On the Credibility of Wireless Sensor Network Simulations: Evaluation of Intrusion Detection System

Andriy Stetsko
Faculty of Informatics
Masaryk University
stetsko@fi.muni.cz

Tobiáš Smolka
Faculty of Informatics
Masaryk University
xsmolka@fi.muni.cz

Filip Jurnečka
Faculty of Informatics
Masaryk University
xjurn@fi.muni.cz

Vashek Matyas*
Faculty of Informatics
Masaryk University
matyas@fi.muni.cz

ABSTRACT

In the field of wireless sensor networks, not many schemes are tested on real hardware. System designers usually give preference to simulations since their preparation and execution require significantly less time and money than experiments on real hardware. In this paper, we present a practical research on four open-source simulators, i.e., Castalia, Cooja, MiXiM and WSNet. Recently, using a simple test case, we demonstrated that usage of different simulators results into different evaluation outcomes even though the simulators are set in the same way, and the same evaluation metric is used – a number of packets received by sensor nodes. We hypothesized possible sources of the differences, but we did not thoroughly examine them. In this paper, we rigorously examine the simulators and present our findings regarding the sources of the differences. Also, we evaluate their impact on the evaluation of a more complex system – the intrusion detection system.

Categories and Subject Descriptors

I.6 [**Computing Methodologies**]: Simulation and Modeling

General Terms

Simulation, wireless sensor network

Keywords

Intrusion detection, Castalia, Cooja, comparison, evaluation, MiXiM, simulator, wireless sensor network, WSNet

*Final work on this paper undertaken as a Fulbright-Masaryk Visiting Scholar at Harvard University.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
Simutools 2012, March 19-23, Desenzano del Garda, Italy
Copyright © 2012 ICST 978-1-936968-47-3
DOI 10.4108/icst.simutools.2012.247708

1. INTRODUCTION

A wireless sensor network (WSN) consists of sensor nodes – standalone devices equipped with a microcontroller, a radio, sensors and a set of batteries. WSNs significantly differ from conventional wireless and wired networks – the devices are constrained in processing power and energy. Also, a WSN is a highly distributed network, often organized in ad-hoc manner and left unattended in a physically unprotected area. Hence, solutions available for conventional wireless and wired networks cannot be directly used in WSNs. A lot of protocols that take the specifics of WSNs into account have been proposed, e.g., medium access control and routing protocols, key distribution and intrusion detection schemes.

In the field of WSNs, researchers usually use simulators for the evaluation of their proposals. Only a small number of them are tested on real hardware. When a network includes dozens or hundreds of sensor nodes, the testing is more time consuming process in comparison to simulations, which provide an easy and relatively fast evaluation of proposals.

In [20], we recently demonstrated that the evaluation results of the same application differ across the simulators even though the simulators are configured in the same way, and the same evaluation metrics are used. Our test case was quite simple – a group of sensor nodes organized into a line, the first node was transmitting while the other were receiving. We compared a number of received packets across the simulators. We hypothesized possible sources of the differences, but we did not thoroughly examine them.

The contribution of this work is the following. Firstly, we rigorously examine the following simulators: Castalia of version 3.2, Cooja, which comes with Contiki 2.5-rc1; MiXiM of version 2.0.1; and WSNet of version 9.07, rev. 484. Secondly, we identify sources of the differences, and finally examine their impact on the evaluation of intrusion detection system (IDS). The choice of IDS is not accidental. We work on a framework that optimizes a network-based IDS for a given network. This work is undertaken in order to understand the differences of simulators, to reveal their design and implementation mistakes, and to investigate their impact on performance evaluation of IDS.

Castalia [17], MiXiM [11] and WSNet [4] are purely network simulators. In contrast, Cooja [14] that comes together with the Contiki operating system [8], allows to perform simulations at three levels of simulation abstraction: at the net-

Simutools 2012, March 19-23, Desenzano del Garda, Italy
Copyright © 2012 ICST
DOI 10.4108/icst.simutools.2012.247708

work level, at the operating system level, and at the machine code instruction set level. We performed simulations at the operating system level. The most precise mode is omitted due to computational requirements and the network level is omitted because it does not implement any networking protocols. When we mention Cooja further in the text, we mean Cooja that simulates a network only at the operating system level of abstraction unless it is stated otherwise.

Simulators often have a modular architecture. They represent a sensor node and a radio medium as a group of modules interconnected between each other. The radio medium may consist of models for radio propagation and noise. The sensor node may consist of models for energy consumption, antenna, physical (PHY), medium access control (MAC), network and application layers. Often, simulators support more than one model for a certain layer. In order to get comparable simulation results, we aimed to choose such energy consumption, radio propagation, noise, antenna, PHY and MAC layer models that were implemented by each of the considered simulators. Furthermore, we aimed to set the selected models in the same way.

We implemented the routing protocol, application, attackers and simple IDS for all four simulators. We evaluated the IDS on the simulators using the same metrics and compared the results between each other. The IDS designed and implemented in this work is simple and quite naïve. However, it was not the goal of this paper to present an effective IDS.

The paper is organized as follows. In Section 3, we describe settings of four considered simulators. We start from generated network topologies, then go through the settings of antenna and radio propagation models, and finally end at the settings of PHY and MAC layers. In Section 4, we present details of the implemented routing scheme, application, attackers and IDS. The same section also contains information on the setting of the implemented modules. We present and compare the results obtained from the simulations in Section 5. In Section 6, we present our findings from more detailed investigation on possible causes of the discovered differences. Section 7 concludes the paper.

2. RELATED WORK

There is a wide range of publications on simulation tools. The publications differ from each other in the level of provided details. Some provide only general summarized information about the simulators. Typically, such works help to get a quick and high-level insight into pros and cons of the existing simulators. In [12], Korkalainen et al. provided a review of ns-2, OMNeT++, Prowler, OPNET and TOSSIM, focusing among others on a type of distribution license, supported hardware platforms, programming / scripting language, mobility support. In [6], Du et al. presented the taxonomy of simulators, dividing them into four groups: network simulators with node models, network simulators with node emulators, node system simulators with network models, node emulators with network models.

Other publications go into more details by evaluating performance or accuracy of existing simulators. Pham et al. validated a wireless channel model of Castalia in [17]. They calibrated the model with the results from a real experiment in such way that connectivity maps obtained from the simulator and real experiment were similar. Also, they implemented the same value propagation application for both the simulator and real hardware (9 TelosB sensor nodes), and compared the results on the application layer. In both cases, they used the same MAC protocol (Tunable MAC). Bergamini et al. evaluated accuracy of Castalia and ns-2 in [2]. The authors compared performance of two protocols (NETSET and Gossip) evaluated in the simulators against the results obtained from real experiments (run on 34 TMote Sky sensor nodes). In [18], Shakshuki et al. evaluated the accuracy of custom based Java simulator and ns-2, demonstrating the differences between amount of energy consumed on real hardware (ESB sensor nodes) and the estimation of consumed energy from the simulators. For both the simulators and real experiment, the authors implemented the same MAC protocol (agent-based S-MAC protocol). Colesanti et al. examined the accuracy of OMNeT++ and its MAC Simulator Framework in [5].

Pediaditakis et al. evaluated Castalia in terms of computational time and memory usage in [16]. In [23], Weingartner et al. compared performance of ns-2, OMNeT++, ns-3, SimPy and JiST/SWANS. Similarly, the authors considered run-time and memory usage of the simulators.

The contribution of our work, which also distinguishes it from other works mentioned before is the following. First, we deeply examine the simulators, looking into their source codes, and identify possible sources of the inconsistencies in the simulation results. Second, we demonstrate the impact of the differences in simulators on the evaluation of a non-trivial system – the IDS. Last but not least, we focus on a different group of simulators, i.e., Castalia, Cooja, MiXiM and WSNet, which were not covered in the literature in such combination so far.

Voigt et al. discussed the issue of incomparable simulation results in [22], but they did not examine deeply the sources of their incomparability. In order to solve the problem partially, they suggested to use common specification language to describe input parameters and output statistics. The work was followed by Li et al. in [13]. The authors focused on the idea of making simulators cooperate by using WiseML – a common format for inputs and outputs.

3. SIMULATOR SETTINGS

In order to get comparable results from different simulators, it is required to run simulations on the same topology and to use the same radio propagation, noise, antenna, PHY and MAC layer models in these simulators. Furthermore, it is necessary to configure these models in the same way. All of the considered simulators provide models for omni-directional antenna and free space radio propagation. Castalia, MiXiM and WSNet implement models for 2.4 GHz CC2420 radio and 802.15.4 unslotted CSMA-CA protocol. We used these models in our simulations and aimed to configure them in the same way. Cooja is a directive environment for node's code to run in. All the networking layers above the physical are part of the operating system (OS) used by nodes. Although we used a different networking stack [7] given by the Contiki OS, we aimed to set it in a similar way to other simulators. The more detailed description of the models used in the simulations and their configuration is provided in the following subsections.

3.1 Topology

We generated two static topologies – *dense* and *sparse* – both in two dimensional space. For this purpose, we used `wsnet-topogen` tool, which comes together with WSNet.

Both topologies include 250 sensor nodes and one base station. In the dense topology, the sensor nodes are uniformly distributed over an 50 m × 50 m area. The sparse topology was generated by "stretching" the dense topology over an 200 m × 200 m area. In both topologies, the base station is located in the center of the area. The generated topologies can be downloaded from [1].

3.2 Radio Propagation

Cooja provides the free space radio propagation model, which calculates the received power $P_r(d)$ at a distance d from a transmitter using the Friis equation. Castalia, MiXiM and WSNet implement the log-normal shadowing model, which can be degraded to the free space model. The received power $P_r(d)$ at a distance d from a transmitter is calculated using the formula:

$$P_r(d) = P_r(d_0) - 10 \cdot n \cdot log_{10}(\frac{d}{d_0}) + X_\delta \qquad (1)$$

where $P_r(d_0)$ is an average power at a reference distance d_0, n is a path loss exponent, and X_δ is a normally distributed random variable with mean 0 and deviation δ. When the path loss exponent and random variable is set to 2 and 0, respectively, the model corresponds to the free space radio propagation model.

Cooja, MiXiM and WSNet calculate the path loss at a reference distance using the Friis equation. They do not provide the possibility to set $P_r(d_0)$ in a configuration file. On the contrary, Castalia does this. We set the value to 40.337 dBm, which we calculated using the Friis equation. Cooja allows to set a wavelength instead of a carrier frequency. We set it to 0.1209 m, which corresponds to the carrier frequency of 2480 MHz. Both Castalia and MiXiM have the configurable threshold on a signal delivery between nodes. We set the value according to the documentation of Castalia, i.e., to −100 dBm. The most important parameters of the radio propagation model are listed in Table 1. The full list of parameters and their set values is available at [1].

3.3 PHY and MAC Layer

The IEEE 802.15.4 is a standard that specifies PHY and MAC layers for low-rate wireless personal area networks, which operate at 800, 900 and 2400 MHz bands [9].

Texas Instruments CC2420 is a 2.4 GHz radio compliant with the 802.15.4 standard [21]. Castalia, MiXiM directly support the CC2420 radio. WSNet provides the 802.15.4 radio model with the possibility to select between 800 MHz, 900 MHz and 2400 MHz bands. We chose the last option as it corresponded to our needs.

A CC2420 radio uses O-QPSK modulation. The packet reception rate highly depends on the modulation. Hence, we decided to check whether the considered simulators implement the right modulation. WSNet implements O-QPSK modulation within its 2.4 GHz 802.15.4 radio model. In Castalia, CC2420 radio uses PSK, which essentially is BPSK modulation. In MiXiM, the modulation is handled by a dedicated decider module. The 802.15.4 compliant decider supports MSK modulation. According to [15], O-QPSK, BPSK and MSK have the same performance, which means that Castalia, MiXiM and WSNet should have the same bit-error-rates (BER) for a given signal-to-noise-plus-interferences ratio (SINR). In Cooja, modulation is realized in Contiki network stack or more specifically, in some of the radio drivers. Although there are drivers for the CC2420 radio, we cannot use them in our level of abstraction unless we write corresponding "glue drivers" to connect to Cooja. Instead, we used the default and only possible option of *cooja-radio*, where there is no modulation.

Table 1: Simulation settings.

Radio propagation	Path loss exponent	2
	Deviation	0 dBm
	Reference distance	1 m
	Carrier frequency	2480 MHz
Physical layer	Radio channel	26
	Transmission power	−25 dBm
	Sensitivity	−95 dBm
	Background noise	−110 dBm
MAC layer	Backoff method	exponential
	Max number of backoffs	5

At the MAC layer, the 802.15.4 standard specifies two protocols – slotted and unslotted CSMA-CA. The slotted CSMA-CA is used for beacon-enabled networks, while the unslotted one – for non beacon enabled networks. For the purpose of this work, we decided to use unslotted CSMA-CA. MiXiM and WSNet implement it. Although Castalia implements the CSMA-CA unslotted protocol, it works only with beacon enabled networks [3]. Hence, we used Tunable MAC, which can be set so it behaves like a simple CSMA-CA MAC protocol with one exception – it does not allow to set the maximum number of CSMA backoffs. In Cooja, we selected for the CSMA implementation that does not consider slots.

According to the 802.15.4 standard, the clear channel assessment (CCA) can be done in three modes [9]. In the first mode, CCA reports a busy medium if it detects any energy above the threshold. In the second mode, a medium is considered busy if a signal compliant with this standard (with the same modulation and spreading characteristics) is detected. The signal may be below or about the threshold. In the third mode, CCA reports a busy medium if it detects a signal compliant with this standard and its energy is about the threshold. Castalia and cooja-radio implement the first, MiXiM the second and WSNet the first and the third modes. In order to make simulations in WSNet, Castalia and Cooja comparable to simulations in MiXiM, we set threshold to −95 dBm, which is equal to the radio sensitivity (see Table 1). The background noise was set to −110 dBm, so only signal compliant with the 802.15.4 standard could be detected above the −95 dBm threshold.

Unfortunately in Cooja, as we discovered later, noise from surrounding radios is considered only for nodes simulated on the network simulation level of abstraction. Therefore, cooja-radio always assumes the channel to be clear.

The most important parameters of the PHY and MAC layers are listed in Table 1. The full list of parameters and their set values is available at [1].

3.4 Energy Consumption

Castalia, MiXiM and WSNet model energy consumption as the battery linear depletion. Unfortunately, in our settings, Cooja does not provide information about battery

charge consumption. Castalia and MiXiM allow to specify values for energy consumptions of a radio chip in different modes (receiving, transmitting and sleeping). Castalia also models energy consumption of the transitions between any of these radio modes and energy consumption of a microcontroller. MiXiM models energy consumption of only two transitions – between receiving and transmitting modes. WSNet allows to specify the energy consumption of a node only in transmitting and receiving modes.

We set the energy consumption of a node in the receiving and transmitting modes to 62.04 mJ and 28.05 mJ, respectively. The values correspond to the documentation of CC2420 radio chip [21], when the supply voltage is 3.3 V and transmission power is −25 dBm. In MiXiM and Castalia, the parameters, which specify the energy consumption of a node in other modes, were set to zero in order to make the simulation results comparable between each other.

4. OUR TEST CASE

For the purpose of this paper, we designed and implemented application, network protocol, attackers and IDS. Their description follows.

4.1 Application

Every node sends a packet every 1000 ms. The packet travels through the static multi-hop path until it reaches the destination – the base station. The application runs for 5 minutes, i.e., a leaf node sends 300 packets. In order to avoid collisions due to nodes synchronization, a node starts transmitting randomly within 1000 ms intervals. The total size of a packet is 115 B.

4.2 Network Layer

The network layer uses a static tree-based packet routing. When a node receives a packet, it forwards the packet to its parent node according to the routing tree. The base station is the root of the tree.

We generated the tree for the dense topology in WSNet. The routing tree construction is initiated by the base station, which broadcasts a packet containing its identification and distance to the sink (set to 0). When a node X receives a packet P from its neighbor A, it sets A as its parent and updates its distance to the sink if the following conditions hold: A is close enough (up to 6 m in the dense topology) and routing via this node will improve current distance to the sink. The limitation of the maximum neighbor distance (6 m) allowed us to create enough hops even in a dense network. If X sets a (new) parent, it waits randomly at most 1 s and broadcasts a packet containing its identification and distance to the base station. The algorithm is outlined below.

Algorithm 1 Routing tree construction

```
if getSinkDist() > getDist(X, A) + getSinkDist(P) then
  if getDist(X, A) < 6 then
    setParentID(A)
    setSinkDist(getDist(X, A) + getSinkDist(P))
    waitAtMost(1000)
    broadcastChange(X, getSinkDist())
  end if
end if
```

We consequently used the generated routing tree in all the simulators. For the sparse topology, we used the same routing tree as for the dense topology. To eliminate implementation errors, we simulated the application, recorded packet paths and compared them between the simulators. Since we did not observe any differences, we concluded that the network layer worked the same in all four simulators.

4.3 Attackers

We considered two kinds of attacker – selective forwarder and hello flooder. Both attacks may have a high impact on the network performance.

The *selective forwarder* refuses to forward packets coming from the neighbors and simply drops them [10]. We set the probability of a packet being dropped to 50%. In order to perform this attack, a malicious node should be a part of the routing tree. We placed 5 attackers at sensor nodes with different forwarding characteristics. We did not consider leaf nodes since they do not forward any packets and the attacker is not motivated to capture these nodes.

The *hello flooder* broadcasts hello packets using a more powerful transceiver than a general sensor node does [10]. There was only one hello flooder in our simulations. We placed it at a node newly added to the network and set its transmission power to 20 dBm. The transmission power remained constant during the whole experiment. The flooder transmitted packets with the same frequency as legitimate nodes.

4.4 Intrusion Detection System

The designed and implemented IDS is simple and quite naïve since our goal is not to design and present an effective IDS but to demonstrate the impact of the differences in simulators on the evaluation of an IDS.

We assume that one instance of an IDS is running on every sensor node and it continuously analyzes all sent and overheard packets. An IDS stores two tables, one for each type of attack. In both tables, each row corresponds to a certain monitored node. The tables contain different statistics about monitored nodes. The number of rows in both tables is limited. When the dense network is simulated, a number of monitored nodes is limited to 20. When a sparse network is simulated, this number is set to 5.

The first table used to detect the selective forwarding attack contains statistics about the number of packets received (PR) and forwarded (PF) by a monitored node. The detection exploits the fact that a monitoring node overhears to some extent both incoming and outgoing packets of a close enough monitored neighbor. If the IDS on a node X overhears a packet P sent by a node A to a node B, B is close enough and B should forward the packet (e.g., B is not a base station), then the IDS stores P in the buffer and increments PR of the monitored node B. Afterwards, when the IDS overhears the packet P being forwarded by the node B, it removes P from the buffer and increments PF of the node B. Since both the table and the buffer are limited, the IDS continuously monitors only the closest nodes and the newest packets. The algorithm is outlined in Algorithm 2.

The second table used to detect the hello flood attack contains statistics about the maximum signal strength (RSS_{max}) of packets received from a monitored node. The detection exploits the fact that a close enough neighbor of a hello flooder receives its packets with suspiciously high sig-

Algorithm 2 Statistics collection for detection of selective forwarding attack

```
if isNodeInTable_1(A) ∧ isPacketInBuffer(P) then
  removePacketFromBuffer(P)
  incrementPFInTable_1(A)
end if
if B ≠ X ∧ ¬isBaseStation(B) then
  if ¬isNodeInTable_1(B) then
    if ¬isFreeSlotInTable_1() then
      C = getFarthestNodeFromTable_1()
      if getDist(X, B) ≥ getDist(X, C) then
        return
      end if
      removeNodeFromTable_1(C)
      removePacketsFromBuffer(C)
    end if
    addNodeToTable_1(B)
  end if
  if ¬isFreeSlotInBuffer() then
    removeOldestPacketFromBuffer()
  end if
  addPacketToBuffer(P)
  incrementPRInTable_1(B)
end if
```

nal strength. If an IDS on a node X overhears a packet P from a node A with the signal strength stronger than all the previously observed signal strengths from the same node, then it updates RSS_{max} of the monitored node A. Again, the table of monitored nodes is limited, therefore the IDS continuously monitors only nodes with the highest RSS_{max}. The algorithm is outlined in Algorithm 3.

Algorithm 3 Statistics collection for detection of selective forwarding attack

```
if A ≠ X then
  if ¬isNodeInTable_2(A) then
    if ¬isFreeSlotInTable_2() then
      C = getWeakestNodeFromTable_2()
      if getRSS(P) ≤ getRSS_maxFromTable_2(C)
      then
        return
      end if
      removeNodeFromTable_2(C)
    end if
    addNodeToTable_2(A)
  end if
  if getRSS(P) > getRSS_maxFromTable_2(A) then
    setRSS_maxInTable_2(A, getRSS(P))
  end if
end if
```

The detection is done at the end of the simulation based on the collected statistics.

The node X considers the node A as a selective forwarder if the dropping ratio of A, i.e., ratio of a number of packets dropped to a number of packet received, is higher than a predefined threshold.

The node X considers the node A as a hello flooder if the node X overhears a packet from A with signal strength higher than a predefined threshold.

In Section 5, we present the detection results for different threshold values.

5. SIMULATION RESULTS

We simulated 4 scenarios: 1) *dense network with five selective forwarders*; 2) *dense network with one hello flooder*; 3) *sparse network with five selective forwarders*; 4) *sparse network with one hello flooder*.

The simulations were terminating simulations, ending after 5 minutes. We evaluated the implemented IDS in terms of a number of false positives, a number of true positives and the energy consumed by nodes (see Subsection 5.1). Simulations were run only once per each scenario, which we believe was enough due to several reasons. Some results presented in this section do not depend on the number of repetitions (BER versus SINR curves, signal strength versus distance). Other results (number of false positives, number of true positives, number of neighbors, energy consumed by a network) depend on the number of packets overheard by each node. According to our results (see Figure 5), each node overheard more than $15'000$ packets on average. Hence, we can expect that any randomness involved in a packet reception is averaged for such number of packets.

5.1 Evaluation Metrics

We used the following metrics to evaluate the effectiveness of an IDS [19]:

- *A number of false positives.* A false positive occurs when a node X considers a node A ($X \neq A$) as a malicious while it is not.
- *A number of true positives.* A true positive occurs when a node X correctly considers a node A ($X \neq A$) as a malicious.
- *The energy consumed by nodes.*

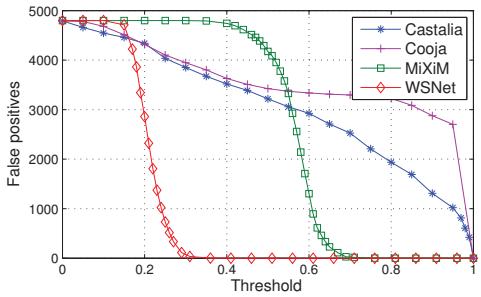
In order to calculate a number of false negatives for a malicious node, it is necessary to assign a set of nodes, which we think should detect the malicious node. For example, the set may contain only direct neighbors of a malicious node. However, as we demonstrate in Section 6, the number of neighbors might differ between simulators, hence making this metric incomparable. Another possibility is to say that all legitimate nodes should detect all malicious nodes in the network. However, this approach will result into a large and not intuitive numbers. Due to the reasons mentioned above, we did not measure the number of false negatives.

5.2 Selective Forwarding Attack

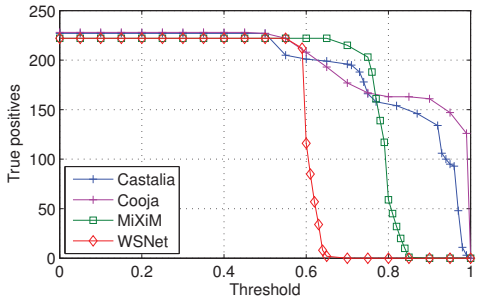
The relationship between the number of false positives (true positives) and threshold is depicted in Figure 1[1]. The threshold here determines the maximum dropping ratio allowed for a node. If the dropping ratio of a node is higher than the threshold, the node is considered as a selective forwarder.

The simulation results differ significantly. The implemented IDS achieves the best results in WSNet. This can be explained by the fact that in WSNet, a packet reception is more reliable that in other simulators. Hence, an IDS overhears more packets and performs better. For more

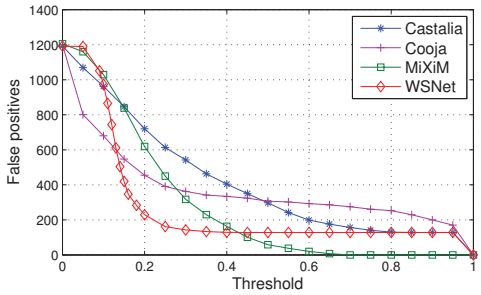
[1]For the clarity of individual graphs, the density of data points was reduced approx. by the factor of 5.



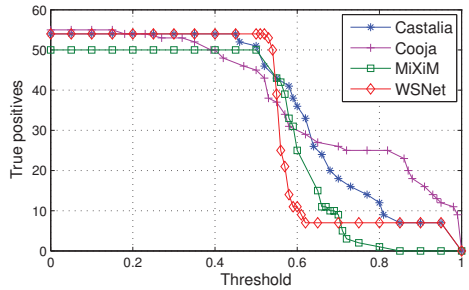
(a) Dense topology, false positives



(b) Dense topology, true positives

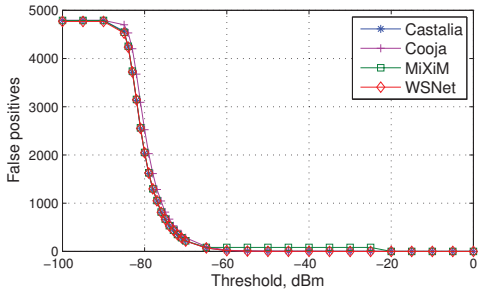


(c) Sparse topology, false positives

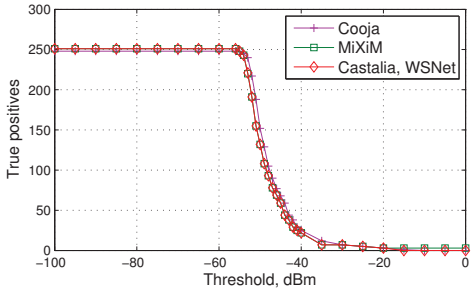


(d) Sparse topology, true positives

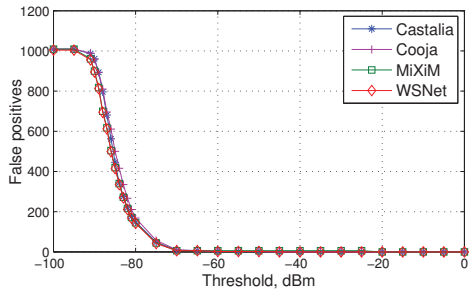
Figure 1: Selective forwarding attack.



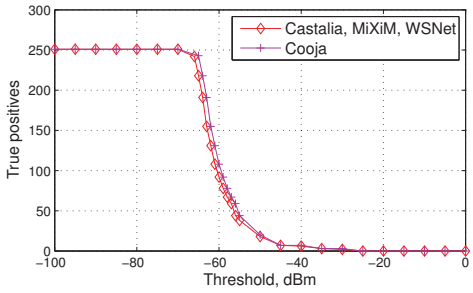
(a) Dense topology, false positives



(b) Dense topology, true positives



(c) Sparse topology, false positives



(d) Sparse topology, true positives

Figure 2: Hello flood attack.

information on the reliability of a packet reception, see Figure 7 and the accompanying description.

We observed that the lists of monitored nodes for a given monitoring node were very similar between the simulators. In all simulation scenarios, almost every node monitored the predefined maximum number of neighbors (5 in the sparse and 20 in the dense network, see Subsection 4.4). This explains the fact that the maximum achievable number of true (false) positives is four times smaller in the sparse network than in the dense one.

The number of false positives is smaller in the sparse network than in the dense network due to interferences. There are more interferences in the dense network and hence an IDS overhears a less number of forwarded packets.

The number of true and false positives in the sparse network is lower bounded for all the simulators except MiXiM (see Figure 1c). In this topology, there is a small number of nodes (mostly leaves) that do not have enough non-leaf neighbors. An IDS running on such a node overhears packets intended for nodes, which are outside of its radio range. These packets are always considered dropped since the IDS cannot hear, that the packets are being forwarded. In MiXiM, the radio range is higher and IDS is able to intercept significantly higher amount of packets, therefore there are no such nodes (see Subsection 6.2).

5.3 Hello Flood Attack

The relationship between the number of false positives (true positives) and threshold is depicted in Figure 2[2]. The threshold here determines the maximum transmission power allowed for a node. If the received signal strength from a node is higher than the threshold, the node is considered as a hello flooder.

The simulation results are almost the same. The technique for the detection of the hello flood attack is based on the received signal strength (see Subsection 4.4). The signal strength for any distance is the same in all of the considered simulators (see Figure 8). In addition, the sets of pairs (monitoring node, monitored node) are very similar between the simulators since an IDS monitors only nodes with the highest signal strength and takes into account only the maximal signal strength received from a monitored node. These properties of the detection technique reduce the effect of different radio ranges and different numbers of interferences.

The only notable difference is that some of the simulators use optimization and neglect path loss for small distances. The received signal strength is equal to the transmission power of a node if a distance is up to 1 m and 0.1 m in MiXiM and Castalia, respectively. This causes a different, yet negligible number of false positives for the high values of the threshold.

5.4 Energy Consumption

The energy consumed by a network in different scenarios is presented in Table 2. While the results are consistent for Castalia and MiXiM, we obtained significantly different results with WSNet. In MiXiM and Castalia, a radio was always in the receiving mode if it was not transmitting at the moment. In WSNet, a radio was in the receiving (transmitting) mode only when the node was receiving (transmitting) a packet. The differences between Castalia, MiXiM and WSNet are smaller in the dense network since a node overhears more packets and hence it spends more time in the receiving mode.

[2]For the clarity of individual graphs, the density of data points was reduced approx. by the factor of 5.

Table 2: Energy consumption (in Joules)

		Dense topology	Sparse topology
Selective forwarding attack	Castalia	4660	4648
	MiXiM	4657	4644
	WSNet	2781	465
Hello flood attack	Castalia	4659	4629
	MiXiM	4658	4652
	WSNet	2804	497

6. FURTHER INVESTIGATION

In order to reveal the sources of the differences (presented in the previous section), we performed a deeper analysis of the considered simulators. We thoroughly examined network, MAC and PHY layers of the simulators by reviewing additional metrics at these layers. We did this only for the scenario where we observed the most significant differences, i.e., with dense network and five selective forwarders.

6.1 Network Layer

At the network layer we measured: 1) *number of packets received by a node*; 2) *number of packets sent by a node*.

At this layer, the number of sent packets involves all the packets that a node is intended to send. Not all the packets are actually sent at the PHY layer. Some packets are dropped at the MAC layer (e.g., a buffer is full or a maximum number of backoffs is exceeded) or at the PHY layer (e.g., a radio is busy). For a legitimate node, the number of sent packets is equal to the number of received (forwarded) packets plus 300 – the number of packets created by the node itself. Since the numbers of received and sent packets are highly correlated, we present only the histogram of the number of packets sent by each node (see Figure 3). As it can be seen from the histogram, the distribution of the number of sent packets is very similar for all simulators. This can be explained by the fact that we used the same static routing tree in all of them. However, in WSNet the average number of sent packets is significantly higher than in others. This is probably caused by the differences on lower layers that will be discussed in the following subsections.

6.2 MAC Layer

At this layer, we measured: 1) *number of neighbors*; 2) *number of packets overheard by an IDS*.

We consider a node A as a neighbor of a node X ($X \neq A$) if the node X overheard at least one packet from the node A during the experiment. The histogram of a number of node neighbors is depicted in Figure 4. While Castalia, Cooja and WSNet have generally consistent results, MiXiM significantly differs – any sensor node is a neighbor of almost any other node in the network. Although we set the radio sensitivity to -95 dBm in all simulators (see Subsection 3.3), sensor nodes keep to receive packets below this value in MiXiM. MiXiM ignores this setting and the decision whether a node receives a packet is only based on the SINR.

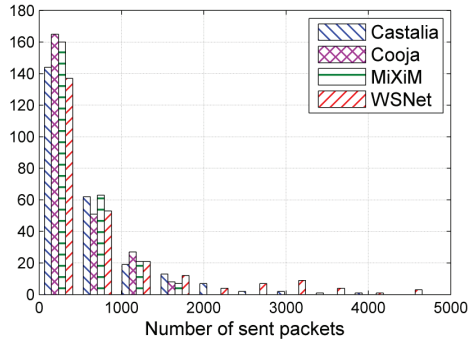


Figure 3: Number of packets sent by every node.

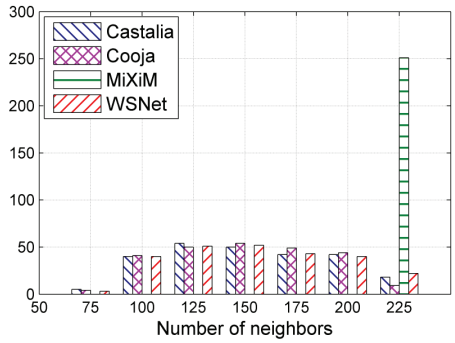


Figure 4: Number of neighbors.

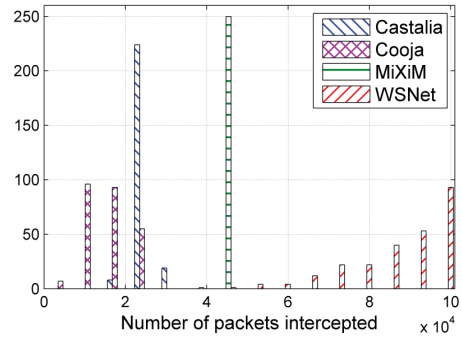


Figure 5: Number of packets overheard by IDSs.

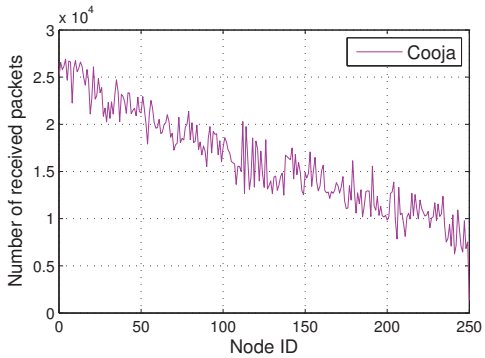


Figure 6: Cooja – packets overheard versus node ID.

In all scenarios, we set the background noise to −110 dBm (see Subsection 3.3). In MiXiM, this results into the 100% reception ratio of packets with the received signal strength higher than −100 dBm (see Figure 7 when SINR is equal to 10).

We differentiate between packets not addressed to the IDS and packets created or forwarded by the IDS. Packets addressed to the IDS are not considered in this scenario since the IDS does not monitor itself when it tries to detect a selective forwarding attack (see Subsection 4.4). The histogram of a number of packets overheard by IDSs is depicted in Figure 5. The results are significantly different and this can be caused by a number of factors, including the differences on other layers.

In Cooja, we observed the correlation between a node identification and the number of packets not addressed to but overheard by the IDS running on the node (see Figure 6). Consider a scenario, where a node A sends a packet to a node B which should be overheard by a node C. The node B forwards this packet. The memory of the node C is rewritten before the first packet is received and the node C receives only the packet forwarded by the node B. Although this does not have any impact on the sink application, the number of packets overheard by an IDS is drastically reduced.

Castalia, Cooja and MiXiM have a buffer for outgoing packets and it is limited to a certain number. In contrast, there is no such limit for a buffer size in WSNet. Hence, in WSNet, none of the packets are dropped because of the full buffer. This might be a source of the higher number of forwarded and hence overheard packets in WSNet.

6.3 Physical Layer

At this layer, we measured: 1) *received signal strength at different distances*; 2) *BER for a different SINR*.

As it was already mentioned in Subsection 5.1, MiXiM and Castalia optimize the path loss computation and do not consider distances smaller than the predefined value. This optimization does not significantly impact the evaluation results.

In WSNet, we observed that a sensor node did not receive a packet if its received signal strength was below −85 dBm. According to the documentation of WSNet, the default value for sensitivity of 2.4 GHz 802.15.4 radio is −92 dBm. The analysis of source code revealed that the minimal possible sensitivity was equal to −85 dBm. That is why a sensor node did not receive packets if their received signal strength was below −85 dBm. This issue has a high impact on the evaluation results. Hence, we fixed it before we started with our simulations. After that the curves representing the relationship between the signal strength and distance are almost the same for all simulators. Since the differences are negligible, we provide only one curve in Figure 8.

As it was already mentioned in Subsection 3.3, MSK, BPSK and O-QPSK modulations have the same performance. Although Castalia, MiXiM and WSNet should have the same BER versus SINR curves, they are not the same (see Figure 7). BER in WSNet is lower than in other simulators for SINR ranging between 0 and 10. This results into a more reliable packet reception than in other simulators, which we encountered in our simulations. The source code analysis revealed that the developers of the simulators implemented different formulas for the calculation of BER based on SINR, although they should be equal for the same radio chip.

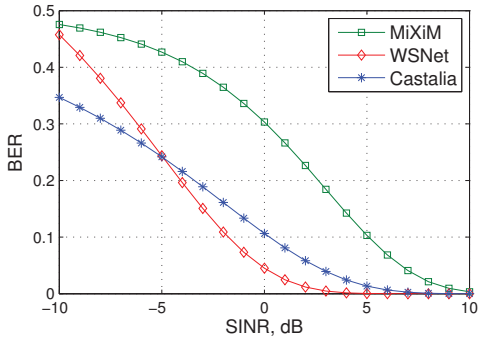


Figure 7: BER versus SINR.

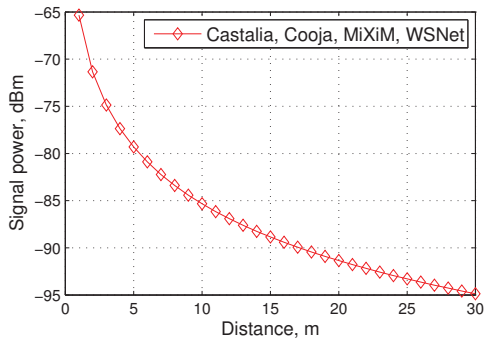


Figure 8: Signal strength versus distance.

In Cooja, a node successfully receives a packet if SINR is greater or equal to a predefined value. However, if the receiver sensitivity is greater than background noise and difference between receiver sensitivity and background noise is greater than the threshold, this threshold gets reset to the difference. Thus, manipulating these values, one can successfully transmit a packet with an arbitrary SINR. With our settings, successful communication requires SINR ≥ 15.

7. CONCLUSIONS

This work compared the evaluation results of the same intrusion detection system on Castalia, Cooja, MiXiM and WSNet simulators. In order to make the results comparable, we aimed to use the same energy consumption, radio propagation, physical and medium access control models in the simulators and set these models in the same way. Also, we implemented the static tree-based routing protocol, the application that constantly sent packets to a base station and two kinds of attacker (selective forwarder and hello flooder), again for four selected simulators.

The simulations revealed that the results from different simulators significantly differed from each other even though the simulators were set in the same way. These differences were caused by mistakes in the implementations and badly or not at all documented parameters hidden in the source codes. We observed the major differences at physical (e.g., a BER / SINR curve, sensitivity) and medium access control (e.g., buffer size, number of backoffs) layers, which resulted into different packet reception probabilities, different number of interferences, different number of packets overheard by intrusion detection system and in some cases a different number of false positives and true positives.

In order to make results more realistic and more comparable across different simulators, we suggest to constantly compare the simulators between each other, at least those parts that should be the same (e.g., log normal shadowing model, O-QPSK modulation). We assume this should be done mainly by the developers of simulators as it is a relatively fast and cheap way to make their products free of bugs. If the problem remains, researchers should rather implement all protocols they want to compare in the same simulator in order to avoid an incorrect comparison result and consequently misleading conclusion.

8. ACKNOWLEDGMENTS

We are grateful to J. Kůr, L. Smolík, M. Stehlík, P. Švenda for the discussions and suggestions that improved the paper. This work was supported by the GAP202/11/0422 project. A. Stetsko and V. Matyáš were additionally supported by the Centre of Excellence GAP202/12/G061 of the Czech Science Foundation.

9. REFERENCES

[1] `http://www.fi.muni.cz/~xstetsko/simutools/`.

[2] L. Bergamini, C. Crociani, A. Vitaletti, and M. Nati. Validation of wsn simulators through a comparison with a real testbed. In *ACM workshop on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks*, PE-WASUN '10, pages 103–104, New York, NY, USA, 2010. ACM.

[3] A. Boulis. Castalia User's Manual. `http://castalia.npc.nicta.com.au/pdfs/Castalia-UserManual.pdf`, 2011.

[4] G. Chelius, A. Fraboulet, and E. Fleury. Worldsens: a fast and accurate development framework for sensor network applications. In *ACM symposium on Applied computing*, SAC '07, pages 222–226, New York, NY, USA, 2007. ACM.

[5] U. M. Colesanti, C. Crociani, and A. Vitaletti. On the accuracy of omnet++ in the wireless sensornetworks domain: simulation vs. testbed. In *ACM workshop on Performance evaluation of wireless ad hoc, sensor,and ubiquitous networks*, PE-WASUN '07, pages 25–31, New York, NY, USA, 2007. ACM.

[6] W. Du, D. Navarro, F. Mieyeville, and F. Gaffiot. Towards a taxonomy of simulation tools for wireless sensor networks. In *ICST Conference on Simulation Tools and Techniques*, SIMUTools '10, ICST, Brussels, Belgium, 2010. ICST.

[7] A. Dunkels. Rime – a lightweight layered communication stack for sensor networks. In *European Conference on Wireless Sensor Networks, Poster/Demo session*, 2007.

[8] A. Dunkels, B. Gronvall, and T. Voigt. Contiki – a lightweight and flexible operating system for tiny networked sensors. In *IEEE International Conference on Local Computer Networks*, pages 455–462, Los Alamitos, CA, USA, 2004. IEEE Computer Society.

[9] IEEE Std 802.15.4-2006. *IEEE Standard for Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements. Part 15.4: Wireless Medium Access Control (MAC)*

and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs). IEEE, New York, NY, USA, 2006.

[10] C. Karlof and D. Wagner. Secure routing in wireless sensor networks: attacks and countermeasures. In *IEEE International Workshop on Sensor Network Protocols and Applications*, pages 113–127, 2003.

[11] A. Köpke, M. Swigulski, K. Wessel, D. Willkomm, P. T. K. Haneveld, T. E. V. Parker, O. W. Visser, H. S. Lichte, and S. Valentin. Simulating wireless and mobile networks in omnet++ the mixim vision. In *Conference on Simulation tools and techniques for communications, networks and systems & workshops*, Simutools '08, pages 71:1–71:8, ICST, Brussels, Belgium, 2008. ICST.

[12] M. Korkalainen, M. Sallinen, N. Kärkkäinen, and P. Tukeva. Survey of wireless sensor networks simulation tools for demanding applications. In *Proceedings of the 2009 Fifth International Conference on Networking and Services*, ICNS '09, pages 102–106, Washington, DC, USA, 2009. IEEE Computer Society.

[13] Q. Li, F. Österlind, T. Voigt, S. Fischer, and D. Pfisterer. Making wireless sensor network simulators cooperate. In *ACM workshop on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks*, PE-WASUN '10, pages 95–98, New York, NY, USA, 2010. ACM.

[14] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt. Cross-level sensor network simulation with cooja. In *IEEE Conference on Local Computer Networks*, pages 641–648, 2006.

[15] S. Pasupathy. Minimum shift keying: A spectrally efficient modulation. *IEEE Communications Magazine*, 17(4):14–22, july 1979.

[16] D. Pediaditakis, Y. Tselishchev, and A. Boulis. Performance and scalability evaluation of the castalia wireless sensor network simulator. In *ICST Conference on Simulation Tools and Techniques*, SIMUTools '10, ICST, Brussels, Belgium, 2010. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[17] H. N. Pham, D. Pediaditakis, and A. Boulis. From simulation to real deployments in WSN and back. In *IEEE Symposium on World of Wireless, Mobile and Multimedia Networks*, WoWMoM '07, pages 1–6. IEEE, 2007.

[18] E. M. Shakshuki, H. Malik, and T. R. Sheltami. Lessons learned: Simulation vs WSN deployment. In *Conference on Advanced Information Networking and Applications*, volume 0, pages 580–587, Los Alamitos, CA, USA, 2009. IEEE Computer Society.

[19] A. Stetsko and V. Matyas. Effectiveness metrics for intrusion detection in wireless sensor networks. In *EC2ND 2009 – European Conference on Computer Network Defense*, pages 21–28, Los Alamitos, CA, USA, 2009. IEEE Computer Society.

[20] A. Stetsko, M. Stehlik, and V. Matyas. Calibrating and comparing simulators for wireless sensor networks. In *2011 IEEE 8th International Conference on Mobile Adhoc and Sensor Systems (MASS)*, pages 733–738. IEEE, 2011.

[21] Texas Instruments. Chipcon CC2420 Datasheet. `http://focus.ti.com/lit/ds/symlink/cc2420.pdf`, 2007.

[22] T. Voigt, J. Eriksson, F. Österlind, R. Sauter, N. Aschenbruck, P. J. Marrón, V. Reynolds, L. Shu, O. Visser, A. Koubaa, and A. Köpke. Towards comparable simulations of cooperating objects and wireless sensor networks. In *ICST Conference on Performance Evaluation Methodologies and Tools*, VALUETOOLS '09, Brussels, Belgium, 2009. ICST.

[23] E. Weingartner, H. vom Lehn, and K. Wehrle. A performance comparison of recent network simulators. In *IEEE Conference on Communications*, ICC '09, pages 1–5. IEEE, 2009.