it resolves the conflicts due to the simultaneity of simulated events by interfering with the TOSSIM scheduler, and prioritizing its queue.

The authors of [15] have recently proposed a somewhat similar approach that suggests the introduction of "sensor network checkpoints" between simulation and testbed to store the state of the simulation so that its execution may be deterministically repeated, or that rollbacks may be executed to restore the network state to previous conditions.

Finally, a hybrid approach to simulation is proposed in [16], whose authors augment the simulation environment by means of transparent addition of actual nodes. The interaction between the two kinds of nodes is implemented via specialized nodes acting as intermediary, and the overall approach shares many similarities with what we propose here. The authors of [16] aim to provide fidelity by distributing the computational load of the simulation process; in order to do so, they re-implement many of the core components of the simulator, such as the radio model, and the battery model. Our approach differs in that it is based on the bare TOSSIM simulator, and the introduced modifications are totally transparent to the users, so that they may be immediately able to test their existing code. Moreover, we intend to exploit the presence of actual nodes in the simulation to construct predictive models from their sensed data, to be used to tune the behavior of virtual nodes, thus increasing simulation realism.

III. A Hybrid Approach to WSN Simulation

We implemented our hybrid WSN simulator as part of a comprehensive framework for the creation, management and visualization of test scenarios. The basic structure of our tool is shown in Figure 1.

Our framework has been implemented in Java, and it may be used with any system where the TinyOS toolchain

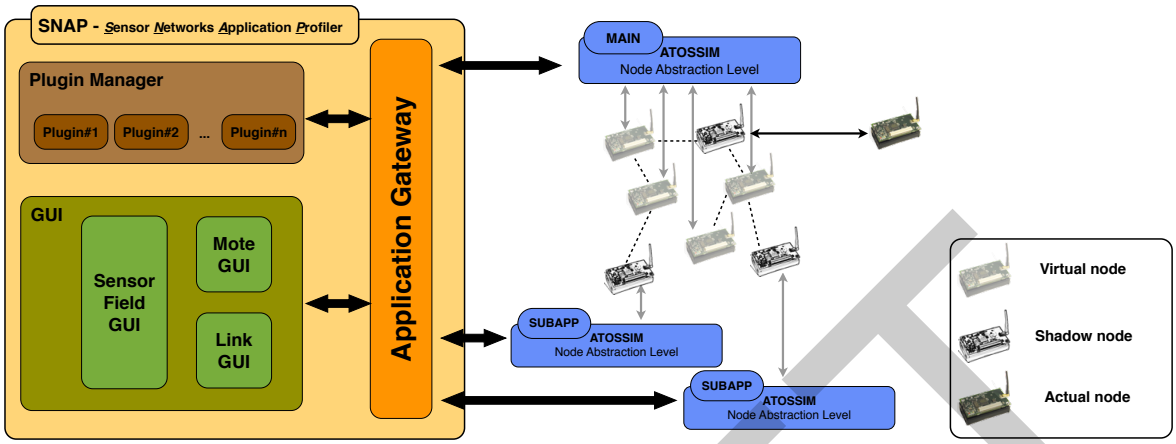


Figure 1. High-level representation of structure of the simulation framework.

can be installed; its GUI allows to create and manage a simulation scenario by setting the nodes positions on the virtual sensor field, managing their links, and customize nodes behavior. Once the virtual network is set up, the actual simulation process is handed over to an augmented version of TOSSIM ("ATOSSIM"), that has been customized in order to introduce the notion of *shadow* nodes, i.e. wrappers whose main purpose is to act as interfaces toward their real counterparts, while appearing identical to other virtual nodes from the simulator point of view. The main function of shadow nodes is thus to collect sensed data from the real world, and to re-route communication from virtual nodes to actual ones.

The typical application scenario we are considering here consists of a minority of actual nodes immersed into a larger virtual network; this may be the case during the development of non traditional applications, such as for instance underground environmental monitoring, where sensor nodes may easily have high individual cost, and a predictive model for their behavior is typically difficult to obtain. Moreover, experience suggests that in a complex WSN it is unlikely that all nodes run the same application logic; this typically occurs when nodes establish some kind of hierarchical relationship (e.g. in cluster-based networks), or more simply when the base station has to perform some specific operation before forwarding sensed data to the storage computer.

This characteristic is impossible to simulate with a tool like traditional TOSSIM; our framework, on the other hand, exploits shadow nodes also to simulate nodes that run code different than the main application, thus implementing a sub-application behavior, seamlessly integrated into the simulation. This is obtained by having several instances of the simulator running, each with its own message receiver; all the instances are coordinated by the application gateway, while one "main" instance of the simulator is devoted to coordinate virtual and shadow nodes.

Finally, the framework functionalities may be specialized or extended via the provided plugin subsystem. Utility plugins have been developed to implement common WSN tools for debugging and visualization purposes, as well as to provide simulation management tools, to allow for the creation of data models from real data, and finally to implement assessment tools.

A. Ensuring Soft Real-time Behavior: Timing Issues

In order to ensure correct coordination among virtual, shadow, and actual nodes, simulation timing must be constrained to satisfy at least soft real-time guarantees; in our architecture this functionality is provided by the application gateway. According to the TinyOS architecture, an application contains a scheduler used to hold the events generated by the application components. TOSSIM is a discrete event simulator, and it follows the same pattern: whenever a virtual node fires an event, this is labeled with the current simulation time and appended to the scheduler, together with the corresponding event handler; the specified actions will be executed upon selection by the scheduler according a non-preemptive, FIFO scheduling policy.

Simultaneous events are handled by TOSSIM by updating the simulation time only for the first event, and leaving it unchanged until the last one is processed. For instance, while managing reception events for nodes that share a common neighboring sender, the simulation time does not advance while all those events are handled. The TOSSIM time model may thus be heavily influenced by the number of simulated nodes: if only one simulated node is present, the simulation time will flow smoothly; on the other hand, if the simulation includes several nodes, the occurrence of several simultaneous events, albeit for different nodes, will prevent the simulation time from advancing while they are processed.

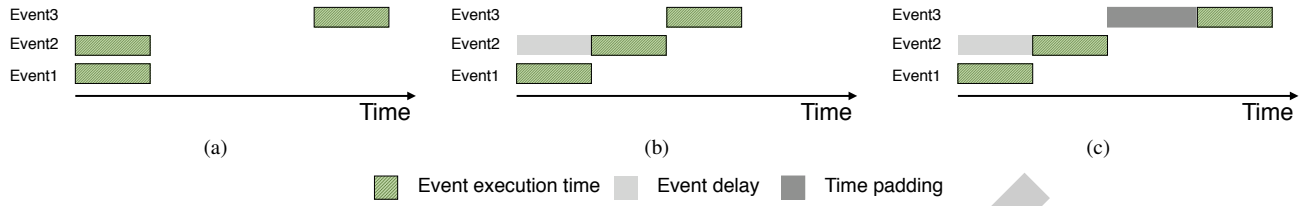


Figure 2. Event scheduling in simulation (a), according to standard TOSSIM processing (b), and with our modifications to ensure timeliness (c).

In the classical TOSSIM timing model, simulation time is totally decoupled from real time, and simulation is executed at maximum hardware speed, compatibly with available resources; for instance, Figure 2a depicts two simultaneous simulated events, followed by another event, after some idle time. Event execution duration is not relevant in "simulation time", but Figure 2b shows how the three events are actually handled by the simulation host processor, that simply enqueues them, and processes them one at a time. Sequencing of events is preserved, but in our scenario, this would pose a problem for the interaction between virtual and actual nodes, whose events would likely take longer to process because of their slower processors. Meeting soft real time constraints implies binding simulation timing to real timing, measured via the system clock; for instance, in order to match simulation time with real time for *Event3* in Figure 2c, a "padding" delay must be artificially added, computed as the difference between the time when the event is selected by the scheduler and the time stored in the event handler descriptor. In other cases, however, the time spent for handling previous events causes subsequent ones to be delayed (as for *Event2* in Figure 2c). Unlike TOSSIM, our simulator thus uses separate threads for event queue management and for simulation time updating; in particular we make use of the *nanosleep()* Unix system call in order to obtain $10\mu s$ refresh time for this thread. If the event handler processing time is low on average, and if simultaneous execution of simulated events does not frequently occur, then soft real time constraints may be met; unfortunately, as pointed out in [17], simulation execution time may even be double than real time in the worst case.

B. Simulating Different Applications in the Same Test Run

As already mentioned, all nodes in a simulated network in TOSSIM are bound to execute the same application, so if different behaviors are needed, the programmer needs to resort to conditional instructions to be executed by the differing nodes only; since dynamic memory allocation is not allowed, this will deprive the majority of nodes of part of their available memory; as a consequence, the behavior of the simulated network will not reliably mirror that of the future deployment.

A typical application, however, is based on the interaction between a limited number of nodes with specialized tasks and the majority of nodes sharing the same behavior. Our framework thus allows several instances of the simulator to run at the same time; the main instance will run the code for the majority of nodes, whereas each other instance will run a different application for each of the specialized nodes. The different instances may communicate via shared memory, and synchronization is guaranteed by the time updating thread, that accepts modifications only from the main simulator instance.

C. Interaction between Virtual and Actual Nodes

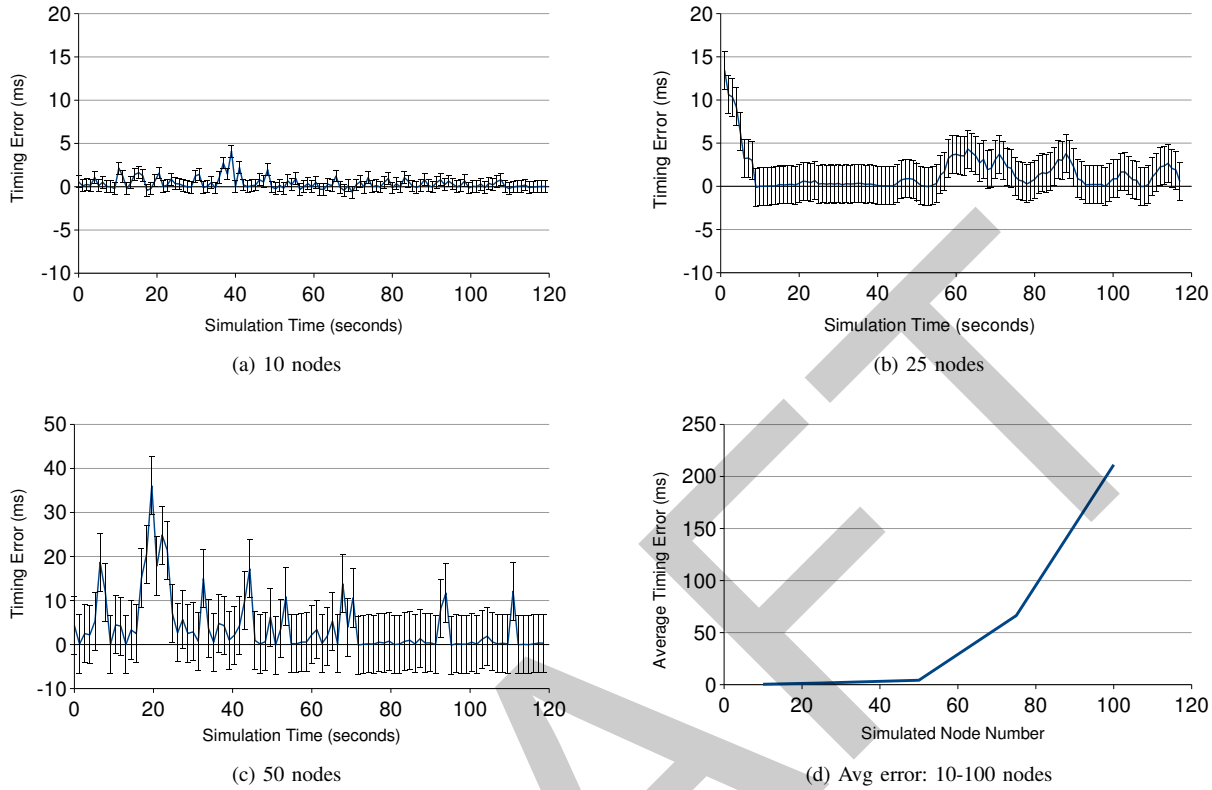
Our framework allows for bi-directional message exchange among virtual and actual sensor nodes, thanks to the presence of *shadow* nodes that act as a bridge between the two worlds. This may be exploited in order to tune the behavior of virtual nodes, and to make them as realistic as possible; in particular they may simulate sensing of data thanks to the information obtained from real nodes.

Although introducing actual data into a simulation usually requires constructing statistical models, after a considerable amount of data is preliminarily stored, which is usually very time consuming. Our approach instead aims to exploit the potential spatial and time correlation among data sensed by actual nodes in order to build a model for the whole network starting from these sparse data. In particular, plugins have been developed that construct regression models for temperature data by using kernel based methods, such as radial basis functions.

IV. Assessment of the Proposed Approach

This Section presents some of the experiments that have been carried on in order to evaluate our hybrid simulator; in particular a few scenarios will be analyzed in order to assess the effectiveness of the interactions between ATOSSIM, the central framework and the real nodes.

All tests have been conducted on a PC with an AMD Turion^{TM}64 processor, Mobile Technology MK-36 (2.0 GHz), with 2 GB RAM. The operating system was a Linux distribution, namely Ubuntu with the 2.6.24-19-generic (x86_64) kernel; we used TinyOS version 2.1. For



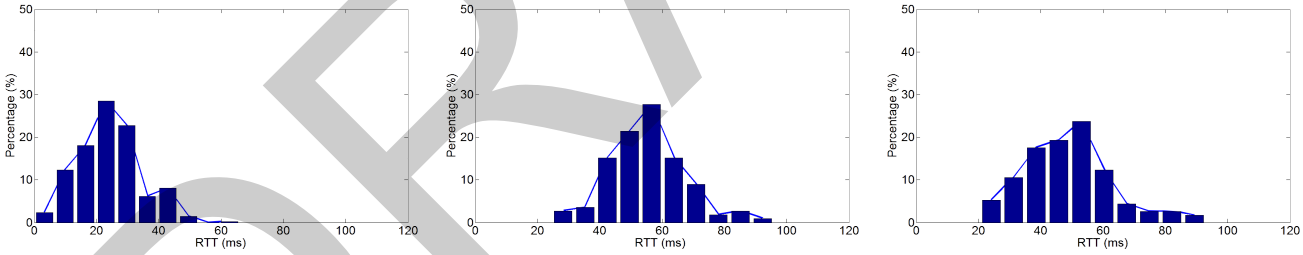
(a) 10 nodes

(b) 25 nodes

(c) 50 nodes

(d) Avg error: 10-100 nodes

Figure 3. Simulation results



(a)

(b)

(c)

Figure 4. Average Simulation RTT. (a) RTT for two real nodes, 1 hop away; (b) RTT for virtual node to real node, 1 hop away; (c) RTT for main application node to sub-application node, 1 hop away.

actual nodes, we chose the common TelosB motes, that mount a Texas Instrument MSP430 microprocessor and a Chipcon CC2420 radio chip; such nodes do not need a separate programmer, and are also equipped with embedded sensors; experiments were carried on by directly connecting them to the PC through their USB interface.

First of all, we wanted to assess the capability of our framework to provide soft real-time simulation, i.e. we want to measure how close is simulation processing time to real processing time. We considered topologies with a number of nodes ranging from 10 to 100, deployed according to a uniform random distribution, and used the *BlinkToRadio* application, i.e. a simple demo application that triggers periodic message sending from each node. The message sending period for the application was set to $\tau = 1s$, and we measured the interval between the times at which two consecutive sending events are popped out of the management queue and processed. In the ideal case, this should be equal to τ, and the difference represents the timing error due to simulation overhead. Figures 3a-3c show the error for topologies with 10, 25, and 50 nodes respectively. Each point was computed as an average among the nodes in the topology, and across 5 different runs.

Figure 3d plots the error as a function of the number of nodes in the network; in our implementation a number of nodes higher than 50 causes a noticeable error, likely wors-

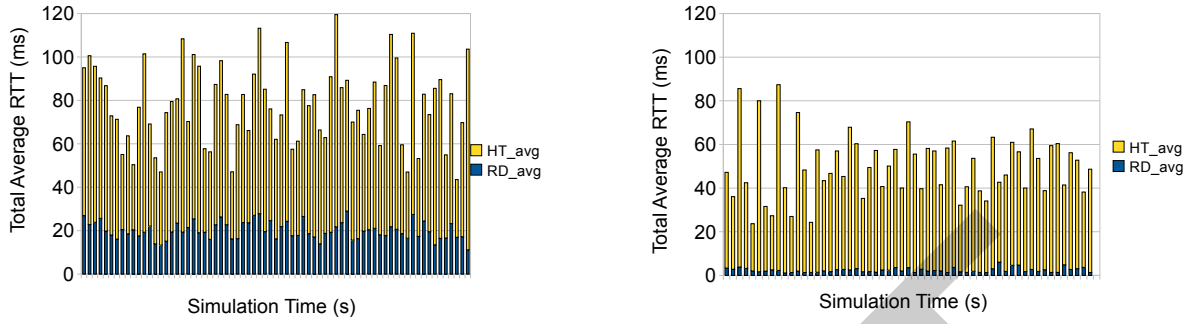


Figure 5. Decoupling the RTT components for two single hop scenarios involving virtual nodes.

ened by the intrinsic non parallel nature of the underlying TOSSIM simulator.

In order to obtain a useful comparison, we reproduced some of the experiments reported in [16]; in particular, we consider the communications between two nodes running different applications. We measured the RTT of a transmission between a virtual node in the main simulation application and an actual node, as well as the RTT between the main simulation application, and another simulation application. Figure 4a represents the base case, and shows how RTT varies for the communication between two real nodes; the histograms show that the most frequently occurring RTT is slightly higher than 20ms. The same code used for real nodes is also used to repeat the experiments with the virtual topology, and Figure 4c shows the distribution of average RTT on a single hop (main application to sub-application, both with virtual nodes). Finally, Figure 4b shows the same measurement, but this time regarding the virtual-actual communication. The three distributions present a similar shape, although the highest frequency RTT in cases (b) and (c) is consistently higher that the base case. It is worth noting that we had to modify the CSMA/CA mechanism used by TOSSIM as it implemented a backoff strategy that forced nodes to randomly delay their initial transmission; this produced an almost uniformly spread distribution for the RTT values, corresponding to a very high variance (this only regarded the initial transmission; the backoff mechanism for collision recovery was not modified).

Figure 5 is meant to provide some insight on the cause of higher RTT values for simulation than in the real case; in fact, when we divide the total average delay into its components (i.e. the routing delay, RD_{avg} and the event handling time, HT_{avg}) we notice that the second component is predominant. This means that the main source of delay is due to TOSSIM event handling, whereas the time spent bu our framework for routing purposes is minimal. The "routing delay" between a virtual and a real node is on average 19.67ms, while in the case of two virtual nodes belonging to different applications this is 2.58ms.

V. Conclusion and On-going Work

We presented here a hybrid simulation framework that exploits the interaction between virtual and actual sensor nodes in order to provide reliable simulation. We have showed how the resulting simulator has promising characteristics in terms of soft real time requirements; together with the possibility of running the same code both in simulation and in the deployed nodes, we believe that the presented tool might represent a valuable support during WSN prototyping.

Our current research aims to investigate a way to further improve the performance of our tool, and we believe that the main issue to be addressed relies in the intrinsically sequential nature of the underlying simulator, so we are currently investigating on possible parallelization techniques.

Moreover, we want investigate the potentialities of the presented approach as regards its use as a model generator, with particular regard to the exploitation of hidden correlations among actual sensed data.

Acknowledgment

The authors would like to thank Salvatore Fiduccia for his help during implementation of a preliminary version of the framework.

References

[1] D. Estrin, L. Girod, G. Pottie, and M. Srivastava, "Instrumenting the world with wireless sensor networks," in *Proc. of Int. Conference on Acoustics, Speech, and Signal Processing (ICASSP 2001)*, Salt Lake City, Utah, May 2001.

[2] I. Akyildiz and E. Stuntebeck, "Wireless underground sensor networks: Research challenges," *Ad Hoc Networks*, vol. 4, no. 6, pp. 669–686, 2006.

[3] J. Cui, J. Kong, M. Gerla, and S. Zhou, "The challenges of building scalable mobile underwater wireless sensor networks for aquatic applications," *IEEE Network, Special Issue on Wireless Sensor Networking*, vol. 20, no. 3, p. 12, 2006.

[4] G. Werner-Allen, P. Swieskowski, and M. Welsh, "Motelab: A wireless sensor network testbed," in *Fourth International Symposium on Information Processing in Sensor Networks, 2005. IPSN 2005.*, 2005, pp. 483–488.

[5] D. Kotz, C. Newport, R. Gray, J. Liu, Y. Yuan, and C. Elliott, "Experimental evaluation of wireless simulation assumptions," in *Proceedings of the 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems*. ACM New York, NY, USA, 2004, pp. 78–82.

[6] L. Gatani, G. Lo Re, and M. Ortolani, "A logical framework for augmented simulations of wireless sensor networks," in *SMC '06. IEEE International Conference on Systems, Man and Cybernetics*, vol. 2, 2006, pp. 1455–1461.

[7] P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications," in *Proc. of the First ACM Conference on Embedded Networked Sensor Systems (SenSys 2003)*, 2003.

[8] S. Park, A. Savvides, and M. B. Srivastava, "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*, 2000, pp. 104–111.

[9] C. G., J. Branch, M. J. Pflug, L. Zhu, and B. Szymanski, "SENSE: A sensor network simulator," *Advances in Pervasive Computing and Networking*, pp. 249–267, 2004.

[10] J. Lessmann, T. Heimfarth, and P. Janacik, "ShoX: An easy to use simulation platform for wireless networks," in *Tenth International Conference on Computer Modeling and Simulation*, 2008.

[11] S. Sundresh, W. Kim, and G. Agha, "SENS: A sensor, environment and network simulator," in *Proceedings of the 37th Annual Simulation Symposium (ANSS04)*, 2004.

[12] L. Shu, C. Wu, and M. Hauswirth, "NetTopo: Beyond simulator and visualizor for wireless sensor networks," Digital Enterprise Research Institute (DERI), Tech. Rep., 2008.

[13] Y. Wen, R. Wolski, and G. Moore, "Disens: scalable distributed sensor network simulation," in *Proceedings of the 12th ACM SIGPLAN symposium on Principles and practice of parallel programming*. ACM New York, NY, USA, 2007, pp. 24–34.

[14] O. Landsiedel, H. Alizai, and K. Wehrle, "When timing matters: Enabling time accurate & scalable simulation of sensor network applications," in *International Conference on Information Processing in Sensor Networks*, 2008.

[15] N. Finne, J. Eriksson, N. Tsiftes, T. Voigt, A. Dunkels, and F. Österlind, "Sensornet checkpointing: enabling repeatability in testbeds and realism in simulations," in *Proceedings of EWSN 2009: 6th European Conference on Wireless Sensor Networks*, Cork, Ireland, 2009.

[16] Y. Wen and R. Wolski, "Simulation-based augmented reality for sensor network development," in *Proceedings of the 5th international conference on Embedded networked sensor systems*. ACM New York, NY, USA, 2007, pp. 275–288.

[17] C. S. Metcalf, "Tossim live: toward a testbed in a thread," Master's thesis, Colorado School of Mines, 2007.