Modelling the Impact of Software Components on Wireless Sensor Network Performance

Óscar Gama, Paulo Carvalho
Department of Informatics
University of Minho, Braga, Portugal
e-mail:{osg,pmc}@di.uminho.pt

P. M. Mendes
Department of Industrial Electronics
University of Minho, Braga, Portugal
e-mail:paulo.mendes@dei.uminho.pt

***Abstract*— Network Simulators are often used to study multiple aspects of data communications in distinct scenarios, including wireless sensor networks (WSN). However, the performance of the software components running in the network nodes is normally neglected by the simulators. This aspect is particularly important in WSNs, as nodes have very limited computing resources. In order to study the impact of software components on WSN performance, a simulated WSN and a physical WSN were setup in the IEEE 802.15.4 domain. Tests revealed that the simulator must take into account the software components of the WSN to produce realistic results. To achieve this, new parameters are proposed to model the impact of the software components on a physical WSN. Tests measuring the packet round-trip delay, delivery error ratio, and duplicated packet ratio showed that the inclusion of this model in a simulator improves significantly the accuracy of the results when compared with those obtained in a physical WSN.**

I. INTRODUCTION

A wireless sensor network (WSN) is composed of tiny resource-constrained devices owning communication and sensing capabilities. WSNs have a wide range of potential applications, such as, environment monitoring, smart buildings, medical care, and industrial control [1].

Many studies have been carried out in the WSN research community, where algorithms and protocols are carried out mostly in simulators. A review based on 151 wireless network articles from a five-year-period reported that 76% from those works used simulations [2]. The preference for simulation tools is justified by the difficulty of deploying real networks, as programming a lot of motes, gathering the performance metrics of the motes, and managing the power sources is tedious and time-consuming. The economical costs required to build a real testbed is another obstacle. Because WSNs use distributed programming, and debuggers are hard to use in the motes, software errors are harder to detect and correct in a testbed than in a simulator. Testbeds also impose strong constraints on the topology and size of the network. On the other hand, simulators allow building and modifying network scenarios easily, the models are easily monitored from the global view of the simulator, and the experiments are reproducible. A comparison of simulators for WSNs is provided in [3, 4].

WSN simulation studies use frequently unrealistic assumptions, such as, flat physical environment, circular radio transmission area, equal range for all radios, channel with bidirectional symmetry, simple relation of signal strength with distance, and no fading or shadowing phenomena. A large set of measurements showed that these assumptions cause simulation results to differ significantly from experimental results [5].

Since simulators can use different models to represent the same physical phenomenon, appreciable divergences in the results may be obtained using distinct simulators. The performance results of a simple algorithm using diverse simulators proved this fact [6]. Furthermore, models cannot represent reality with absolute accuracy [7]. Simulation scenarios can also ignore diverse hardware and software aspects that may influence the final results. Examples of these aspects are the time required by the base-station (BS) and the motes to process the incoming or outgoing packets, the queuing delay in the transmission and receiving buffers, the time required to switch channels or between transmitter and receiver mode, and the link speed between the BS and the decision center, as explained later. Moreover, simulation tests usually do not consider any external interfering traffic on the WSN. This aspect is important when the WSN operates in license-free bands. For example, an IEEE 802.15.4 WSN operating in the 2450 MHz band may have to share channels with IEEE 802.11 networks. These aspects may lead to simulation results significantly different from those obtained in a real WSN.

Studies presenting experimental validation tests of simulators against results obtained in real networks are not abundant, because of the difficulty of implementing a real testbed.

The accuracy of the ns-2 simulator is evaluated in [8]. The authors compare the network characteristics of a simulated, an emulated, and a real IEEE 802.15.4 multi-hop mesh network with sixteen stations in a static indoor environment. The results showed that the packet delivery ratios, the connectivity graphs, and the packet latencies are represented in the simulated model with an average error of 0.3%, 10%, and 57% respectively.

The experimental validation results for the SWAN simulator showed that the simulations with the two-ray ground

radio propagation model differ from reality with around 80%, while with the shadowing model differ about 10% in an outdoor mobile IEEE 802.15.4 network [5][9].

The reliability of OMNeT++ is evaluated in [10]. The authors consider an experimental setup made of six motes to test the performance of the flooding algorithm. The results of the testbed are compared with the results of the simulations of the same scenarios on OMNeT++. Experiments showed that simulation results tend to over-estimate the metrics collected in the testbed. The authors do not present any explanation for the difference noted in the results.

To validate some high-level aspects of Castalia [11], the authors of this WSN simulator deployed a real network involving nine motes [12]. Important differences in the results from the real network and the simulation were noted. The authors of these works were unable to justify satisfactorily the registered differences.

Aware of the difficulty that a simulator may have in presenting accurate results, this work studies software-related aspects of a WSN that contribute to the differences found in simulation results against real measurements. This is an important topic that is usually neglected in WSN simulations. First, it is evaluated in the IEEE 802.15.4 domain how different the results obtained in a simulated WSN are from those obtained in an analogous physical scenario. Then, the causes of the divergences in the results are identified. At last, a model using empirical software-related parameters to improve the accuracy of the simulation results is proposed. Instead of trying to present accurate values for the model parameters, which are necessarily specific to each testbed, this work intends to model software-related issues which have influence on the testbed results, and which may also occur in another WSN testbed. The main contribution of this paper is to present a parameterized model reflecting the impact of the software components on a physical WSN. The proposed model is generic to be easily implemented in current WSN simulators, being also an important contribution for future development of simulation tools.

The remaining of this paper is structured as follows: the simulated and physical platforms, as well as the test conditions used in the experiments are presented in Section II; the results obtained in the tests are shown in Section III; new parameters are proposed for the simulator in Section IV; the simulation results using the new parameters are presented in Section V; finally, the conclusions are discussed in Section VI.

II. Experimental Platforms

The physical and simulated experimental platforms, as well as the test conditions used in this work are presented next.

The reference testbed is composed of sixteen ZigBit-A2 motes placed statically in a semi-circle around the BS, about one meter away from the BS. To evaluate the impact of software components in the performance of a WSN, a static small-area WSN was adopted to minimize the effects of additional source of errors, such as, nodes mobility and fading phenomena. The reference testbed cannot admit more than sixteen motes due to the RAM memory limitation of the BS. Indeed, a minimum amount of memory in the BS is required to hold data for packet statistical analysis, and this memory is dependent on the number of active motes in the WSN.

The ZigBit-A2 mote is an IEEE 802.15.4/ ZigBee-compliant module operating in the 2.4 GHz frequency band. Motes contain one AT86RF230 transceiver coupled to a dual chip antenna, and one ATmega1281V microcontroller comprising 128 kB flash memory, 4 kB EEPROM, and 8 kB SRAM. Motes run the TinyOS operating system. This testbed uses the BS included in the kit available from the manufacturer. Since the BS is built in around a ZigBit-A2 module, in terms of software performance the BS is identical to a mote. The IEEE 802.15.4 standard defines the physical layer and medium access control (MAC) layer specifications for low data rate WSNs. It specifies a maximum physical packet size of 133 B.

To study the validity of the model proposed in this paper at a different test scenario, controlled traffic from another IEEE 802.15.4 WSN is admitted on the channel used by the reference testbed. The reference WSN and the interfering WSN have distinct personal area network identifications, and are close enough to sense the carrier signals mutually.

The scenario described for the physical testbed was equally implemented in the Castalia simulator. Castalia is an open-source, discrete event-driven simulator, programmable in C++, and designed specifically for WSNs. It uses the communication model proposed in [13]. Castalia provides parameters to model the physical layer in accordance with the transceiver characteristics, and packet buffers to all communication layers. Castalia also features clock drift, sensor bias, sensor and CPU energy consumption, and monitors resources such as memory usage and CPU time.

A. Test Conditions

In the reference WSN, each mote transmits to the BS a packet with a total length of 107 bytes (B) (17 B of physical and MAC overhead plus 90 B of MAC payload) every 250 ms approximately. In the interfering WSN, a mote sends a packet of fixed size (100 B of MAC payload) to the BS every 50 ms approximately.

The non-slotted CSMA-CA MAC protocol described in IEEE 802.15.4 standard was used in the reference WSN and interfering WSN. In the reference WSN, the CSMA-CA algorithm used the default parameters: the minimum backoff exponent is three, the maximum number of backoffs is four, and the maximum number of frame retries is three. The interfering WSN also used these parameters except the maximum number of frame retries, which is zero to guarantee that the CSMA algorithm execution ends before fifty milliseconds.

The reference WSN was configured to operate in a wireless channel free of IEEE 802.15.4 traffic. For this purpose, a channel analyzer was used to find free channels. It was selected the channel 25 of the IEEE 802.15.4 spectrum. To reduce the impact of spurious interferences on this channel, motes transmit at maximum power (3 dBm).

The BS of the reference WSN is connected to the serial port of a computer. The serial link rate is 500 kbit/s. It was noted that when the BS is sending data to the computer, the capacity of the BS to receive or transmit packets becomes significantly reduced. To reduce the influence of this aspect on the final results, the BS sends to the computer every two minutes only the relevant statistics of the traffic flow received from each mote relative to this time period.

The tests were carried out in the physical testbed and in the simulator for an increasing number of active motes in the WSN, with and without IEEE 802.15.4 interfering traffic in the selected channel. The test duration was sixteen minutes for each set of active motes. The results obtained are presented next.

III. EXPERIMENTAL RESULTS

The results for round-trip (RT) delay, and Delivery Error Ratio (DER) obtained both in the physical testbed and in the simulator are discussed in this section. Both metrics are considered from the perspective of the application layer. In the context of this paper, round-trip delay is defined as the time spent between sending an application data packet from a mote and the success confirmation of the operation, which occurs after receiving the MAC acknowledgement frame from the BS. As the round-trip delay of a packet is calculated using only the clock of the source mote, no time synchronization mechanism is required. Each packet carries in the payload the round-trip delay of the packet sent previously. DER expresses the probability of failing the delivery of an application data packet sent from a mote to the application layer of the BS. The results for the duplicated packet ratio will be shown in section V.

A. Tests without interfering traffic

Figures 1 and 2 present the results obtained when IEEE 802.15.4 interfering traffic was not present. Represented in a logarithm scale, the graphical curves of Fig. 1 show the simulation results for the DER when increasing the number of motes sending packets to the BS. The graphical bars correspond to the DER obtained in the physical testbed. For each number of active motes in the WSN, it is represented the maximum, average, and minimum DER values. For example, if five motes are active in the physical testbed, the DER considering all packets received by the BS from all motes is 0.8% (average value); the DER considering only the traffic flow from the mote that presented more undelivered packets is 1.0% (maximum value); the DER considering the traffic flow from the mote that presented less undelivered packets is 0.5% (minimum value). Figure 2 shows the maximum and average round-trip delays obtained in the simulator and in the physical testbed.

In Figure 1, while the simulation results reveal a WSN scaling up to 16 nodes with a maximum DER always below 1%, the physical testbed results show that above six active motes the maximum DER becomes higher than 1%. Figure 2 reveal that maximum and average round-trip delays obtained in the simulator are significantly distinct from the real results.

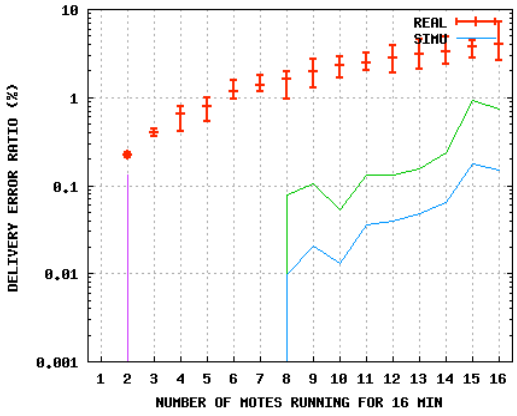

Figure 1. DER without interferences

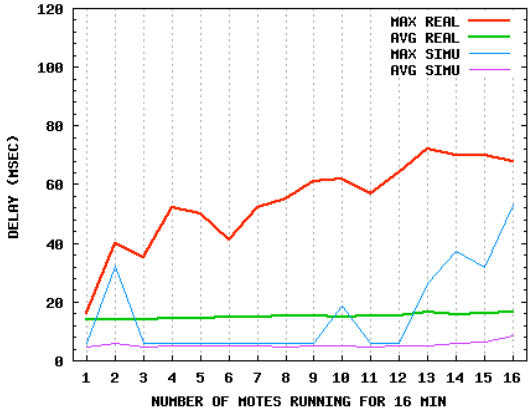

Figure 2. RT delay without interferences

B. Tests with interfering traffic

Figures 3 and 4 present the DER and round-trip delay results obtained in presence of interfering traffic. The results shown in both figures were obtained in the simulator and in the physical testbed. As expected, the network performance degrades before the presence of interfering traffic. The differences in the results registered in the physical testbed and in the simulator are considerably distinct.

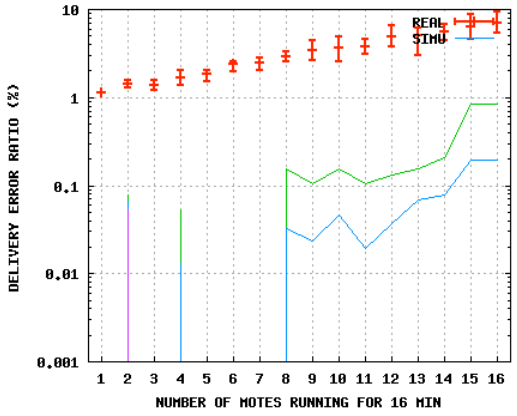

Figure 3. DER with interferences

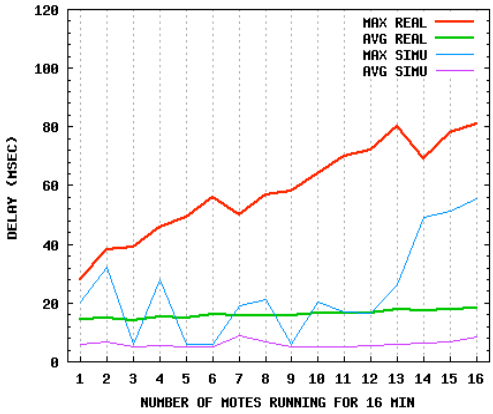

Figure 4. RT delay with interferences

IV. NEW SIMULATION PARAMETERS

In this section, the main causes for the divergence in the results obtained in the simulator and in the physical testbed are identified and discussed. As result of this analysis, new parameters are proposed for the simulator in order to minimize the differences to the testbed results.

The first reason for the differences observed in the results is that the simulator does not take into account the time to process the protocol layers software code, as well as the behavior of the operating system used in the network devices. As TinyOS can only schedule and handle single events, and computing resources are very limited, significant delays may occur in scheduling and processing those events.

This overhead in terms of delay may be responsible for packet loss. To understand why, let us suppose that a packet has been received by the BS transceiver. After processing it, the physical layer software triggers events to forward the payload to the upper protocol layers. Since the delivering time to the application layer is not null, another packet may be received by the BS transceiver during this transactional phase. In this case, the operating system cannot attend the hardware interrupt from the transceiver indicating that a new packet is ready to be transferred to the microcontroller. Consequently, the new received packet is dropped. This situation was confirmed experimentally.

To implement this behavior in the simulator, a delivery time parameter was introduced: $T_{mac \rightarrow app}$. This parameter indicates the time required to process the packet at MAC layer and deliver the data to the application layer. Therefore, this parameter reflects both the event scheduling delay and the packet processing delay imposed by the link, network, and transport layers. The delivery time parameter $T_{phy \rightarrow mac}$ was also implemented to reflect the time required to process the packet at physical layer and deliver the payload to the MAC layer. The process time parameter T_{app} indicates the time required for the application layer of the BS to process the received payload. So, an incoming packet is completely processed by the application layer of the BS after a time interval $T_{BS\ totRX}(n)$:

$$T_{BS\ totRX} = T_{BS\ phy \rightarrow mac} + T_{BS\ mac \rightarrow app} + T_{BS\ app} \qquad (1)$$

Analogously, T_{totTX} is the total time required for a mote to transmit an application packet. Hence, the application packet delay comes increased by the sum of T_{totRX} and T_{totTX}.

The computing performance of the BS in the physical testbed is similar to a mote. This situation is not normally found in a WSN, since a BS presents typically stronger computing resources and a more efficient operating system than motes. In this case, the value of T_{totRX} and T_{totTX} may be negligible. However, in a multi-hop WSN the packets may be routed through the motes, and so the value of these parameters can influence significantly the network performance.

The second reason for the differences in the results is that the motes present an appreciable time drift. The cause of this time drift is distinct of the CPU clock time drift, which is typically a few microseconds per second. While the latter is due to physical characteristics of the semiconductor components, the former is mainly due to the CPU internal software performance running under limited computing resources. To reflect this feature, the drift parameter D_{ab} was introduced in the simulator. To set this parameter correctly, measurements were carried out using the BS and pairs of motes. Generically, if the drift between mote *a* and the BS is D_a, and the drift between mote *b* and the BS is D_b, then the drift between mote *a* and mote *b* is $D_{ab} = D_a - D_b$. This means that if mote *a* and mote *b* start to transmit separated in time by T_{ab} seconds, and if $D_a > D_b$, then both motes will contend for the wireless channel after sending T_{ab} / D_{ab} packets. The D_{ab} value can be calculated experimentally through the relation:

$$D_{ab} = ((Ta_{i+1} - Tb_{i+1}) - (Ta_i - Tb_i)) / (Tb_{i+1} - Tb_i) \qquad (2)$$

where Ta_i, Ta_{i+1}, Tb_i, and Tb_{i+1} express the local time of the BS when this received packet *i* and packet *i+1* from mote *a* and mote *b*, respectively. It is assumed that packet *i* from mote *b* arrives after packet *i* from mote *a*, as well as all successive received packets from both motes during the period Tb_i and Tb_{i+1}. Since $T_{ab} < 250$ ms in the physical testbed, and assuming $D_{ab} = 0.1\%$, channel contentions between a pair of motes may occur whenever 250 packets are sent at maximum. However, no channel contention occurs if D_{ab} is zero and T_{ab} is above the full-loaded packet transmission time. In this situation, the simulator results presented a null DER in a WSN with more than sixteen active motes. To prevent this unrealistic situation, the simulation results in Figures 1 to 4 were taken using a D_{ab} equal to 0.005%.

The discussion above shows that the mismatch in the results derives from the limited performance of the software running in the network devices. Since this software performance behaviour is inherent to motes of any WSN, the presented discussion applies generically to all WSNs, especially to WSNs running event-based operating systems.

A. *Setting of the new parameters*

Whenever possible, the tuning of the new parameters was accomplished based on experimental measurements performed in the physical testbed. The values for those parameters are proposed below.

Physical to MAC Layer Delivering Time ($T_{phy \rightarrow mac}$)

Since the IEEE 802.15.4 standard is implemented in the firmware of the transceiver, the time required to deliver a packet from the physical layer to the MAC layer is very hard to measure experimentally. This time includes the period required for the transceiver to send the MAC frame to the microcontroller through the serial peripheral interface. In a ZibBit mote, this period is about 0.29 ms for a full-load MAC frame (127 B). Considering the results of Fig. 1, a delay of 1.2 ms were estimated for $T_{phy \rightarrow mac}$, for data packets carrying 90 B of payload. Tests revealed that this parameter depends on the packet size.

MAC to Application Layer Delivering Time ($T_{mac \rightarrow app}$)

Measurements done with an one-millisecond resolution timer revealed that the time required to deliver a payload of 90 B from the MAC layer to the application layer presented a value of 1 ms, 2 ms, and 3 ms in 10%, 83%, and 7% of all packets delivered to the application layer, respectively. These percentages depend on the payload size. Indeed, the time required to deliver a payload of 30 B from the MAC layer to the application layer presented a value of 1 ms, and 2 ms in 77%, and 23% of all packets delivered to the application layer, respectively. The simulator was programmed so that $T_{mac \rightarrow app}$ varies randomly according to a uniform distribution through these delays in accordance with the respective percentages, and payload size.

Packet Processing Time at Application Layer (T_{app})

Measurements revealed that the time required for the application layer of the BS to process the received payload was around 1.2 ms. Therefore, T_{app} was set to this value.

Total Transmission Time (T_{totTX})

It was assumed that the time required for a mote to deliver an application packet to the transceiver (T_{totTX}) is equal to the time of delivering a packet from the transceiver to the application layer (T_{totRX}). Recall that T_{totRX} is the sum of $T_{phy \rightarrow mac}$, $T_{mac \rightarrow app}$, and T_{app} (see Eq. 1).

Software Time Drift (D_{ab})

Measurements showed that the software time drift between motes may have values up to 0.3%, depending on the pair of motes used. Therefore, the simulator was programmed so that each mote at the start-up chooses randomly a D_{ab} up to 0.3%.

V. Simulation Results with the New Parameters

Figures 5 and 6 show the simulation results using the new parameters when IEEE 802.15.4 interfering traffic was not present. It is observed that the DER simulation results approximate closely to the DER values found in the physical scenario (the corresponding physical testbed results are also replicated for better comparison). The results of the maximum and average round-trip delays become also close to those obtained in the physical scenario.

Figures 7 and 8 present the simulation results when IEEE 802.15.4 interfering traffic was present. The DER results keep close to the DER values found in the physical scenario. The results of the average and maximum round-trip delays are also identical to those obtained in the physical scenario.

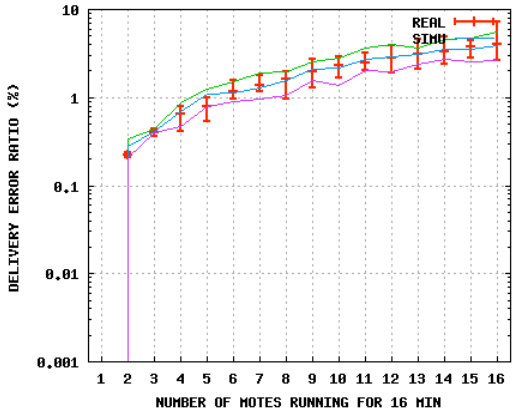

Figure 5. DER without interferences

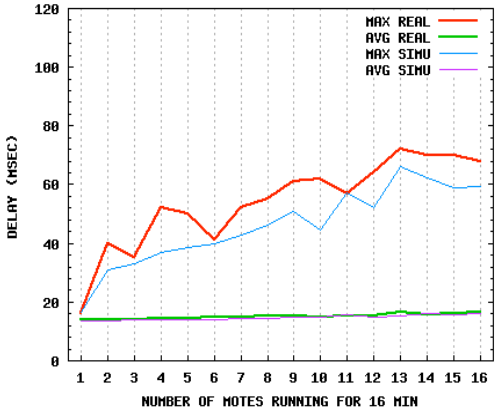

Figure 6. RT delay without interferences

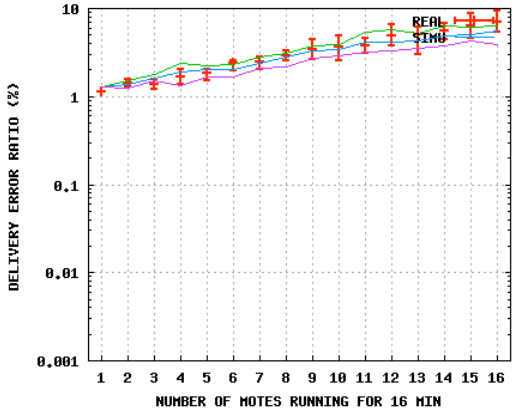

Figure 7. DER with interferences

With the CSMA-CA algorithm, a mote may send a duplicated packet if the acknowledgement packet from the BS is not received by the mote. Duplicated packets must be avoided to save bandwidth and energy consumption. Figure 9 shows the average Duplicated Packet Ratio (<DPR>) obtained with and without the presence of IEEE 802.15.4 interfering traffic, not considering the use of the proposed parameterized model. Figure 10 presents de average DPR considering this model. In this last case, the simulation results are very identical to those obtained in the physical testbed.

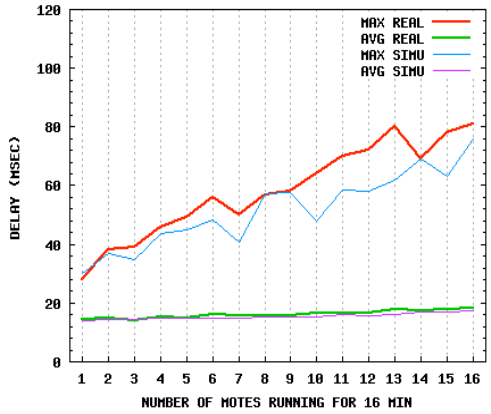

Figure 8. RT delay with interferences

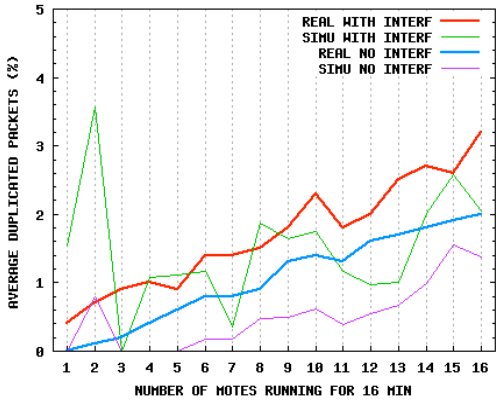

Figure 9. <DPR> without the proposed model

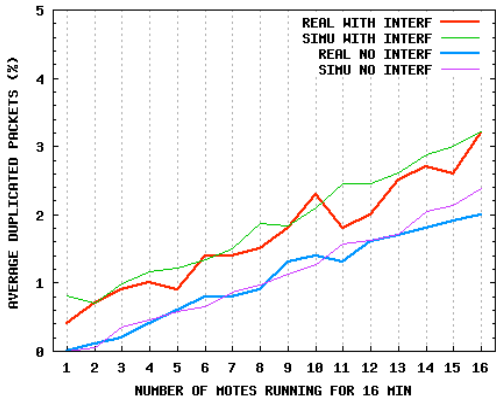

Figure 10. <DPR> with the proposed model

VI. CONCLUSIONS

Since motes present typically very limited computing resources, the performance of the operating system and high-level software running inside the motes impose significant constrains to the overall performance of a WSN. This paper has showed that if such software performance limitations are not taken into account, the simulation tests may produce results significantly more optimistic from those obtained in real conditions. Indeed, tests showed that it is very difficult to obtain satisfactory simulation results using uniquely the parameters of the wireless channel, the physical layer, and the MAC layer provided by the WSN simulator. This very important aspect is often neglected in many works presenting WSN evaluation studies carried on simulators.

In order to obtain satisfactory simulation results, distinct software-related parameters were proposed, measured, and included in the simulator. Simulation tests showed that the average values of the results obtained with the new parameters get satisfactory match to those obtained in real conditions. Therefore, the inclusion of the proposed parameters in the model of a WSN simulator helps to improve the confidence on the simulation results.

ACKNOWLEDGMENT

Óscar Gama work is supported by FCT (SFRH/BD/34621/2007), Portugal.

REFERENCES

[1] J. Yick, B. Mukherjee, D. Ghosal, "Wireless sensor network survey", Computer Networks, Vol. 52, No. 12., pp. 2292-2330, 2008.

[2] S. Kurkowski, T. Camp, M. Colagrosso, “Manet simulation studies: the incredibles,” ACM SIGMOBILE Mobile Computing and Communications Review, vol. 9, no. 4, pp. 50–61, 2005.

[3] C. Singh, O. Vyas, M. Tiwari “A Survey of Simulation in Sensor Networks.” In Proceedings of the International Conference on Computational intelligence For Modelling Control & Automation. CIMCA. IEEE Computer Society, Washington, DC, 867-872, 2008.

[4] M. Korkalainen, M. Sallinen, N. Kärkkäinen, P. Tukeva “Survey of Wireless Sensor Networks Simulation Tools for Demanding Applications.” In Proceedings of the 5th International Conference on Networking and Services - Volume 00. ICNS. IEEE Computer Society, Washington, DC, 102-106, 2009.

[5] D. Kotz, C. Newport, R. S. Gray, J. Liu, Y. Yuan, and C. Elliott. Experimental evaluation of wireless simulation assumptions. In MSWiM ’04: Proc. of the 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems, pages 78–82, New York, ACM Press, 2004.

[6] D. Cavin, Y. Sasson, A. Schiper. On the accuracy of manet simulators. In POMC ’02: Proc. 2th ACM international workshop on Principles of mobile computing, pp. 38–43, New York, ACM Press, 2002.

[7] J. Banks, J. Carson, B. Nelson, “Discrete-Event System Simulation”, 2nd ed. Prentice Hall, 1996.

[8] S. Ivanov, A. Herms, G. Lukas. “Experimental validation of the ns-2 wireless model using simulation, emulation, and real network” 4th Workshop on Mobile Ad-Hoc Networks, 2007.

[9] J. Liu, Y. Yuan, D. Nicol, R. Gray, C. Newport, D. Kotz, L. Perrone, “Simulation validation using direct execution of wireless ad-hoc routing protocols,” in PADS ’04: Proceedings of the eighteenth workshop on Parallel and distributed simulation. New York, NY, USA: ACM Press, pp. 7–16, 2004.

[10] U. Colesanti, C. Crociani, A. Vitaletti, “On the accuracy of OMNET++ in the wireless sensor networks domain: simulation vs. testbed,” in PE-WASUN ’07: Proc. 4th ACM workshop on Performance evaluation of wireless ad hoc, sensor and ubiquitous networks, New York, pp. 25–31, ACM press, 2007.

[11] Castalia: A Simulator for WSN, http://castalia.npc.nicta.com.au.

[12] H. 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, 2007.

[13] M. Zuniga, B. Krishnamachari, “Analyzing the transitional region in low power wireless links,” 1st IEEE Annual Conf. on Sensor and Ad Hoc Communications and Networks, 2004.