



HAL
open science

Modeling tools for detecting DoS attacks in WSNs

Paolo Ballarini, Lynda Mokdad, Quentin Monnet

► **To cite this version:**

Paolo Ballarini, Lynda Mokdad, Quentin Monnet. Modeling tools for detecting DoS attacks in WSNs. Security and communication networks, 2013, 6 (4), pp.420-436. 10.1002/sec . hal-01817483

HAL Id: hal-01817483

<https://hal.science/hal-01817483>

Submitted on 18 Jun 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

RESEARCH ARTICLE

Modeling tools for detecting DoS attacks in WSNs

Paolo Ballarini¹, Lynda Mokdad^{2*} and Quentin Monnet²¹Laboratoire MAS, École Centrale de Paris²Laboratoire LACL, Université Paris-Est, Créteil

ABSTRACT

Detecting Denial of Service (DoS) attacks and reducing the energy consumption are two important and frequent requirements in Wireless Sensor Networks (WSNs). In this paper, we propose an energy-preserving solution to detect compromised nodes in hierarchically clustered WSNs. DoS detection is based on using dedicated inspector nodes (cNodes) whose role is to analyze the traffic inside a cluster and to send warnings to the cluster-head whenever an abnormal behavior (i.e. high packets throughput) is detected. With previously introduced DoS detection schema cNodes are statically displaced in strategic positions within the network topology. This guarantees a good detection coverage but leads to quickly draining cNodes battery. In this paper we propose a dynamic cNodes displacement schema according to which cNodes are periodically elected among ordinary nodes of each atomic cluster. Such a solution results in a better energy balance while maintaining good detection coverage. We analyze the tradeoffs between *static* and *dynamic* solutions by means of two complementary approaches: through simulation with the NS2 simulation platform and by means of statistical model checking with the Hybrid Automata Stochastic Logic.

Copyright © 0000 John Wiley & Sons, Ltd.

KEYWORDS

DoS attacks; Detection method; Statistical model checking; Modeling tools

*Correspondence

Lynda Mokdad — UPEC Laboratoire LACL, 61 avenue du Général de Gaulle, 94010 Créteil, France.

E-mail: lynda.mokdad@u-pec.fr

Received . . .

INTRODUCTION

Detecting phenomena such as forest fires or seismic activities implies to keep watch over wide areas. Wireless Sensor Networks (WSNs) are often used so as to achieve this watch. The sensors that make up those WSNs are devices able to perform measurements on their surrounding environment, and to send the collected data to a Base Station (BS). Due to their small size, the sensors have very limited resources: memory, computing capability as well as available energy must be spent with care [GA09][BY10][BF12].

Other uses of WSNs include activities such as preventing chemical, biological or nuclear threats in an area, or collecting data on a military field [LB09] [CS11]. In such sensitive domains, the deployment of a WSN brings out strong requirements in terms of security. Various works deal with ways of preventing unauthorized access to data, or with the necessary precautions to guarantee data authenticity and integrity inside the network [MS11] [BBBP12] [BB12][YB10]. But confidentiality as well as

authentication are of poor use if the network is not even able to deliver its data correctly.

Denial of Service in WSNs. Denial of Service (DoS) attacks indeed aim at reducing, or even annihilating the network ability to achieve its ordinary tasks, or try to prevent a legitimate agent from using a service [HS05][DFHV10] [FH11]. Because of the limited resources of their nodes, WSNs tend to be rather vulnerable to DoS attacks. For instance, a compromised sensor node can be used in order to send corrupted data at a high rate, either to twist the results or to drain the node's energy faster.

The problem we deal with in this paper is the development and analysis of detection mechanisms which are efficient both in terms of detection (i.e. they guarantee a high rate of detection of flooding nodes) and in terms of energy (i.e. they guarantee a balanced energy consumption throughout the network).

Clustered WSNs. One way to save some battery power during communications may reside in the choice of the network architecture and of the protocol used to route the

data from a sensor to the BS [ATB05]. In a hierarchical WSN, the network is divided into several clusters. The partition is done according to a clustering algorithm such as LEACH [HW00] [OX09], HEEDS [YF04], or one based on ultra-metric properties [FL11]. In each cluster, a single common node is elected to be a Cluster Head (CH), responsible for directly collecting data from the other nodes in the cluster. Once enough data has been gathered, the CHs proceed to data aggregation [YL11]. Then they forward their results to the BS. CHs are the only nodes to communicate with the BS, either directly, through a long-range radio transmission, or by multi-hopping through other CHs. So as to preserve the nodes' energy as long as possible, the network re-clustering is repeated periodically, with different nodes being elected as CHs. Note that clustering is not limited to a "single-level" partition. We can also subdivide a cluster into several "sub-clusters". The CHs from those "sub-clusters" would then send their aggregated data to the CHs of their parent clusters.

DoS detection: from static to dynamic guarding policies. In a hierarchically organised WSN, a *control node* (cNode in the remainder) is a node which is chosen to analyze the traffic directed to the CH of the cluster it belongs to, and potentially detect any abnormal behavior. Therefore cNodes provide us with an efficient way to detect DoS attacks occurring in the network. Note that cNodes are only meant to detect DoS attacks, thus they do not perform any sensing, nor do they send any data (apart from attack detection alarms). cNodes-based detection was first presented in [LC08], but the authors do not mention any periodical (cNodes) re-election scheme. One can suppose that the renewal of the election occurs each time the clustering algorithm is repeated. In [GM12], we proposed a dynamic approach: cNodes are re-elected periodically (any node in a cluster may be chosen, except the CH) with the election period selected to be shorter than that between two network clusterings. Intuitively such dynamic approach (in comparison to that of [LC08]), leads to a more uniform energy consumption while preserving a good detection ability.

Our contribution. In this paper we address the problem of validating the above conjecture by means of modeling techniques. More specifically our contribution regards the following aspects:

- i. we present a characterization of Markov chains models for representing DoS detection mechanisms and detail relevant steady-state measures analytically (i.e. we give the expression for the probability of detection of attacks in the Markov chain model);
- ii. we present a number of numerical results obtained by simulation of DoS detection on WSN models by means of the network simulator NS2 [NS2]. In particular we simulate models of grid topology WSN including DoS (*static* and *dynamic*) detection policies;
- iii. we present formal models of the DoS detection mechanisms expressed in terms of Generalized Stochastic Petri Nets (GSPN). In combination to GSPN models we also present a number of performance and dependability properties formally expressed in terms of the the Hybrid Automata Stochastic Logic (HASL) [BDDHP11].

The structure of the paper is as follows: in Section 1 we give an overview of DoS attack detection for cluster-based WSNs. In Section 2 we detail the networking solution we want to model including LEACH clustering algorithm which we refer to. In Section 3 we describe the structure of Markov chain models for modelling an attacked network. In Section 4 we present simulation experiments obtained with the NS2 platform. In Section 5 we present the application of statistical model checking performance analysis to Petri Nets models of attacked WSNs. Finally we give some final remarks in the Conclusion.

1. RELATED WORKS

To deal with DoS attacks in wireless sensor networks, many research studies have been conducted.

In [LC08], the authors propose a system detection based on static election of a set of special nodes called "guarding nodes" which analyze the network traffic. When detecting abnormal traffic from a given node, "guarding nodes" identify it as a compromised node and they inform the cluster head of it. In this study, the authors show the benefit of their method by presenting numerical analysis of detection rate but they don't consider the energy of the elected node which dies very quickly.

Back in 2001, most works focused on making WSNs feasible and useful. But some people already involved themselves into security. For instance, *Perrig et al.* proposed SPINS (Security Protocols for Sensor Networks) in [AP01] to provide networks with two symmetric key-based security building blocks. The first block, called SNEP (Secure Network Encryption Protocol), provides data confidentiality, two-party data authentication and data freshness. The second block, called μ TESLA ("micro" version of the Timed, Efficient, Streaming, Loss-tolerant Authentication Protocol) assumes authenticated broadcast using one-way key chains constructed with secure hash functions. No mechanism was put forward to detect DoS attacks.

A sensor network may be recursively and periodically reclustered with an algorithm such as LEACH, as in our proposal. The resulting hierarchically clustered network often presents a good ability for distributing the energy consumption among the sensor nodes. But security concerns (other than DoS) also apply to those networks. In [LO07], *Oliveira et al.* propose to add security mechanisms *via* a revised version of LEACH protocol. SecLEACH uses random key pre-distribution as well as μ TESLA (authenticated broadcast) so as to

protect communications. But the authors do not mention any mechanism to fight DoS attacks.

Elements from game theory have been used in several studies to detect DoS attacks in WSNs. In [MM09], Mohi *et al.* propose another way to secure the LEACH protocol against selfish behaviors. With S-LEACH, the BS uses a global Intrusion Detection System (IDS) while LEACH CHs implement local IDSs. The interactions between nodes are modeled as a Bayesian game, that is, a game in which at least one player (here, the BS) has incomplete information about the other player(s) (in this case: whether the sensors have been compromised or not). Each node has a “reputation” score. Selfish nodes can cooperate (so as to avoid detection) or drop packets. The authors show that this game has two Bayesian Nash equilibriums which provide a way to detect selfish nodes, or to force them to cooperate to avoid detection.

The best way to detect for sure a DoS attack in a WSN is simply to run a detection mechanism on each single sensor. Of course, this solution is not feasible in a network with constraints. Instead of fitting out each sensor with such mechanism, Islam *et al.* propose in [HI09] to resort to heuristics in order to set a few nodes equipped with detection systems at critical spots in the network topology. This optimized placement enables distributed detection of DoS attacks as well as reducing costs and processing overheads, since the number of required detectors is minimized. But those few selected nodes are likely to run out of battery power much faster than normal nodes.

Sensors authentication and DoS detection in clustered networks may be assumed by a single architecture. In [MH07], Hsieh *et al.* present SecCBSN, an adaptive security design intended to Secure Cluster-Based Communication in Sensor Networks. Each node is equipped with a system that includes three modules. One is involved in the cluster head election, and responsible for remembering the decision which were made. Another module provides ciphered communication and secure authentication protocols between sensors. It uses the TESLA certificate to enable deployed sensors to authenticate new incoming nodes. It allows the creation of secure channels as well as broadcast authentication between neighboring sensors. The last security module is responsible for the detection of compromised nodes. When a node is suspected to harm the network, alarm protocols are used to warn the base station. The use of trust value evaluation then enables the setting and the propagation of black and white lists of sensors.

Some works examine the possibility to detect the compromising of nodes as soon as an opponent physically withdraw them from the network. In the method that Ho develops in [JH10], each node keeps watching on the presence of its neighbors. The Sequential Probability Radio Test (SPRT) is used to determinate a dynamic time threshold. When a node appears to be missing for a period longer than this threshold, it is considered to be dead or captured by an attacker. If this node is later redeployed in the network, it will immediately be considered as

compromised without having a chance to be harmful. Nothing is done, however, if an attacker manages to compromise the node without extracting the sensor from its environment.

In [SM10], Misra *et al.* propose a revised version of the OLSR protocol. This routing protocol called DLSR aims at detecting distributed denial of service (DDoS) attacks and at dropping malicious requests before they can saturate a server’s capacity to answer. To that end, the authors introduce two alert thresholds regarding this server’s service capacity. They also introduce the use of Learning Automata (LAs), automatic systems whose choice of next action depends on the result of its previous action. There is no indication in their work about the overhead or the energy load resulting from the use of the DLSR protocol.

Son *et al.* propose in [JS10] a novel broadcast authentication mechanism to cope with DoS attacks in sensor networks. This scheme uses an asymmetric distribution of keys between sensor nodes and the BS, and uses the Bloom filter as an authenticator, which efficiently compresses multiple authentication information. In this model, the BS or sink shares symmetric keys with each sensor node, and proves its knowledge of the information through multiple MAC values in its flooding messages. When the sink floods the network with control messages it constructs a Bloom filter as an authenticator for the message. When a sensor node receives a flooded control message, it generates their Bloom filter with its keys and in the same way the sink verifies message authentication.

Li and Batten expose in [LB09] their method to detect and to recover from Path-based DoS (PDoS) attacks in wireless sensor networks. They consider WSNs whose aim is to collect data and to store it into small databases. PDoS attacks may prevent legitimate communication, lead the sensors to battery exhaustion and corrupt the gathered data. So the authors introduce the use of Mobile Agents (MAs), which use hash function values, node IDs and traffic table to analyze the traffic and identify compromised sensors. Thus the MAs are able to detect PDoS attacks with ease and efficiency, and to reply to the attack by proceeding to a recovery process. There are three distinct recovery processes available, depending on the percentage of compromised nodes in the network. Note that the authors use the assumption that MAs can not be compromised.

2. DETECTION OF DOS ATTACKS

2.1. Wireless Sensor Networks

We focus on the problem of detecting denial of service (DoS) attacks in a WSN. We recall that a WSN consists of a finite set of sensors plus a fixed base station (BS). Traffic in a WSN (mainly) flows from sensor nodes towards the BS. Furthermore since WSN nodes have inherently little energy, memory and computing

capabilities, energy efficiency is paramount when it comes with mechanisms/protocols for WSN management. Also communications between sensors and the BS rely on wireless protocols. In the following we assume that the nodes' mobility is limited or null.

Our goal is to set an efficient method to detect compromised nodes which may try to corrupt data, or to saturate the network's capacity, by sending more data than it should. In this case, efficiency can be measured in two respects:

- the detection rate of the compromised node(s);
- the network's lifetime, as we want to spend as little energy as possible.

In order to achieve these goals we focus on the following techniques: hierarchical network clustering, and dynamical election of control nodes responsible for monitoring the traffic.

2.2. Hierarchical clustering

The class of WSNs we consider is that of hierarchically cluster-based networks. The set of sensors has been partitioned into several subsets called "clusters". Those clusters are themselves split into "sub-clusters". For a better clarity, we will call *l-clusters* the sets resulting from the first clustering of the global set, and *k-clusters* the subset issued from the splitting of any $(k-1)$ -cluster. The successive clusterings are carried out with the use of any existing clustering algorithm, such as LEACH [HW00] [OX09], HEEDS [YF04], algorithms based on ultra-metric properties [YL11], etc. Each cluster contains a single cluster head (CH), designated among the normal nodes. The CH is responsible for collecting data from the other nodes of the subset. To follow up our naming conventions, we will call *k-CHs* the CHs belonging to the *k*-clusters. The *k*-CHs send the data they gathered to their $(k-1)$ -CH, the "0-CH" being the base station. In that way, the *k*-CHs are the only nodes to emit send packets towards the $(k-1)$ -CHs. Normal nodes' transmissions do not have to reach the base station directly, which would often consume much more energy than communicating with a neighbor node.

LEACH functioning

LEACH is probably one of the easiest algorithm to apply to recluster the network. It is a dynamical clustering and routing algorithm. We use it for our simulations using NS2. It splits a set of nodes into several subsets, each containing a cluster head. This CH is the only node to assume the cost-expensive transmissions to the BS.

Here is the LEACH detailed processing. Let P be the average percentage of clusters we want to get from our network at an instant t . LEACH is composed of cycles made of $\frac{1}{P}$ rounds. Each round r is organized as follows:

1. Each node i computes the threshold $T(i)$:

$$T(i) = \begin{cases} \frac{P}{1 - P \cdot (r \bmod \frac{1}{P})} & \text{if } i \text{ has not been CH yet} \\ 0 & \text{if } i \text{ has already been CH} \end{cases}$$

Each node chooses a pseudo-random number $0 \leq x_i \leq 1$. If $x_i \leq T(i)$ then i designates itself as a CH for the current round. $T(i)$ is computed in such a way that every node becomes CH once in every cycle of $\frac{1}{P}$ rounds: we have $T(i) = 1$ when $r = \frac{1}{P} - 1$.

2. The self-designed CH inform the other nodes by broadcasting a message with the same transmitting power, using carrier sense multiple access (CSMA) MAC.
3. The other nodes choose to join the cluster associated to the CH whose signal they receive with most power. They message back the CH to inform it (with the CSMA MAC protocol again).
4. CHs compile a "transmission order" (time division multiple access, TDMA) for the nodes which joined their clusters. They inform each node at what time it is expected to send data to its CH.
5. CHs keeps listening for the results. Normal sensors get measures from their environment and send their data. When it is not their turn to send, they stay in sleep mode to save energy. Collisions between the transmissions of the nodes from different clusters are limited thanks to the use of code division multiple access (CDMA) protocol.
6. CHs aggregate, and possibly compress the gathered data and send it to the BS in a single transmission. This transmission may be direct, or multi-hopped if relayed by other CHs.
7. Steps 5) and 6) are repeated until the round ends.

It is possible to extend LEACH by adding the remaining energy of the nodes as a supplementary parameter for the computation of the $T(i)$ threshold [MH02].

Note that each node decides whether to self-designate itself as a CH or not. Its decision does not take into account the behavior of surrounding nodes. For this reason, we can possibly have, for a given round, a number of CHs very different from the selected percentage P . Also, all the elected CHs may be located in the same region of the network, leaving "uncovered" areas. In that case, one can only hope that the spatial repartition will be better during the next round.

k-LEACH

Once the LEACH algorithm has been applied to determine a first set of clusters, nothing prevents us to apply it again on each cluster. This is the way we got our *k*-clusters: we applied k times the LEACH algorithm recursively. We call those recursive iterations the *k-LEACH* algorithm. In practice, we had k equal to 2, for the following reasons:

- so as to save more energy than what we would do with 1-LEACH;

- so as to have a finer clustering of the network, in order to elect control nodes in each of the 2-clusters, to maximize the cover area and the probability to detect compromised nodes.

Other algorithms

Other possible clustering algorithms include HEED [YF04], which is designed to save more energy than standard LEACH, and could lead to a better spatial repartition of the CHs inside the network. But in our network, all the sensors have the same initial available energy, and every one of them is able to directly reach the BS if need be. Under those assumptions, LEACH may not consume more energy than HEED protocol, and remains easier to use.

2.3. Attacks detection through cNodes

Along with normal nodes and cluster heads, a third type of node is present in the lower *k-clusters* of the hierarchy.

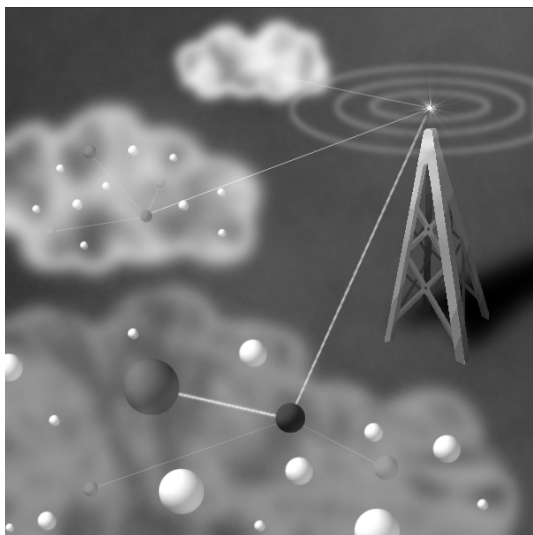


Figure 1. Cluster-based sensor network with cNodes

The *cNodes* — for *control nodes* — were introduced in [LC08] to analyze the network traffic and to detect any abnormal behavior from other nodes in the cluster. We refer the reader to [LC08] for a detailed description of the cNodes based detection mechanism. In brief, cNodes analyze the input traffic for the 2-CH of their 2-cluster, and watch out for abnormal traffic flows. Detection takes place whenever a cNode observes that at least one amongst the sensor nodes under its controlled perimeter sends data at a rate that is not within “regular behavior” thresholds. In that case the cNode sends a warning message to the CH. Once the CH has received warnings from a sufficiently large number of distinct cNodes (note that in order to prevent a compromised cNode to declare legitimate nodes as compromised the detection protocol requires that the

CH receives warnings by a minimum number of distinct cNodes before actually recognising the signalling as an actual anomaly), it starts ignoring the packets coming from the detected compromised sensor. cNodes may also monitor output traffic of the CHs and warn the BS if they come to detect a compromised CH.

cNodes are periodically elected among normal sensors. The guarding functionality of cNodes may lead to an energy consumption higher than that of “normal” (i.e. sensing) nodes. In order to maximize the repartition of the energy load, we propose a scheme by which a new set of cNodes is periodically established with an election period shorter than the length of a LEACH round (that is, the period between two consecutive CH elections). We propose three possible methods for the election process: self-election as for the CHs, election processed by the CHs and election processed by the BS.

Distributed self-election

A first possibility to elect the cNodes is to reuse the distributed self-designation algorithm defined for the election of the CHs. With this method, each non-CH node chooses a pseudo-random number comprised between 0 and 1. If this number is lower than the average percentage of cNodes in the network that was fixed by the user, then the node designates itself as a cNode. Otherwise, it remains a normal sensor.

This method has two drawbacks. Firstly, each node has to compute a pseudo-random number, which may not be necessary with other methods. Secondly, each node chooses to designate (or not) itself, without taking into account at any moment the behavior of its neighbors. As a result, the election is proceed with no consideration for the clustering that has been realized in the network. Indeed it is unlikely that the set of elected cNodes will be uniformly distributed among the 2-clusters that were formed, and it is even possible to end up with some 2-clusters containing no cNodes (thus being completely unprotected against attacks).

A possible workaround for this second drawback could be a two-steps election: in a first round nodes self-designate (or not) themselves. Then they signal their state to the 2-CHs they are associated to. In the second round, the 2-CHs may decide to designate some additional cNodes if the current number of elected nodes in the cluster is below a minimal percentage.

CH-centralized election

A second possibility is to get the cNodes elected by the 2-CHs. In this way, each 2-CH elects the required number of cNodes (i.e. corresponding to user specifications). For example, if the 2-cluster contains 100 nodes and the desired percentage of cNodes in the network is 10 %, the 2-CH will compute 10 pseudo-random numbers and associate them with node IDs corresponding with sensors of its 2-cluster. This solution is computationally less demanding as only the 2-CHs have to run a pseudo-random

number generation algorithm. However it has yet another drawback: if a CH gets compromised, it won't be able to elect any cNode in its cluster thus leaving the cluster open to attacks. As, with the LEACH protocol, every sensor node becomes, sooner or later, a CH, the problem may occur for any compromised node hence propagating, potentially, throughout the network. Note that, nothing prevents a compromised sensor to declare itself as a CH node to the others at any round of the LEACH algorithm.

This method is the one that we have implement in our NS2 simulation whose simulation outcomes will be discussed in Section 4.

BS-centralized election

A third method consists in a centralized approach where the BS performs cNodes election. With this method CHs send the list of nodes that compose their clusters to the base station and the BS returns the list of elected cNodes. Observe that, opposite to sensor nodes, the BS has no limitation in memory, computing capacity nor energy. Thus the clear advantage of BS-centralized election is that all costly operations (i.e pseudo-random numbers calculation) can be re-iterated in a (virtually) unconstrained environment (i.e. the BS) This technique is explained in detail in [GM12].

From a robustness point of view not the method is not completely safe either. In fact if a compromised node was to declare itself as a CH, its escape method to avoid detection would consist in declaring its cluster as empty (i.e. by sending an empty list instead of the actual sensors in its cluster to the BS). In this case, the BS would not elect any cNode in its cluster hence the compromised CH would not be detected. To avoid such situation the BS should react differently in case it receives an indication of empty cluster from some nodes. Specifically, in this case, the BS would have to consider that nodes not detected as or by CHs might not simply be dead, thus still consider them as eligible cNodes. The main drawback of this method is that the distributed nature of election (together with its advantages) is completely lost.

3. MODELLING USING MARKOV CHAINS

Continuous Time Markov Chains (CTMC) are a class of discrete state stochastic process suitable to model discrete-event systems that enjoy the so-called *memory less* property (Markov property): i.e. systems such that the future evolution depends exclusively on the current state (and not on the *history* that lead into it). It is well known that in order to fulfill the Markov property delay of events must be Exponentially distributed.

In this section we describe how to structure Continuous Time Markov chains (CTMC) models for modelling of a WSN subject to DoS attacks and equipped with DoS detection functionalities. To illustrate the CTMC modeling

approach we focus on a specific (sub)class of WSN corresponding to the following points:

- the network consists of a single cluster containing one CH, N sensing nodes and K cNodes
- (exactly) one amongst the N sensing nodes is a compromised node.
- sensing node i ($1 \leq i \leq N$) generate traffic according to a Poisson process with rate λ_i
- the compromised node c generates traffic according to a Poisson process with rate $\lambda_c \gg \lambda_i$
- each cNodes periodically performs a detection check with period distributed Exponentially with rate μ . On detection of abnormal traffic a cNode reports the anomaly to the CH
- the network topology corresponds to a connected graph: each node node can reach any other node in the cluster

The dynamics of WSN systems agreeing with the above characterization can straightforwardly be modelled in terms of a $K \cdot (N+1)$ -dimensional CTMC. States of such a CTMC consist of K -tuples $x = (x_1, x_2, \dots, x_K)$ of macro-states $x_k = (x_{k_1}, x_{k_2}, \dots, x_{k_N}, x_{k_d})$ encoding the number of overheard packets by cNode k . More precisely component x_{k_j} ($1 \leq j \leq N$) of macro-state x_k is a counter storing the total number of packets sent by node j and overheard by cNode k , whereas component x_{k_d} is a boolean-valued variable which is set to 1 on detection, by cNode k , of abnormal traffic. We also consider a *threshold function* $f: \mathbb{N}^N \rightarrow \{0, 1\}$ which is used (by cNodes) to decide whether traffic rate have exceeded the "normal" threshold. The arguments of f are an (N) -tuples (n_1, \dots, n_N) , where $n_i \in \mathbb{N}$ is the number of overheard packets originating from node i .

We illustrate the transition equations for such a CTMC. For simplicity we illustrate only equations regarding transitions for a generic macro-state x_k : the equations for transitions of a generic (global) state $x = (x_1, x_2, \dots, x_K)$ can be straightforwardly obtained by combination of those for the macro-states. In the following x_{k_c} denotes the counter of received packets from the compromised node.

- $x_k \rightarrow$ Normal transmission
- $\rightarrow (x_{k_1}, \dots, x_{k_i} + 1, \dots, x_{k_c}, \dots, x_{k_N}, 0)$ with rate $\lambda_i \neq \lambda_c$
- \rightarrow Transmission by compromised node
- $\rightarrow (x_{k_1}, \dots, x_{k_i}, \dots, x_{k_c} + 1, \dots, x_{k_N}, 0)$ with rate λ_c
- \rightarrow Check and Detection of abnormal traffic
- $\rightarrow (0, \dots, 0, \dots, 0, \dots, 0, 1)$
with rate $\mu \times 1_{f(x_k) \geq \text{threshold}}$
- \rightarrow Check and No-Detection of abnormal traffic
- $\rightarrow (0, \dots, 0, \dots, 0, \dots, 0, 0)$
with rate $\mu \times 1_{f(x_k) < \text{threshold}}$

We assume that in the initial state all counters x_{k_i} as well as the boolean flag x_{k_d} are set to zero. The above equations can be described as follows. When cNode k is in state x_k a “Normal transmission” from node i ($1 \leq i \leq N$, $i \neq c$) takes place at rate λ_i leading to a state such that the corresponding counter x_{k_i} is incremented by one, leaving all remaining counters unchanged. Similarly a “Transmission by the compromised node” c happens with rate λ_c leading to a state such that the corresponding counter x_{k_c} is incremented by one. Finally checking for abnormal traffic conditions happens at rate μ and whenever the controlling function f detects that in (macro) state x_k the number of overheard packets from any node is above the considered threshold ($f(x_k) \geq \text{threshold}$), the detection flag x_{k_d} is raised (i.e. alarm is sent to the CH), and counters x_{k_j} are all resets (so that at the next check they are update with “fresh” traffic data). On the other hand if traffic has not been abnormal over the last $\text{Exp}(\mu)$ duration ($f(x_k) < \text{threshold}$) counters x_{k_j} are reset while the detection flag is left equal to zero.

The detection probability for cNode k (DP_k) can be computed in terms of the steady-state distribution of the above described CTMC in the following manner:

$$DP_k = \sum_{x_{k_1}, \dots, x_{k_N}} \pi(x_{k_1}, \dots, x_{k_N}, x_{k_d} = 1)$$

where $\pi(x_{k_1}, x_{k_2}, \dots, x_{k_N}, x_{k_d})$ denotes the steady-state probability at (macro)state $x_k = (x_{k_1}, x_{k_2}, \dots, x_{k_N}, x_{k_d})$ of the CTMC.

Discussion. The above described CTMC modeling approach relies on the assumption that the period with which detection checking is performed is an Exponentially distributed random variable. Indeed such an assumption may introduce a rather significant approximation as in reality detection checking happens at interval of fixed length, or even “continuously”. Therefore stochastic modeling of DoS attacks detection requires to exit the Markovian sphere and to consider non-Markovian stochastic processes. (more specifically periodic detection checking can more accurately be modeled by means Deterministic Distributions). We discuss non-Markovian modeling of DoS detection mechanisms in Section 5.

4. NUMERICAL RESULTS

A possible alternative to stochastic modeling is to develop executable implementations of the WSNs of interest by means of existing simulative framework, such as, for example, the NS-2 Network Simulator [NS2]. In this section, we present a selection of numerical results obtained by simulation of NS-2 models of WSN systems equipped with DoS detection mechanisms. The experiments we present are referred to one cluster consisting of a (10×10) regular grid topology with the following characteristics (see Figure 2):

- grid is a square of size a ;

- cluster head is placed at the centre of the grid (i.e. red node in Figure 2);
- the grid contains 100 (sensing) nodes displaced regularly;
- each node can communicate directly with the cluster head (i.e. the transmission power is such that all nodes — for example: the nodes in green in Figure 2 — can reach a circle of radius $a\sqrt{2}/2$. In this way all nodes, included corner’s, can reach the CH). No power adjustment is done by the nodes for transmission.

In such network cNodes (represented in green in Figure 2) are elected periodically either using the static approach or using the dynamic election mechanism described in previous sections. We have designed our experiments focusing on two performance measures: the rate of detection of attacks and the overall energy consumption. Table I reports about the (range of) parameters considered in our simulation experiments.

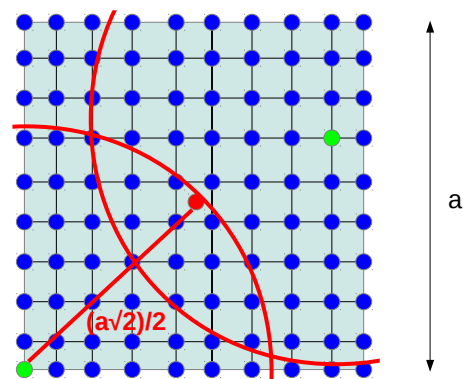


Figure 2. A 10×10 regular-grid cluster of size a

4.1. Detection rate

In order to evaluate the considered performance measure that is attack detection rate, we have considered the parameters given in table I. We have assumed that the traffic generation follows a Poisson distribution with rate λ which varies as the average transmission of an attacking node exceeds the average transmission of a normal node. In the experiments we have considered a cluster with 100 nodes.

Figure 3 represents the detection rate for different numbers of cNode groups and for groups of different sizes. The same node is considered compromised in all the graphs. Notice that for 10 cNodes, the group 2 did not detect any attack. With 15 cNodes, in average 3 nodes detect an attack in each group. We also note that when we increase the number of cNodes (20 and 25), the behavior

Simulation time	100–3,600 s
Rate	10–800 kbits/s
Packet size	500–800 bytes
Nodes number	100 (+ cluster head)
cNodes number	0–30
Compromised nodes number	1–10
Nodes queue size	50

Table I. Simulation parameters

remains similar which suggests that we do not need to use more nodes than 15 nodes in each group.

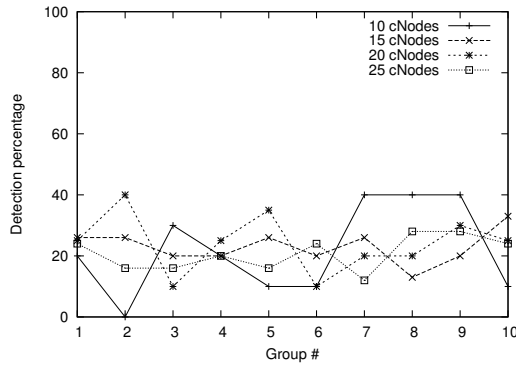


Figure 3. Detection versus Group.

Above $\lambda = 4$ packets/s, the dynamic method detects more attacks than the static one. To enhance this difference, we give other results in figure 4 below for an average of 10 compromised nodes.

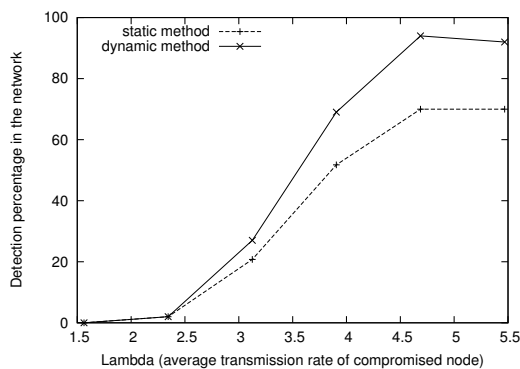


Figure 4. Detection versus Lambda.

In Figure 4, we notice that as the average transmission of attacking nodes increases, our dynamic solution detects more attacks than the static solution.

Number of sensor nodes	100
Simulation time	500 seconds
Reception consumption	0.394 W
Emission consumption	0.660 W

Table II. Simulation parameters

4.2. Consumed energy

All the simulations which were run to produce the results presented in this section used the parameters given in table II.

The Figure 5 shows the average energy consumption for all nodes (except for the cluster head and the flooding compromised node, which consume much more than usual nodes, and act in the same way for both methods) at the end of the simulation, for various percentages of elected cNodes. The number of cNodes goes from 0 (no detection) to 30 % (nearly one third of the nodes).

Note that the “normal nodes” (non-cNodes sensors) do not receive messages from their neighbors, as they are “sleeping” between their sending time slots (see LEACH detailed functioning).

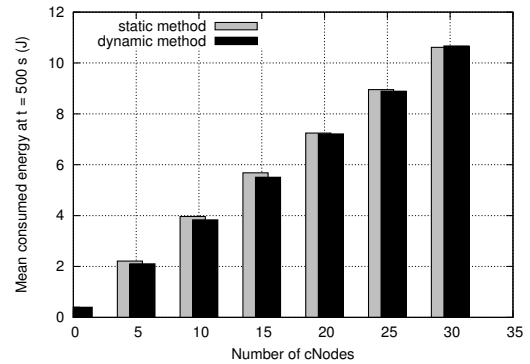


Figure 5. Average energy consumption.

The average consumption is the same for static and dynamic method: both method use the same quantity of normal and cNode sensors.

The Figure 6 depicts the standard deviation for the energy consumption at the end of the simulation. Once again, the cluster head and the compromised node are not taken into account.

One can observe that the standard deviation is much higher for the static solution: only the initial (and not re-elected) cNodes have a significant consumption over the simulation time, while the consumption is distributed among all the periodically-elected nodes in the dynamic solution.

For the Figure 7, we have supposed that the nodes have an initial energy of 4 J. This is a small value, but 500 seconds is a small duration for a sensor lifetime. According to Wikipedia values and to what we have computed, a

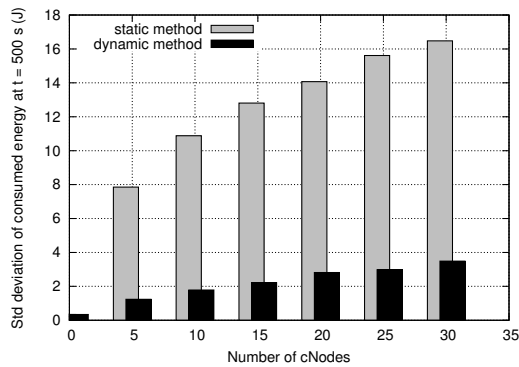


Figure 6. Energy consumption standard deviation.

lithium battery (CR1225) can offer something like 540 J, and a LR06 battery would provide something like 15,390 J. Note that the compromised node was given an extra initial energy (we did not want it to stop flooding the network during the simulation). However, we set the initial energy to 4 J, and we notice for the first node’s death for several percentages of cNodes.

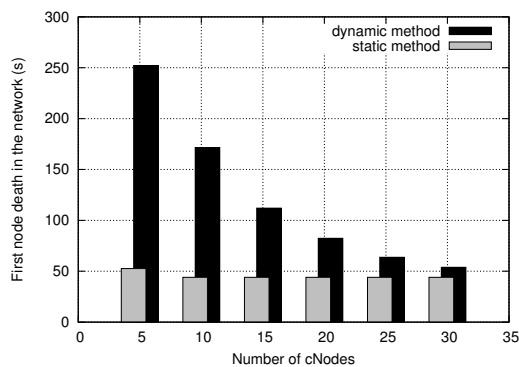


Figure 7. First death in the network.

As the cNodes are re-elected and the consumption is distributed for the dynamic method, the first node to run out of battery power logically dies later (up to 5 times later with few cNodes) than in the static method.

4.3. Nodes’ death and DoS detection

The duration of this new simulation was extended to one hour (3,600 seconds). 10 % of the sensors are elected as cNodes. The initial energy power was set to 10 J. So the considered parameters are given in table IV.

The Figure 8 shows the evolution of the number of alive nodes in time.

As for the previous section, the non-cNodes sensors barely consume any energy regarding to cNodes’ consumption (cNodes consume each time they analyze a message coming from one of their neighbor; other sensors

Number of sensor nodes	100
cNodes percentage	10 %
Simulation time	3,600 seconds
Reception consumption	0.394 W
Emission consumption	0.660 W
Initial energy amount	10 J

Table III. Simulation parameters

Table IV. .

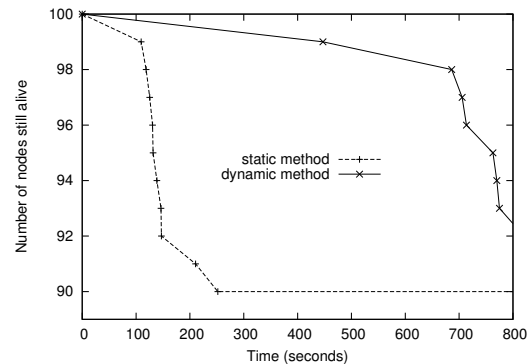


Figure 8. Nodes remained alive

don’t). In the static method, elected cNodes consume their battery power, and die (at about $t = 150$ seconds). That is why the ten first sensors die quickly, whereas the other nodes last much longer (we expect them to live for 5 hours). For the static method, the cNodes are re-elected, so the first node to die lives longer than for the previous method. It is a node that was elected several times, but not necessarily *each* time. Only two nodes have run out of energy at $t = 700$ seconds for the dynamic method. But at this point, the amount of alive nodes decreases quickly, and there is only one node left at the end of the first hour of simulation. Note that this was not reported on the above curve.

It is obvious that the nodes die much faster in the dynamic method, given that cNodes, the only nodes whose consumption is significant, are re-elected, whereas there are no more consuming cNodes in the network for the static method after the ten first nodes are dead. Hence it is interesting to consider how many nodes do effectively detect the attack as the time passes by. This is what is shown on the Figure 9. The average number of cNodes which detected the attack (out of 10 cNodes) is presented for each 60 second-long period.

After the fourth minute, every cNode is dead for the static method, and the compromised node is no more detected. With the dynamic method, a raw average of 6.5 out of 10 cNodes detect the compromised nodes during each 10 second-long period corresponding to the dynamic

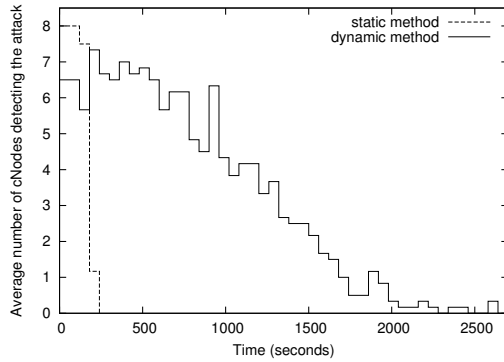


Figure 9. DoS detection.

election. The flooding sensor is still detected by more than one node after half an hour.

In the following, we present a modeling approach of DoS detection using of Generalized Stochastic Petri Nets (GSPN) and we give some numerical results.

5. NON-MARKOVIAN MODELING AND VERIFICATION OF DOS

In previous sections we have pointed out that using Markov chains to model DoS detection mechanisms may inherently imply a significant approximation. To obtain more accurate models of DoS detection it is necessary to resort to a more general class of stochastic processes, namely the so-called Discrete Event Stochastic Processes (DESP, also often referred to as Generalized Semi-Markov Processes or GSMP). The main characteristics of DESP is that they allow for representing generally distributed durations, rather than, as with CTMC, being limited to Exponentially distributed events.

In this section we present a modeling approach of DoS detection in terms of Generalized Stochastic Petri Nets (GSPN) [AMBCDF95], a class of Petri Nets suitable for modeling stochastic processes. By definition the GSPN formalism is a high-level language for representing CMTCs. However herein we refer to its straightforward extension where *timed-transitions* can model generally distributed durations. Such extended GSPN (eGSPN in the following) becomes a high-level language for representing DESPs. Furthermore eGSPN is also the formal modeling language supported by the COSMOS [BDDHP11] statistical model checker, a tool which allows for verification of (sophisticated) performance measures in terms of the Hybrid Automata Stochastic Logic (HASL) [BDDHP11]

In the following we provide a succinct description of both the GSPN modeling formalism and the HASL verification approach, before describing their application to the DoS attack detection case.

5.1. Generalized Stochastic Petri Nets

A GSPN model is a bi-partite graph consisting of two classes of nodes, *places* and *transitions* (Figure 10). Places (represented by circles) may contain *tokens* (representing the state of the modeled system) while transitions (represented by bars) indicate the events the occurrence of which determine how tokens “flow” within the net (thus encoding the model dynamics). The state of a GSPN consists of a *marking* indicating the distribution of tokens throughout the places (i.e. how many tokens each place contains). Roughly speaking a transition is enabled whenever all of its *input places* contains a number of tokens greater than or equal to the multiplicity of the corresponding input arc (e.g. transition T1 in left-hand part of Figure 10 is enabled, while T2 is not). An enabled transition may *fire* consuming tokens (in a number indicated by the multiplicity of the corresponding input arcs) from all of its input places and producing tokens (in a number indicated by the multiplicity of the corresponding output arcs) in all of its output places. Such informally described rule is known as the Petri Net *firing rule*. GSPN transitions can be either *timed* (denoted by empty bars) or *immediate* (denoted by filled-in bars, e.g. transition T2 in left hand side of Figure 10). Generally speaking transitions are characterized by: (1) a distribution which randomly determines the delay before firing it; (2) a priority which *deterministically* selects among the transitions scheduled the soonest, the one to be fired; (3) a weight, that is used in the random choice between transitions scheduled the soonest with the same highest priority. With the GSPN formalism the delay of timed transitions is assumed *exponentially* distributed, whereas with eGSPN it can be given by any distribution with non-negative support. Thus whether a GSPN timed-transition is characterized simply by its weight $t \equiv w$ ($w \in \mathbb{R}^+$ indicating an $Exp(w)$ distributed delay), a eGSPN timed-transition is characterized by a triple: $t \equiv (\text{Dist-}t, \text{Dist-}p, w)$, where Dist- t indicates the type of distribution (e.g. Unif, Deterministic, LogNormal, etc.), dist- p indicates the parameters of the distribution (e.g. $[\alpha, \beta]$) and $w \in \mathbb{R}^+$ is used to probabilistically choose between transitions occurring with equal delay*

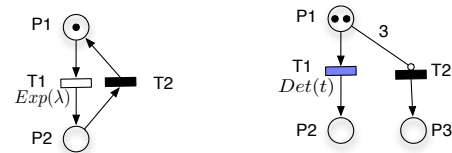


Figure 10. Simple examples of eGSPN: timed-transitions, immediate transition and inhibitors arcs

*a possible condition in case of non-continuous delay distribution

In the following we describe how eGSPN models can be derived for modeling WSN scenario with DoS mechanisms. More specifically in our eGSPN models we will use only two types of timed transitions, namely: Exponentially distributed timed transitions (denoted by empty-bars, e.g. T1 in left-hand side of Figure 10) and Deterministically distributed timed-transitions (denoted by blue-filled-in bars, e.g. T1 in right-hand side of Figure 10). In our Petri Nets models we will also extensively exploit *inhibitor arcs*, an additional element of the GSPN formalism. An inhibitor arc is denoted by an edge with an empty-circle in place of an arrow at its outgoing end (e.g. the arc connecting place P1 to transition T2 in the right hand side of Figure 10). In presence of inhibitor arcs the semantics of GSPN *firing rule* is slightly modified, thus: a transition is enabled whenever all of its *input places* contains a number of tokens greater than or equal to the multiplicity of the corresponding input arc and strictly smaller than the multiplicity of the corresponding inhibitor arcs (e.g. transition T2 in right-hand part of Figure 10 is also enabled, because P1 contains less than 3 tokens). Having summarized the basics of the syntax and semantics of the eGSPN formalism we now describe how it can be applied to formally represent WSN systems featuring DoS mechanisms.

5.2. Modeling DoS attacks with eGSPN

We describe the eGSPN models we have developed for modeling DoS attacks in a grid-like network. For simplicity we illustrate an example referred to a 9x9 grid topology. The proposed modeling approach can easily be extended to larger networks.

In a WSN with DoS detection mechanisms the functionality of sensing nodes is different from that of cNodes. Here we describe GSPN models for representing: i) sensing nodes, ii) statically elected cNodes and iii) dynamically eligible cNodes.

5.2.1. GSPN model of sensing nodes

Sensing nodes functionality is trivially simple: they simply keep sending sensed data packets at a pace which (following Section 3) we assume, being Exponentially distributed with rate λ_i . This can be modeled by a simple

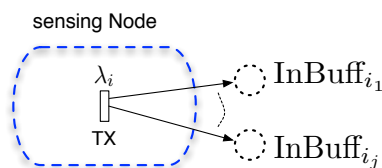


Figure 12. GSPN model of a sensing node

GSPN that consists of a single exponentially distributed timed-transition (labeled TX) with no input places (i.e. always enabled) and with as many outgoing arcs leading to the input buffer of the neighboring nodes (represented

by dashed places labelled InBuff_i , in Figure 12). Note that transition TX in Figure 12 has no input places, which means (according to the Petri Net firing rule) that it is always (i.e. perpetually) enabled. Note also that TX is an exponentially distributed timed transition with rate λ_i , which complies with the assumption that each sensor node performs a sensing operation every δ_s time with $\delta_s \sim \text{Exp}(\lambda_i)$. To summarize: the *sensing* functionality of a specific node in WSN is modeled by a single timed-transition provided with as many outgoing arcs as the number of neighbors of that node. The complete sensing functionality of a WSN can be modeled by combining several such GSPN modules.

5.2.2. GSPN model of cNodes

A cNode functionality, on the other hand, is entirely devoted to monitoring of traffic of the portion of WSN it is guarding on. From a modeling point of view a distinction must be made between the case of statically elected cNodes (as in [LC08]) and that of dynamically eligible cNodes (as in [GM12]). In fact with dynamic cNodes election each node in the network can be elected as cNode, therefore each node can switch between a sensing-only functionality and a controlling functionality. On the other hand static cNodes will be control-only nodes.

GSPN models for both *static* and *dynamic* cNodes are depicted in Figure 11(a), respectively Figure 11(b). A cNode detects an attack whenever the overheard traffic throughput (i.e. number of overheard packets per observation period) exceeds a given threshold ρ_{attack} . Place “InBuff” (Figure 11(a)) represents the *input buffer* of a node, where packets received/overheard from neighbors nodes are placed. The “InBuff” place receives tokens (corresponding to overheard packets) through input arcs originating from neighbors sensing-node modules (i.e. the input arcs of place “InBuff” are the output arcs of the timed-transition representing the corresponding sensing activity of each neighbor node).

To model the traffic monitoring functionality of cNodes we employ two mutually exclusive, deterministically distributed timed-transitions labelled “checkYES” and “checkNO” in Figure 11(a) and Figure 11(b). They correspond to the periodic verification performed by the cNode to check whether the frequency of incoming traffic has been abnormal (over the last period). At the end of each (fixed) interval $[0, \Delta]$ either: transition “checkYES” is enabled, if at least k packets have been received (i.e. place “InBuff” contains at least k tokens); or transition “checkNO” is enabled, if less than k packets have been received (i.e. place “InBuff” contains less than k tokens); in the first case (i.e. “checkYES” enabled) a token is added in the output place “det” representing the occurrence of an DoS detection, otherwise (i.e. “checkNO” enabled) no tokens is added to place “det”. After firing of either the “checkYES” or the “checkNO” transition the emptying of the input-buffer starts by adding a token in place “empty”. This enables either immediate transition “e-on” (which

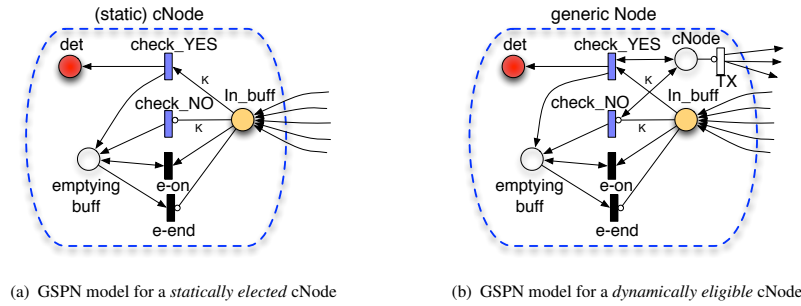


Figure 11. GSPN components representing cNodes behaviour in a WSN with DoS detection mechanisms

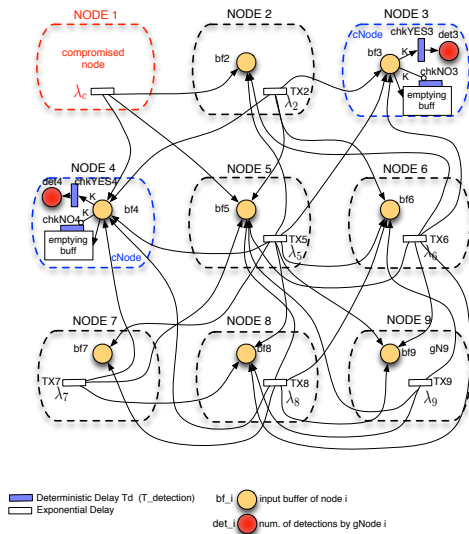


Figure 13. GSPN model of a 9x9 grid-topology with 1 (fixed) compromised node and 2 static cNodes

iteratively fires until the input buffer is empty), or “e-end” which represents the end of the emptying cycle. Note that buffer emptying does not consume time, and it is needed in order to correctly measure the frequency of traffic at each successive sampling interval $[0, \Delta]$.

The GSPN model for the dynamic cNodes (Figure 11(b)) is a simple extension of that for static cNodes obtained by adding an auxiliary place “cNode” and an auxiliary exponentially distributed timed-transition “TX”. This is needed because with dynamically elected cNodes, each node in the network may periodically switch from sensing-only to controlling-only functionality, hence the corresponding GSPN model must represent both aspects. If the auxiliary place “cNode” contains a token then the “controlling” functionality (i.e. the left part of the GSPN) is switched-on, and in that case the GSPN of Figure 11(b) behaves exactly as that of Figure 11(a). Conversely if place cNode is empty then the “sensing” functionality is switched-on (i.e. transition “TX” is enabled due to the

inhibitor arc between place “cNode” and transition “TX”) while the “controlling” part of the net is disabled (i.e. in this case the net of Figure 11(b) behaves exactly as that of Figure 12).

The above described GSPN models for sensing-nodes, static cNodes and dynamic cNodes can be used as basic building blocks to compose models of specific WSN topologies. In the following we provide examples of GSPN for 9x9 WSN grid-topology equipped with DoS detection functionalities.

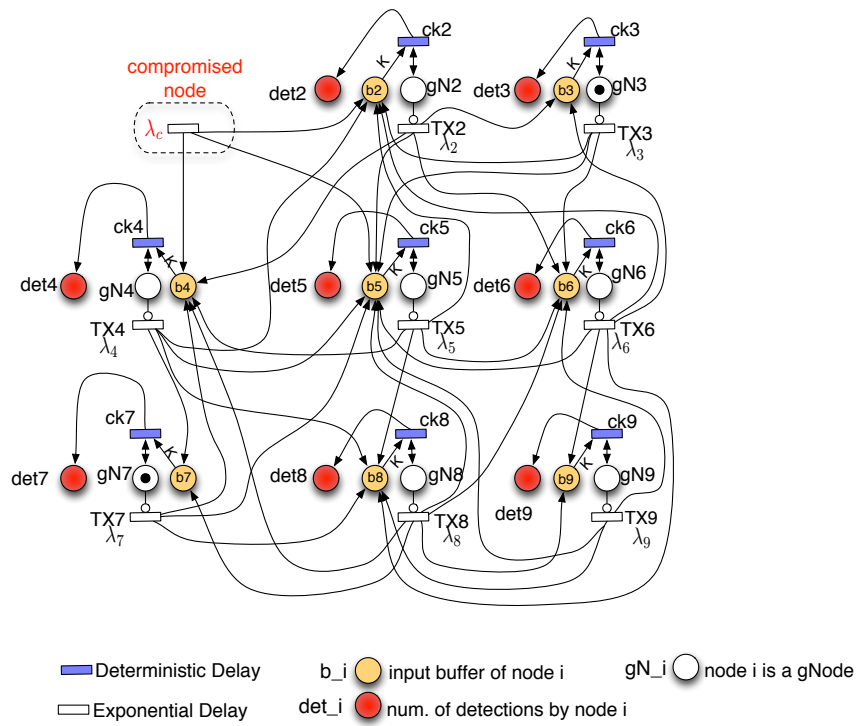
5.2.3. GSPN model of DoS detection with static cNodes

Figure 13 illustrates a complete GSPN model for a 9x9 grid topology representing an example of DoS detection with static election of cNodes (as in [LC08]). In particular in this example we consider the presence of 2 cNodes (i.e. node 3 and 4) and 1 compromised node (i.e. node 1). Note that for simplicity the “emptying buffer” part in the GSPN modules of the cNodes (i.e. node 3 and 4) is depicted as a box (i.e. the content of that box corresponds to the subnet responsible for emptying the “inBuff” place as depicted in Figure 11(a) and Figure 11(b)).

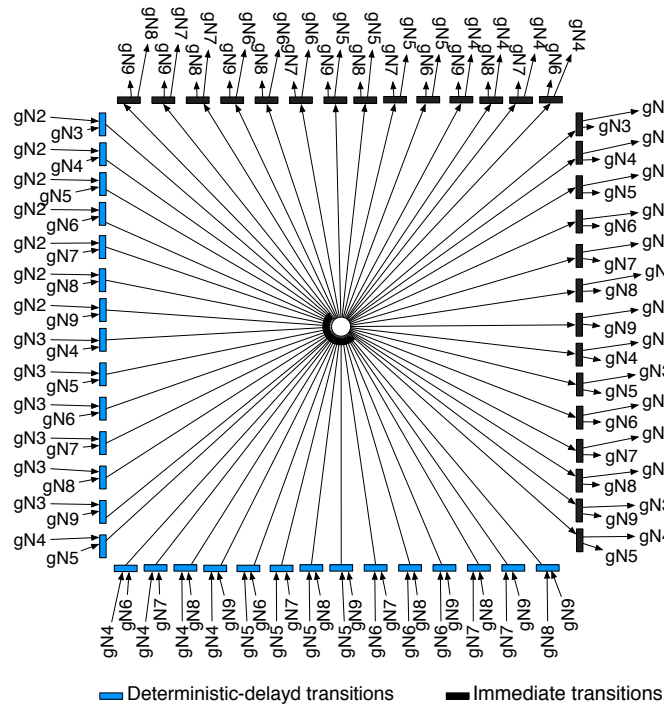
This model can be used to study the performances of DoS detection with static cNodes in many respect, such as: measuring the expected number of detected attacks within a certain time bound, or also e.g. assessing the average energy consumption of cNodes. In the next section we describe how to build GSPN models of WSNs with DoS detection and dynamic election of cNodes. The resulting GSPN is more complex than that for statically elected cNodes, as it must include an extra module, namely a GSPN module for periodically electing the cNodes.

5.2.4. GSPN model of DoS detection with dynamic cNodes

Figure 14 illustrates the GSPN model of a 9x9 grid topology for the case of DoS detection with dynamic election of cNodes (as in [GM12]). For simplicity Figure 14 consists of two parts: the actual network topology part (Figure 14(a)) and the cNodes random election mechanism (Figure 14(b)). The network model



(a) the traffic part in a 9x9 topology



(b) the random election policy part: 2 cNodes are elected out of 8

Figure 14. GSPN model of a 9x9 grid-topology with 1 (fixed) compromised node and 2 randomly elected cNodes

(Figure 14(a)) is obtained by composition of node's GSPN component in the same fashion as for the model of the WSN for DoS detection with static cNodes, only that now all nodes must be reconfigurable as either sensors or controllers (thus the basic GSPN components used to build the network topology are those of Figure 11(b)). The cNodes election component (Figure 14(b)), on the other hand, consists of a single place, n mutually-exclusive deterministically distributed timed-transitions (blue-filled) and n mutually-exclusive immediate transitions (black-filled) (with $n = \binom{8}{2} = 28$, as we assume that, at each round, 2 cNodes are elected out of 8 possible candidates, thus, for simplicity we rule out the compromised node from the eligible ones). The deterministically distributed timed-transitions (blue-filled) of Figure 14(b) correspond to all possible different pairs of "cNode" places. At the end of each selection period only (exactly) one such timed-transition will be enabled and will fire retrieving, in this way, the tokens from the current pairs of active cNodes and inserting one token in the only (central) place of the net in Figure 14(b). At this point all 28 immediate transitions will become enabled and a random choice will take place resulting in the selection of only (exactly) one of them. The selected transition will fire and by doing so will insert one token into each "cNode" place of the corresponding pair of cNodes to which it is connected, activating, in this way, the controlling functionality of the newly elected cNodes.

5.3. HASL verification of DoS detection models

One of the main motivation for developing GSPN models of discrete-event systems is that a fairly large and well established family of formal methods can be applied to analyze them. Recently a new formalism called Hybrid Automaton Stochastic Logic (HASL) has been introduced which provides a unified framework both for model checking and for performance and dependability evaluation of DESP models expressed in GSPN terms. In essence, given a GSPN model, we can express sophisticated performance measures in terms of an HASL formula and apply a statistical model checking functionalities to (automatically) assess them. In the following we informally summarize the basics about the HASL verification approach, referring the reader to [BDDHP11] for formal details.

5.3.1. HASL model checking

Model checking [CGP99] is a formal verification procedure by which given a (discrete-state) model M and a property formally expressed in terms of a temporal logic formula ϕ , an algorithm automatically decides whether ϕ holds in M (denoted $M \models \phi$). In the case of stochastic models (i.e. stochastic model checking [KNP07a]) formulae are associated with a measure of probability and verifying $M \models \phi$ corresponds to assess the probability of ϕ with respect to the stochastic model M . HASL model checking extends this very simple concept in the sense that an HASL formula can evaluate

to any real number (thus it can represents a measure of probability as well as other performance measures). To do so HASL uses Linear Hybrid Automata (LHA) as machineries to encode the dynamics (i.e. the execution paths, or trajectories) of interest of the considered GSPN model. An LHA, simply speaking, is a generalization of Timed Automaton where clock-variables are replaced by real-valued data-variables. In practice a formula of HASL consists of two parts:

- an LHA used as a selector of relevant of timed execution of the considered DESP (path selection is achieved by synchronization of a generated DESP trajectory with the LHA).
- an expression Z built on top of data variables of the LHA according to the syntax given in (1) and which represent the measure to be assessed.

$$\begin{aligned}
 Z &::= E(Y) \mid Z + Z \mid Z \times Z \\
 Y &::= c \mid Y + Y \mid Y \times Y \mid Y/Y \mid \text{last}(y) \mid \text{min}(y) \\
 &\quad \mid \text{max}(y) \mid \text{int}(y) \mid \text{avg}(y) \\
 y &::= c \mid x \mid y + y \mid y \times y \mid y/y
 \end{aligned} \tag{1}$$

The informal meaning of an HASL expressions Z (1) is as follows: x is a data-variable of the LHA automaton associated to the expression. y is an (arithmetic) expression of data-variables. Y is a path random variable, i.e. a variable which is evaluated against a synchronization path, a path resulting by the synchronization of a trajectory of the DESP with the LHA associated to the formula. The basic operators (i.e. $\text{last}(y)$, $\text{min}(y)$, $\text{max}(y)$, $\text{int}(y)$, $\text{avg}(y)$) on top of which a path variable Y is built have intuitive meanings. In particular: $\text{last}(y)$ indicates the last value of expression y along an accepted synchronized path; $\text{min}(y)/\text{max}(y)$ indicates the minimum (maximum) of y along a path; $\text{int}(y)$ the integral of y along a path; $\text{avg}(y)$ the average of y along a path.

The HASL statistical model checking procedure works as follow:

- it takes a GSPN model and an HASL formula
- it iteratively generate trajectories of GSPN model state-space and synchronize them with the LHA
- the trajectories that have been "accepted" by the LHA are considered in the estimation of the measure of interest, the others are dropped.

5.4. HASL formulae for DoS models

Having seen the nature of HASL verification we provide here few examples of HASL formulae (i.e. LHA + expression) which can be used to assess performance measures of the DoS (GSPN) models presented in the previous section. Such formulae may be readily assessed through the COSMOS model checker and the results can be used to compare different DoS detection mechanisms.

The LHA we present are based on the following data-variables:

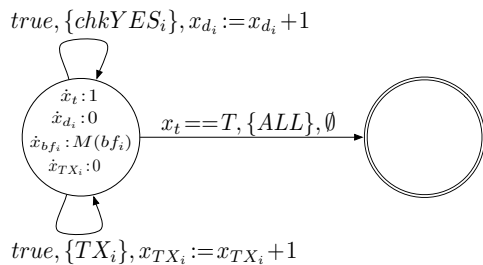


Figure 15. An LHA for assessing relevant measures of DoS GSPN models

- x_t : global time
- x_{d_i} : num. of attacks detected by cNode i ($1 \leq i \leq N$)
- x_{TX_i} : num. of data transmitted by node i ($1 \leq i \leq N$)
- $x_{b_{f_i}}$: flow of packets in buffer of node i ($1 \leq i \leq N$)

The LHA in Figure 15 is a template automaton that can be used for calculating different measures of a node (either a sensing or a cNode) of a WSN model. It refers to GSPN models (Figure 13, Figure 14). It consists of 2 locations and refers to the 4 data-variables above described. In the initial-location (l_1) the rate of change (i.e. the first derivative) of data-variables is indicated (inside the circle). The global time variable x_t is incremented with rate $\dot{x}_t = 1$ following the linear flow of time. Counter variables x_{d_i} and x_{TX_i} (used to count occurrences of events) are unchanged in location l_1 (i.e. their rates are zero). Finally variable $x_{b_{f_i}}$ is incremented with rate proportional to the number of tokens in the input buffer of cNode i (i.e. $\dot{x}_{b_{f_i}} = M(b_{f_i})$); this data-variable can be used to measure the average length of overheard packets by cNode i , and thus to measure the average energy consumption of a cNode. The two self-loops transitions on location l_1 are used to increment the the counter variables x_{d_i} and x_{TX_i} on occurrence of the associated events in the GSPN model. For example transition $l_1 \xrightarrow{\text{true}, \{chkYES_i\}, x_{d_i} := x_{d_i} + 1} l_1$ indicate on occurrence of the GSPN transition labeled $chkYES_i$ (i.e. detection of an attack by cNode i) the variable x_{d_i} is incremented by 1. Transition $l_1 \xrightarrow{x_t == T, \{ALL\}, \emptyset} l_2$ from l_1 to the accepting location l_2 indicates when the synchronization stop and the processed path is accepted. Precisely this happens as soon as $x_t == T$, where $T \in \mathbb{R}$ denotes a time-bound, that is: as soon as the observed trajectories is such that the simulation time is T . In this case, no matter which GSPN transition is occurring (i.e. synchronization set is $\{ALL\}$) the transition from l_1 to l_2 will fire and the path generation will stop by accepting the path. In other words the LHA in Figure 15 trivially accepts all paths of time duration T . The value of the 4 data variables collected during synchronization of the LHA with the GSPN model will be then used for estimating relevant Z expressions.

In the following we describe few examples of Z expressions that can be used in association to the LHA in

Figure 15 to evaluate relevant measures of the DoS GSPN models.

- $Z_1 \equiv E(\text{Last}(x_{d_i}))$: the expected num. of detected attacks by cNode i after T time units
- $Z_2 \equiv E(\text{Last}(x_{d_i} + x_{d'_i}))$: the sum of attacks detected by cNode i and i' after T time units
- $Z_3 \equiv E(\text{Last}(x_{TX_i}))$: the expected value of packets transmitted by node i after T time units
- $Z_4 \equiv E(\text{Avg}(x_{b_{f_i}}))$: the expected cumulative flow of packets received by node i within T time units

6. CONCLUSION

Detection of DoS attacks is a fundamental aspect of WSN management. In this paper we have considered a class of DoS detection mechanisms designed to operate on clustered WSNs. The detection methods we have considered are based on deployment of special control nodes in the sensing field: i.e. specific nodes which are responsible for monitoring the throughput of traffic of specific parts of the sensing field and signaling the presence of suspected attacked nodes in case anomalies are detected. Control nodes election is a crucial aspect of DoS mechanisms. In the literature two basic election approaches have been proposed: a static election and a dynamic (random) election.

In this paper we presented different modeling approaches for obtaining models of WSNs with DoS functionalities. First we have described how Markov chains model should be structured for modeling DoS attack and detection, pointing out that because of the nature of DoS detection, Markovian models may inherently come with some significant approximation. We have then presented numerical results obtained with virtual WSN implementation by means of the NS-2 network simulator. The outcome of such simulative experiments confirm the intuition that cNodes dynamic allocation guarantees a more uniform energy consumption (throughout the network) while preserving a good detection capability. Finally we have presented formal non-Markovian models of DoS detection in terms of Generalized Stochastic Petri Nets, a high level formalism for generic Discrete Event Stochastic Process. We have illustrated how model of WSNs with DoS can be built “incrementally” by combination of small GSPN modules of single (sensing/controlling) nodes up to obtaining a model of the desired network. We have also stressed how the GSPN formalism is naturally well suited for modeling of the dynamic random cNodes election policy. Finally we have briefly presented how expressive performance measures of the DoS GSPN models can be formally expressed and assessed by means of the recently introduced Hybrid Automata Stochastic Logic. Future developments of this work include the execution of actual verification experiments on the presented GSPN models by means of the COSMOS statistical model checker, as well as the

extension of the proposed modeling approaches to consider more complex network (different topologies and scales).

REFERENCES

- [GA09]. G. Anastasi et al: *Energy conservation in wireless sensor networks: A survey*. Ad Hoc Networks 7, p. 537–668 (2009).
- [LB09]. B. Li and L. Batten: *Using mobile agents to recover from node and database compromise in path-based DoS attacks in wireless sensor networks*. Journal of Network and Computer Applications 32, p. 377–387 (2009).
- [CS11]. W.R. Claycomb and D. Shin: *A novel node level security policy framework for wireless sensor networks*. Journal of Network and Computer Applications 34, p. 418–428 (2011).
- [MS11]. M.A. Simplicio Jr. et al: *Comparison of Authenticated-Encryption Schemes in Wireless Sensor Networks*, 36th Annual IEEE Conference on Local Computer Networks, Bonn, Germany, 2011.
- [HS05]. F. Hu and N. Sharma: *Security considerations in ad hoc sensor networks*. Ad Hoc Networks 3, pp. 69–89 (2005).
- [ATB05]. J. Ai, D. Turgut and L. Bölöni: *A Cluster-based Energy Balancing Scheme in Heterogeneous Wireless Sensor Networks*. Proceedings of International Conference on Networking, p.467–474 (April 2005).
- [HW00]. W.R. Heinzelman et al: *Energy-Efficient Communication Protocol for Wireless Microsensor Networks* Proceedings of the IEEE Hawaii international conference on system sciences, 2000.
- [OX09]. S. Ozdemir and Y. Xiao: *Secure data aggregation in wireless sensor networks: A comprehensive overview*. Computer Networks 53, p. 2022–2037 (2009).
- [YF04]. O. Younis and S. Fahmy: *HEED: a hybrid, energy-efficient distributed clustering approach for ad hoc sensor networks*. IEEE Trans. Mobile Comput. 3 (4), p. 366–379 (2004).
- [FL11]. S. Fouchal and I. Lavallée: *Fast and Flexible Unsupervised Clustering Algorithm Based on Ultrametric Properties*. Proceedings of the 7th ACM Symposium on QoS and Security for Wireless and Mobile Networks, Miami, USA, 2011.
- [YL11]. Y. Liang: *Efficient Temporal Compression in Wireless Sensor Networks*, 36th Annual IEEE Conference on Local Computer Networks, Bonn, Germany, 2011.
- [LC08]. G. Hsin Lai and C-M. Chen: *Detecting Denial of Service Attacks in Sensor Networks*. Journal of Computers Vol.18, No.4 (January 2008).
- [GM12]. M. Guechari, L. Mokdad and S. Tan: *Dynamic Solution for Detecting Denial of Service Attacks in Wireless Sensor Networks*. ICC (June 2012).
- [YB10]. Bashir Yahya, Jalel Ben-Othman: *Energy efficient and QoS aware medium access control for wireless sensor networks*. Concurrency and Computation: Practice and Experience 22(10): 1252-1266 (2010)
- [BY10]. Jalel Ben-Othman, Bashir Yahya: *Energy efficient and QoS based routing protocol for wireless sensor networks*. J. Parallel Distrib. Comput. 70(8): 849-857 (2010)
- [MM09]. M. Mohi, A. Movaghar and P.M. Zadeh: *A Bayesian Game Approach for Preventing DoS Attacks in Wireless Sensor Networks*. International Conference on Communications and Mobile Computing (2009).
- [JH10]. J-W. Ho: *Distributed Detection of Node Capture Attacks in Wireless Networks*. Smart Wireless Sensor Networks, H.D. Chinh and Y.K. Tan (Ed.) (2010).
- [SM10]. S. Misra et al: *An adaptive learning routing protocol for the prevention of distributed denial of service attacks in wireless mesh networks*. Computers and Mathematics with Applications 60, p. 294–306 (2010).
- [AP01]. A. Perrig et al: *SPINS: Security Protocols for Sensor Networks*. Mobile Computing and Networking, Rome, Italy (2001).
- [LO07]. L.B. Oliveira et al: *SecLEACH – On the security of clustered sensor networks*. Signal Processing 87, p. 2882–2895 (2007).
- [HI09]. M.H. Islam et al: *Optimal Sensor Placement for Detection against Distributed Denial of Service Attacks*. Pak. J. Engg. & Appl. Sci. Vol. 4, p. 80–92 (2009).
- [JS10]. J-H. Son et al: *Denial of service attack-resistant flooding authentication in wireless sensor networks*. Computer Communications 33, p. 1531–1542 (2010).
- [MH07]. M-Y. Hsieh et al: *Adaptive security design with malicious node detection in cluster-based sensor networks*. Computer Communications 30, p. 2385–2400 (2007).
- [MH02]. M. J. Handy, M. Haase, D. Timmerman: *Low Energy Adaptive Clustering Hierarchy with Deterministic Cluster-Head Selection*. Proceedings of the 4th IEEE conference on Mobile and Wireless Communications Networks, Stockholm, Sweden (2002).
- [SF11]. S. Fouchal, I. Lavallée: *Fast and Flexible Unsupervised Clustering Algorithm Based on Ultrametric Properties*. Proceedings of the 7th ACM Symposium on QoS and Security for WSN, Miami, USA (2011).
- [BDDHP11]. P. Ballarini, H. Djafri, M. Duflo, S. Haddad, N. Pekergin: *HASL: An Expressive Language for Statistical Verification of Stochastic Models*. Proceedings of the 5th International Conference on Performance Evaluation Methodologies and Tools (ValueTOOLS11), Cachan, France (2011).
- [BDDHP11+]. P. Ballarini, H. Djafri, M. Duflo, S. Haddad, N. Pekergin: *COSMOS: a Statistical Model Checker for the Hybrid Automata Stochastic Logic*. Proceedings of the 8th International Conference on Quantitative Evaluation of Systems (QEST11), Aachen, Germany (2011).

- [NS2]. *The Network Simulator NS-2*. <http://www.isi.edu/nsnam/ns/>
- [AMBCDF95]. M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. *Modelling with Generalized Stochastic Petri Nets*. John Wiley & Sons (1995).
- [BBBP12]. J. Ben-Othman, Karim Bessaoud, Alain Bui, Laurence Pilard: *Self-stabilizing algorithm for efficient topology control in Wireless Sensor Networks*. Elsevier, Journal of Computational Science (JOCS), In press.
- [FH11]. H. Fouchal, H. and Z. Habbas: *Distributed backtracking algorithm based on tree decomposition over wireless sensor networks*. Concurrency and Computation: Practice and Experience. doi: 10.1002/cpe.1804 (September 2011).
- [BB12]. J. Ben-Othman and Y. Saavedra Benitez: *IBC-HWMP: a novel secure identity-based cryptography-based scheme for Hybrid Wireless Mesh Protocol for IEEE 802.11s*. Concurrency and Computation: Practice and Experience, Wiley, In press
- [DFHV10]. N. Dessart, H. Fouchal, P. Hunel et N. Vidot: *Anomaly Detection with Wireless Sensor Networks* The 9th IEEE International Symposium on Network Computing and Applications (NCA 2010), IEEE CS Press, Cambridge, MA, USA (July 2010).
- [BF12]. T. Bernard and H. Fouchal: *A low energy consumption MAC protocol for WSN*. IEEE ICC 2012, Ottawa, Canada (June 2012).
- [KNP07a]. M. Kwiatkowska and G. Norman and D. Parker: *Stochastic Model Checking*, In book “Formal Methods for the Design of Computer, Communication and Software Systems: Performance Evaluation (SFM’07)”, LNCS (Tutorial Volume), Vol. 4486 (2007).
- [CGP99]. E. M. Clarke and O. Grumberg and D. A. Peled: *Model Checking*. MIT Press (1999).