Modeling and Simulation of Computer Networks and Systems

Methodologies and Applications

Mohammad S. Obaidat, Faouzi Zaraia, Petros Nicopolitidis

Elsevier
April 2015

Link: http://store.elsevier.com/
Modeling-and-Simulation-of-Computer-Networks-and-Systems/
isbn-9780128008874/

Part 6:

Modeling and simulation for system security

Chapter 28

(pages 797 to 842)

This version received minor corrections in May 2015.

Contents

1 Introduction **2**

2 Related work **5**
2.1 Security in WSNs 5
2.2 DoS-specific mechanisms 5
2.3 Clustering algorithms and energy preservation 7
2.3.1 LEACH functioning 7
2.3.2 LEACH improvements 8

3 Pseudo-random self-election of the cNodes **9**
3.1 Detection of DoS attacks 9
3.1.1 Wireless Sensor Networks 9
3.1.2 Hierarchical clustering 10
3.1.3 Attacks detection through cNodes 11
3.1.4 Dynamical selection process 14
3.2 Numerical results 15
3.2.1 Detection rate 16
3.2.2 Consumed energy 17
3.2.3 Node death and DoS detection 20

4 Modeling with Markovian processes and GSPN model **22**
4.1 Modeling using Markov chains 22
4.1.1 Discussion 24
4.2 Non-markovian modeling and verification of DoS 24
4.2.1 Generalized Stochastic Petri Nets 24
4.2.2 Modeling DoS attacks with eGSPN 26
4.2.3 HASL verification of DoS detection models 32
4.2.4 HASL formulae for DoS models 33

5 Energy-based designation of the cNodes **35**
5.1 cNodes selection mechanism 35
5.1.1 Using vNodes to ensure a secured deterministic election . 35
5.1.2 Cluster coverage in case of heterogeneous activity 39
5.1.3 Observations 40
5.2 Selection in practice: results from simulation 40

6 Conclusion **44**

DoS detection in WSNs: Energy-efficient designs and modeling tools for choosing monitoring nodes

Quentin MONNET Lynda MOKDAD

Abstract

The use of wireless sensor networks (WSNs) has increased rapidly over the last years. Due to their low resources, sensors come along with new issues regarding network security and energy consumption. Focusing on the network availability, previous studies proposed to protect clustered network against denial of service attacks with the use of traffic monitoring agents on some nodes. Those control nodes have 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. But if the control nodes (cNodes) die out of exhaustion, they leave the network unprotected. To better fight against attacks, we try to enhance this solution by renewing periodically the election process. Furthermore, we propose two energy-aware and secure methods to designate the cNodes in a hierarchically clustered WSN. The first one is a self-election process where nodes randomly designate themselves. We analyze the trade-offs between *static* and *dynamic* solutions by means of two complementary approaches: through simulation with the ns-2 simulation platform and by means of statistical model checking with the Hybrid Automata Stochastic Logic. The second algorithm for choosing cNodes is purely based on the residual energy of the sensors. We discuss limitations of this deterministic process concerning security and cluster coverage, and suggest workarounds. Again, experimental results from simulation experiments are provided to analyse the energy repartition in the network. All experimental outcomes show improvements of the load balancing in the network, while maintaining good detection coverage.

Keywords: Wireless sensor networks; Reliability, availability, and serviceability; Energy-aware systems; Markovian processes; Model-checking; Petri networks; Simulation.

1 Introduction

Wireless sensor networks *Smart cities* or the *Internet of Things* are foreseen to deeply change people's daily lives. Such projects will interconnect a multitude of devices and bring many functionalities to the end users through an extensive use of sensors. Ambient light, temperature, air pollution degree measurement, or traffic monitoring are just a few examples of applications involving those sensors. There will be sensors everywhere, to gather amounts of data that human beings alone could not measure: sensors deployed as networks can perform constant measuring tasks over wide—and sometimes hard to access—areas.

Such networks are called *wireless sensor networks* (WSNs). The sensors (or *nodes*) are small devices able to gather data on their physical environment. They communicate with one another through radio transmission, but they have low resources at their disposal: limited computing power, limited memory, as well as a limited battery [1, 2]. They are often dropped into hostile areas (by helicopter for instance), or may generally be difficult to access, so the batteries must be considered as single-use. The sensors have to self-organize themselves and to deploy low-consuming routing algorithms so as to create a functional network. All relevant data is typically forwarded to an entity called the *base station* (BS), which does not have the same limitations as the sensors, and acts as an interface between the WSN and the user (or the external world) as displayed on Figure 1.

Wireless sensor networks may be deployed for all kinds of applications, some of them being crucial. For instance there is a lot at stakes when sensor networks are used for watching forest fires. Critical cases also involve all military uses of the sensors: they can be used to detect the presence of biological, chemical or nuclear agent, or to monitor infantry units over battlefields [3]. Such contexts bring strong requirements in terms of security guarantees to the network. Various works deal with ways of preventing unauthorized access to data, or with the necessary precautions to guarantee data authenticity and integrity inside WSNs [4, 5]. But confidentiality as well as authentication are of little 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 trying to prevent a legitimate agent from using a service [6]. Because of the limited resources of their nodes, WSNs tend to be rather vulnerable to DoS attacks. Concrete attacks include jamming the communications, monopolizing the channel ("greedy" attacks) or attempting sleep deprivation on "normal" sensors, for example. They are launched from the outside as well as from the inside of the network: 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 nodes' energy faster. Attacks can target all layers of the network, although we mainly focus here on the Media Access Control (MAC) and routing layers. The problem we tackle 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

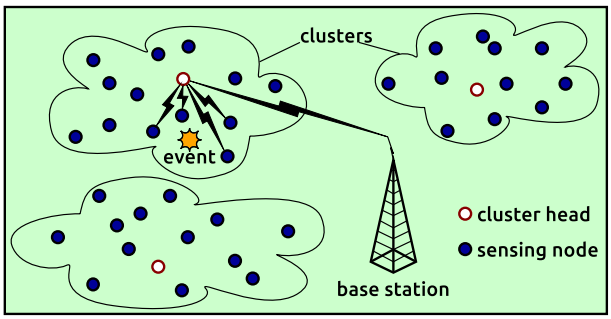


Figure 1: Clustered wireless sensor networks scheme

of compromised 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 data from a sensor to the BS. In a hierarchical WSN, the network is divided into several clusters. The partition is done according to a clustering algorithm such as LEACH [7, 8], HEED [9], or one based on ultra-metric properties [10, 11]. In each cluster, a single common node is designated and becomes 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 [12]. Then they forward their data 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 (see Figure 1). So as to preserve the nodes' energy as long as possible, the network reclustering 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 "subclusters". The CHs from those "subclusters" 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 of this chapter) is a node that 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 [13], 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 [14], we proposed a dynamic approach: cNodes are re-elected periodically (any node in a cluster may be chosen, except the CH) with the elec-

tion period selected to be shorter than that between two network clusterings. Intuitively such a dynamic approach (in comparison to that of [13]), leads to more uniform energy consumption while preserving good detection ability.

Our contribution We propose a dynamic renewal of the designation process of the cNodes. The process itself can be performed by applying different algorithms: nodes can self-elect as cNodes, or can be designated by a central authority (cluster head or base station) depending on several criteria. Two ways to proceed are presented here.

We will first consider a self-election model and address the problem of validating the above conjecture on energy consumption and efficiency by means of modeling techniques. More specifically, this consists in the following aspects:

1. We present a number of numerical results obtained by simulation of DoS detection on WSN models by means of the network simulator ns-2. In particular we simulate models of grid topology WSN including DoS (*static* and *dynamic*) detection policies;

2. 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);

3. We present formal models of the DoS detection mechanisms expressed in terms of Generalized Stochastic Petri Nets (GSPN). In combination with GSPN models we also present a number of performance and dependability properties formally expressed in terms of the Hybrid Automata Stochastic Logic (HASL) [15].

We then propose another designation process which is based on energy, in order to obtain an even better load balancing. We propose to designate the sensors for the cNode position according to their residual energy, but we show that several problems occur with deterministic election. Indeed, compromised nodes could see a flaw to exploit in order to take over the cNode role and decrease the odds of being detected by announcing high residual energy. We address this issue by introducing a second role of surveillance: we choose "vNodes" responsible for watching over the cNodes and for matching their announced consumption against mathematical model. We also recommend that every node in the cluster be monitored by at least one cNode, to prevent all the cNodes to be elected inside the same spatial area of the cluster at each election iteration. Once again, simulation results indicate a better load balancing.

Chapter structure The remainder of this chapter is organized as follows: in Section 2 we give an overview of DoS attack detection for cluster-based WSNs. Section 3 presents the self-election method: it includes network topology and protocols introduction (subsection 3.1) and simulation results (subsection 3.2). In Section 4 we use modeling tools to represent our network under attack: we

provide the structure of Markov chains for modeling our WSN (subsection 4.1), as well as the application of statistical model checking performance analysis to Petri Nets models of attacked networks (subsection 4.2). Section 5 proposes a second way to designate the nodes by using their residual energy (subsection 5.1), coming with associated results from simulated experiments (subsection 5.2). Finally Section 6 permits us to sum up our contribution and to consider future work leads.

2 Related work

This section is divided into three parts: security in wireless sensor networks, denial of service specific mechanisms, and clustering algorithms and energy preservation.

2.1 Security in WSNs

Denial of service is not the only type of attack a WSN should resist to. Security in general in sensor networks has attracted quite a lot of interest during the last few years. Hence it has been the subject of many studies in literature, as well as several state-of-the-art articles [16, 17].

Confidentiality and integrity must be ensured to prevent attackers access to or tampering with sensitive data. A number of solutions have been proposed [8], many of them involving strong [4] and/or homomorphic [18] cryptography, some relying on other mechanisms such as multi-path based fragmentation of the packets [19] or game theory [20].

Authentication brings to participants the guarantee that the peer they are communicating with truly is what it pretends to be; that is another important point. It has been deeply investigated as well [21]. Many lightweight proposals for key management in WSNs have been suggested [22, 23].

Apart from those, there have been a variety of proposals to secure other elements, on a basis than any information about any aspect of the network might be valuable to an attacker. Hence there are approaches, for instance, to secure the geographical location of the nodes through epidemical information dissemination [24] as well as through more conventional mechanisms [25].

2.2 DoS-specific mechanisms

Denial-of-service attacks embrace many different attacks, which can target all layers of the network [26]. Jamming the radio frequencies as well as disturbing the routing protocols are just two examples of ways to harm the network. In reaction to these, a number of solutions have been proposed [27]. As stated in the introduction, we focus in this paper on inside attackers attempting to bend the MAC protocol parameters to their needs, be it to achieve better performances for themselves (greedy attacks) or to generally harm the network (jamming attacks or sleep deprivation). To detect such attackers, many solutions rely on

trust models [28, 29] with agents applying a set of rules [30] on traffic to attribute a trust value to each of the nodes in the network. Below are outlined some notable proposals.

Back in 2001, most work focused on making WSNs feasible and useful. But some people already involved themselves into security. For instance, SPINS (Security Protocols for Sensor Networks) was proposed in [31] 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 μTESLA ("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.

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, it is proposed in [32] 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.

Some works examine the possibility of detecting the compromising of nodes as soon as an opponent physically withdraws them from the network. In the method that is developed in [33], each node keeps a watch on the presence of its neighbors. The Sequential Probability Radio Test (SPRT) is used to determine 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 [34], a revised version of the OLSR protocol is proposed. 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. The authors also use 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.

A novel broadcast authentication mechanism can also be deployed so as to cope with DoS attacks in sensor networks such as in [35]. 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.

Much of our work relies on the work of Lai and Chen who proposed in [13] a system detection based on static election of a set of nodes called "guarding nodes" which analyze traffic in a clustered network. When detecting abnormal traffic from a given node, "guarding nodes"—we call them cNodes—identify it as a compromised node and inform the cluster head of it. On reception of reports from several distinct cNodes (to prevent false denunciation from a compromised node), the CH virtually excludes the suspicious node from the cluster. The authors show the benefit of their method by presenting numerical analysis of detection rate. Although the method is efficient for detecting rogue nodes, the authors do not give details of the election mechanism for choosing the cNodes. Also, there is no mention in their study of renewing the election in time, which causes the appointed cNodes to endorse heavier energy consumption on a long period.

2.3 Clustering algorithms and energy preservation

A lot of approaches intended to bring security into a WSN are cluster-based [36]. But the main purpose of clustering a sensor network usually resides in scaling possibilities, improved nodes management and energy savings brought by partitioning. Several clustering algorithms have been proposed [37]. They generally aim at determining which nodes in the network will be the cluster heads, often basing the choice on energetic considerations. Basically, choosing a cluster head in a network is not so different than selecting cNodes in a cluster. But in the latter case we have some additional constraints on security.

One of the easiest clustering algorithms to implement, and probably one of the most used, is the LEACH algorithm [38].

2.3.1 LEACH functioning

LEACH is likely one of the easiest algorithm to apply to recluster the network. It is a dynamical clustering and routing algorithm. 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 $T(i)$:

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

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 CHs 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.

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.

2.3.2 LEACH improvements

There are a number of proposals derived from LEACH, to improve either its efficiency [39, 40] or its security. In [41], the authors propose to add security mechanisms via a revised version of LEACH protocol. SecLEACH uses random key predistribution as well as μTESLA (authenticated broadcast) so as to protect communications. But the authors do not mention any mechanism to fight DoS attacks.

In [42], the authors propose another way to secure the LEACH protocol against selfish behaviors, using elements from game theory. 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.

Other algorithms Other possible clustering algorithms include HEED [9], 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.

Note that, aside from clustering, the importance of energy issues in WSNs has led to proposals of several mechanisms to cut down its consumption [43], based for example on packet priority [44].

3 Pseudo-random self-election of the cNodes

3.1 Detection of DoS attacks

3.1.1 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.

3.1.2 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 subclusters. For more clarity, we will call 1-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 [7, 8], HEED [9], algorithms based on ultra-metric properties [12], *et cætera*. 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 send packets to 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. An example 2-clustered network is displayed on Figure 2.

k-LEACH Once a clustering algorithm has been applied to the network 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 the LEACH algorithm k times recursively. We call those recursive iterations the k-LEACH algorithm. In practice, we had k equal to 2, for the following reasons:

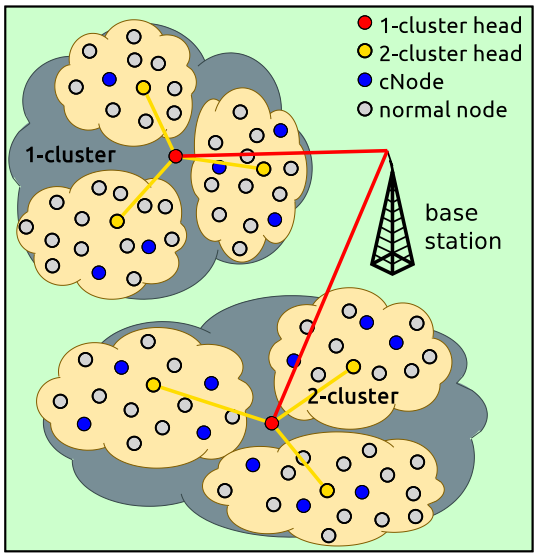


Figure 2: Scheme of a twice clustered WSN

- 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 maximized the cover area and the probability of detecting compromised nodes.

3.1.3 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 (see also Figure 3). The cNodes—for control nodes—were introduced in [13] to analyze the network traffic and to detect any abnormal behavior from other nodes in the cluster.

Control nodes watching over the input traffic allow the detection of various types of denial of service attacks. This is achieved with agents running on the cNodes and applying specific rules on overheard traffic [30]. Each rule is used to fight against one (sometimes a few) specific attack(s): jamming, tampering, black hole attacks, and so on. Each time a cNode notices that a rule is broken by a node, it raises a bad behavior for this node, and send an alert to the cluster head. Following are some example rules:

- Rate rule: assuming that minimal and maximal rates for data each node sends are enforced, a bad behavior will be reported if those rates are not respected. With this rule, monitoring agents should be able to detect negligence (if minimal rate is not reached) or flooding (if maximal rate is exceeded) attacks.

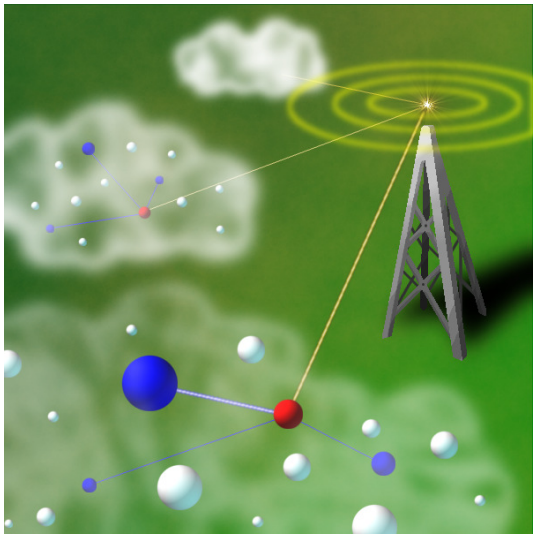


Figure 3: Cluster-based sensor network with cNodes

- Retransmission rule: a cNode overhearing a packet supposed to be retransmitted by one of its neighbor (the neighbor node is not the final destination for this packet). If the concerned neighbor does not forward the packet, it may be undertaking a black-hole (full dismissal of packets) or a selective forwarding attack.
- Integrity rule: a bad behavior will be raised if a neighbor of the node running the monitoring agent tampers with a packet before forwarding it. Applying this rule assumes that the nodes are not expected to proceed either to data aggregation or compression before forwarding.
- Delay rule: forwarding a packet should not exceed a threshold delay.
- Replay rule: a message should be sent no more than a limited number of times.
- Jamming rule: an unusually high number of collisions (compared to average, or concerning only some nodes) may be related to the presence of a jamming node. If jamming is done with random noise, without legitimate packets containing a node identifier, it may be difficult to detect the source of it, but several cooperating agents should be able to detect it.
- Radio transmission range rule: a node sending messages with an unexpectedly high power may be trying to launch a hello flood (it tries to appear in the neighbor list of as many nodes as possible) or wormhole attack (it redirects a part of the overheard traffic to another part of the network). Hence it may be consider as a bad behavior.

In the rest of this study, we will not describe in details each one of the mentioned attacks, nor will we detail the associated solutions to counter them. When details are needed, we will consider only one example: flooding attacks. The model of a flooding attack is the following: a malicious node sends a high amount of data to prevent legitimate nodes from communicating by saturating the medium, or by establishing too many connections with the receiver node [45]. In wireless sensor networks, it is also used to drain the energy of neighbor nodes.

So 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 rule is broken. In that case the cNode sends a warning message to the CH. 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 signaling as an actual anomaly. Once the CH has received warnings from a sufficiently large number of distinct cNodes 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. First, each node has to compute a pseudo-random number, which may not be necessary with other methods. Second, 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 proceeds 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 implemented in our ns-2 simulation whose outcomes will be discussed in subsection 3.2. It is also the method we consider in all Section 4 for modeling.

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 or energy. Thus the clear advantage of BS-centralized election is that all costly operations (*i.e.*, pseudo-random numbers calculation) can be reiterated in a (virtually) unconstrained environment (*i.e.*, the BS) This technique is explained in detail in [14].

From a robustness point of view note that this 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 of 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 a 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, and 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.

3.1.4 Dynamical selection process

The dynamical renewing of the selection process is an essential part of our proposal. Many of the recent intrusion detection systems proposed for WSNs tend to be lightweight, to consume little energy. We believe that a dynamical renewing of the selected cNodes helps a lot to balance the load inside the cluster. Depending on the application running in the network, maybe this balancing is not worth the constraints induced by periodical re-election, but generally energy preservation is a priority in WSNs and distributing the consumption among all the nodes helps to maintain the highest possible amount of nodes in activity for as long as possible. Also, lightweight IDSs themselves may be designed to minimize the disparities in energy consumption inside a network, but we argue that with a system as simple as the cNodes, simulation results indicate that the savings are not negligible.

A second thing to consider is that the recursive clustering as well as the dynamic renewing of the monitoring nodes can be used with other detection systems than the cNodes we use here. If an IDS is good at preserving energy and balancing nodes, but needs to be run only by a subset of the nodes in the network, dynamical selection processes presented in this work can be applied so as to select the sensors which will run the system (provided the monitoring sensors do not need any specific hardware that would differentiate them from the “normal” nodes).

3.2 Numerical results

In an attempt to validate the efficiency of the proposed method, we have developed an implementation of an example WSN by means of existing simulative framework, the ns-2 Network Simulator. In this subsection, 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 4):

- grid is a square of size a;
- cluster head is placed at the centre of the grid (*i.e.*, red node in Figure 4);
- the grid contains 100 (sensing) nodes regularly displaced;
- 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 4—can reach a circle of radius $a\sqrt{2}/2$. In this way all nodes, included corners, can reach the CH). No power adjustment is done by the nodes for transmission.

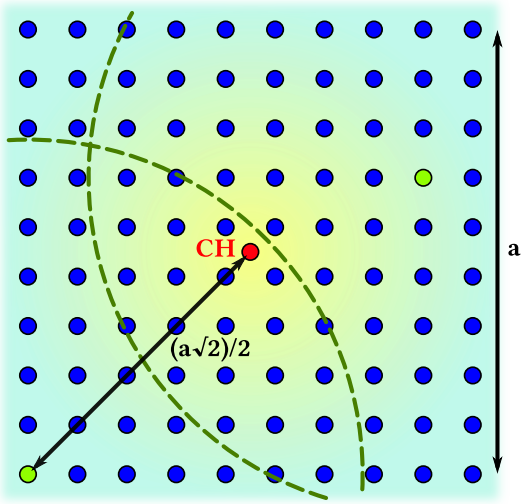


Figure 4: A 10×10 regular-grid cluster of size a

In such network cNodes (represented in green in Figure 4) are elected periodically either using the static approach or using the dynamic election mechanism described in previous subsections. We have designed our experiments focusing on two performance measures: the rate of detection of attacks and the overall energy consumption. Table 1 reports about the (range of) parameters considered in our simulation experiments.

Table 1: Simulation parameters

Parameter	Value
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

3.2.1 Detection rate

In order to evaluate the considered performance measure, namely attack detection rate, we have considered the parameters given in Table 1. We have assumed that the traffic generation follows a Poisson distribution with rate λ. This rate is low (10 kbit/s) for normal nodes. Compromised nodes are trying to flood the network; hence they send numerous messages in order to saturate the medium and/or to exhaust the resources of the other nodes. Their transmission rate is much higher, and was set at 800 kbit/s. In the experiments we have considered a cluster with 100 nodes.

Figure 5 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.

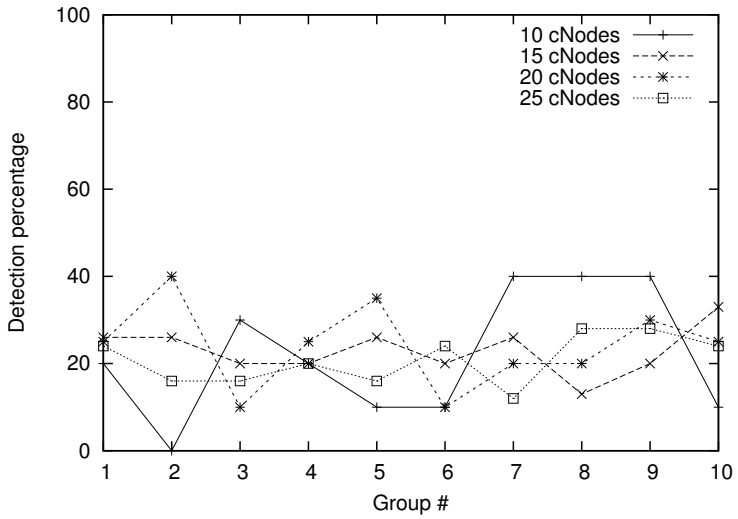


Figure 5: Detection versus group

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 remains similar, which suggests that we do not need to use more nodes than 15 nodes in each 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 6 below for an average of 10 compromised nodes.

In Figure 6 we notice that, as the average transmission of attacking nodes increases, our dynamic solution detects more attacks than the static solution.

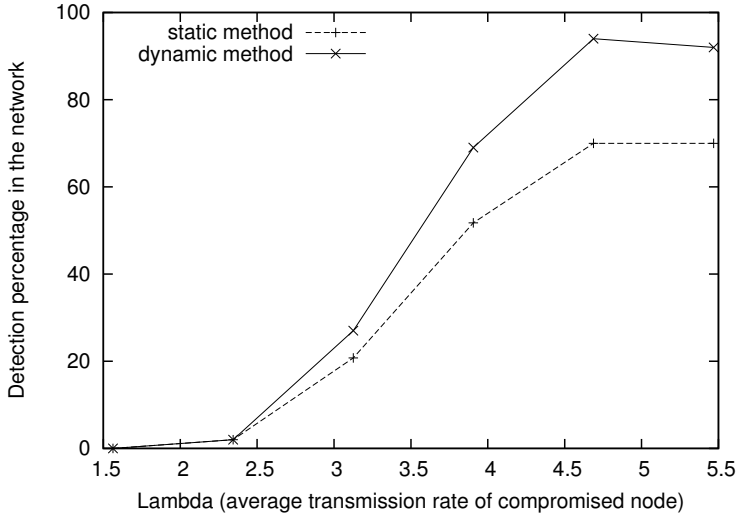


Figure 6: Detection versus lambda

3.2.2 Consumed energy

All the simulations which were run to produce the results presented in this subsection used the parameters given in Table 2.

Table 2: Simulation parameters

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

Figure 7 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).

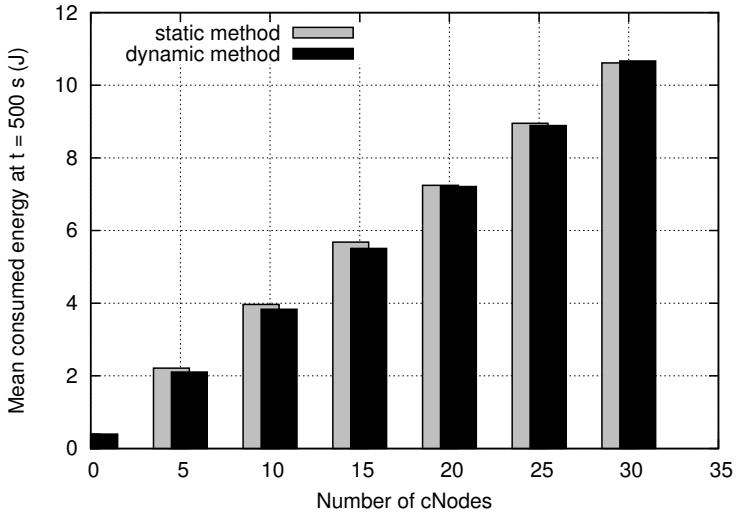


Figure 7: Average energy consumption

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

Figure 8 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 Figure 9, 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. A 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.

As the cNodes are re-elected and the consumption is distributed for the dy-

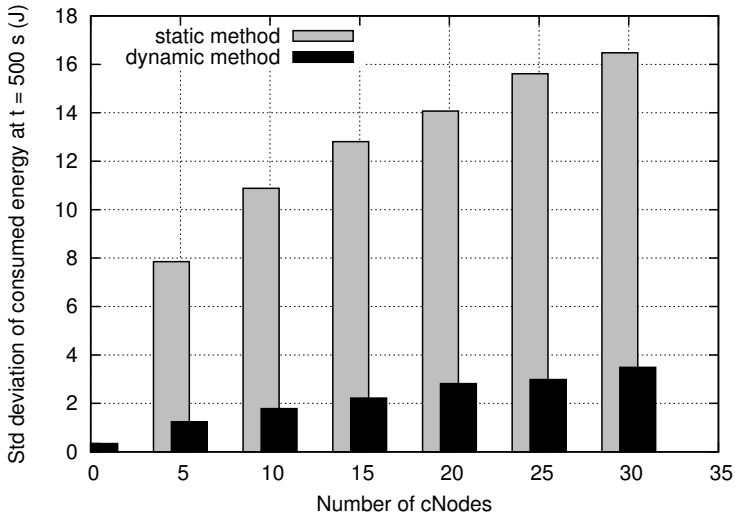


Figure 8: Energy consumption standard deviation

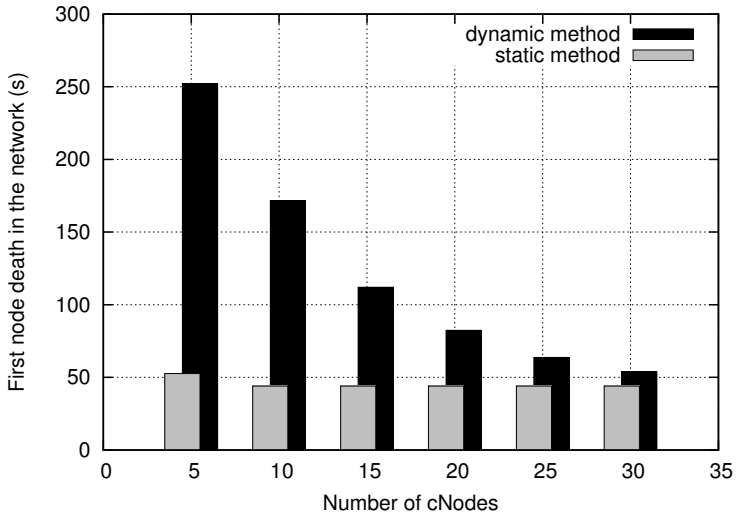


Figure 9: First death in the network

namic 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.

3.2.3 Node 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 3.

Table 3: Simulation parameters

Parameter	Value
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

Figure 10 shows the evolution of the number of alive nodes in time. As for the previous subsection, 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 don't). In the static method, elected cNodes consume their battery power, and die (at about

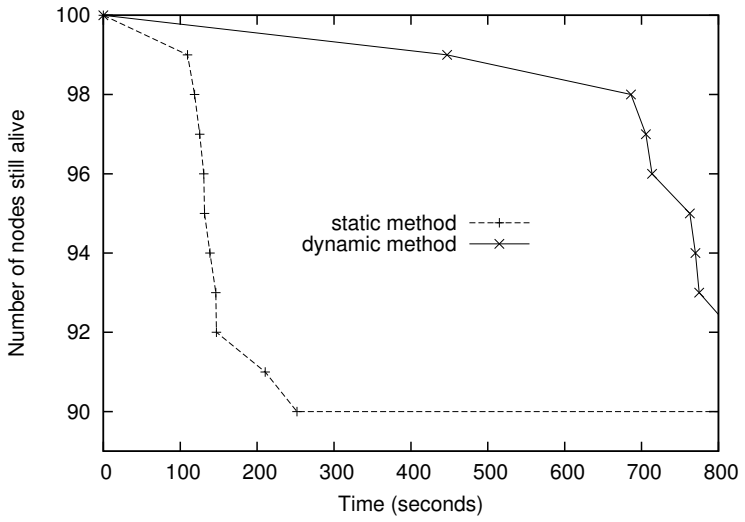


Figure 10: Nodes remained alive

$t = 150$ 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 curve above.

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 Figure 11. The average number of cNodes which detected the attack (out of 10 cNodes) is presented for each 60 second-long period.

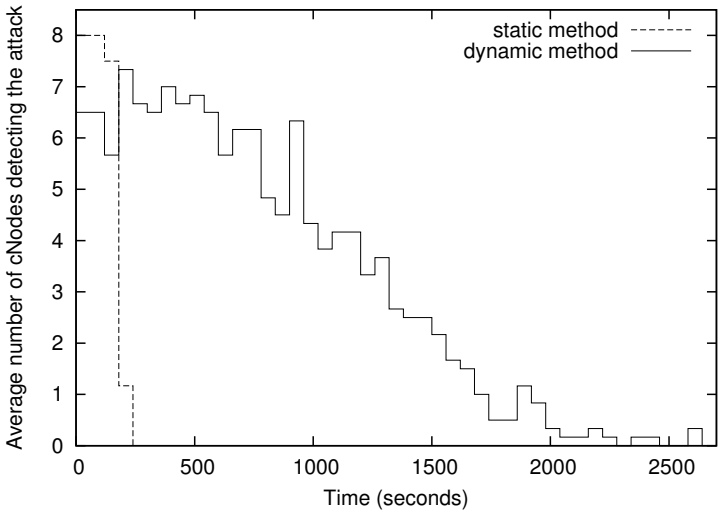


Figure 11: DoS detection

After the fourth minute, every cNode is dead for the static method, and the compromised node is no longer 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 election. The flooding sensor is still detected by more than one node after half an hour.

Simulations are a fine way to obtain results from a proposed algorithm. But they do not have the rigor of mathematical models. In next section we will attempt to model our network under attack using formal verification tools.

4 Modeling with Markovian processes and GSPN model

4.1 Modeling 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 in which the future evolution depends exclusively on the current state (and not on the history that led into it). It is well known that in order to fulfill the Markov property, delay of events must be exponentially distributed.

In this subsection we describe how to structure Continuous Time Markov Chains (CTMC) models for modeling 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 $\lambda_c >> \lambda_i$.
- Each cNode 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 modeled in terms of a $K \cdot (N+1)$-dimensional CTMC. States of such a CTMC consist of K-tuples $x = (x_1, x_2, \ldots, x_K)$ of macro-states $x_k = (x_{k_1}, x_{k_2}, \ldots, x_{k_N}, x_{k_d})$ 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* $f : \mathbb{N}^N \to \{0, 1\}$ which is used (by cNodes) to decide whether traffic rate has exceeded the "normal" threshold. The arguments of f are an (N)-tuples $(n_1, \ldots n_N)$, 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 $x = (x_1, x_2, \ldots, x_K)$ 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.

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

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:

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

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

4.1.1 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 of deterministic distributions). We discuss non-Markovian modeling of DoS detection mechanisms in subsection 4.2.

4.2 Non-markovian modeling and verification of DoS

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 are that these processes allow for representing generally distributed durations, rather than, as with CTMC, being limited to exponentially distributed events.

In this subsection we present a modeling approach of DoS detection in terms of Generalized Stochastic Petri Nets (GSPN) [46], a class of Petri nets suitable for modeling stochastic processes. By definition, the GSPN formalism is a high-level language for representing CTMC. 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 DESP. Furthermore, eGSPN is also the formal modeling language supported by the COSMOS [47] statistical model checker, a tool that allows for verification of (sophisticated) performance measures in terms of the Hybrid Automata Stochastic Logic (HASL) [15].

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.

4.2.1 Generalized Stochastic Petri Nets

A GSPN model is a bipartite graph consisting of two classes of nodes, *places* and *transitions* (Figure 12). 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

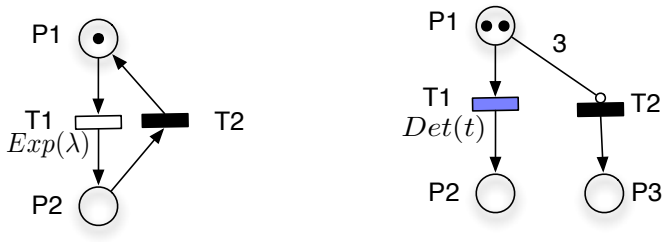


Figure 12: Simple examples of eGSPN: timed-transitions, immediate transition and inhibitors arcs

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 the left-hand part of Figure 12 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 an 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 12). 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, which 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 $\mathrm{Exp}(w)$ distributed delay), an 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, *et cætera*), 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[1].

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 on the left-hand side of Figure 12) and *deterministically distributed* timed-transitions (denoted by blue-filled-in bars, *e.g.,* T1 on the right-hand side of Figure 12). 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

[1] a possible condition in case of non-continuous delay distribution

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 12). In the presence of inhibitor arcs the semantics of the GSPN firing rule is slightly modified, thus: a transition is enabled whenever all of its input places contain 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 the right-hand part of Figure 12 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.

4.2.2 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 3×3 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.

GSPN model of sensing nodes Sensing nodes functionality is trivially simple: they simply keep sending sensed data packets at a pace which (following subsection 4.1) we assume being exponentially distributed, with rate λ_i. This can be modeled by a simple 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$_{i_j}$" in Figure 13). Note that transition TX in Figure 13 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

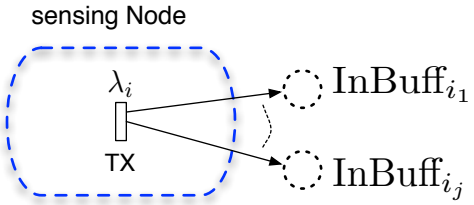


Figure 13: GSPN model of a sensing node

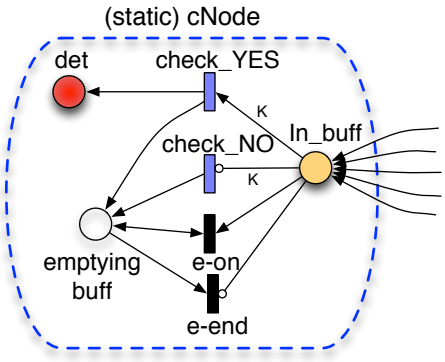
time with $\delta_s \sim \mathrm{Exp}(\lambda_i)$. To summarize: the sensing functionality of a specific node in WSN is modeled by a single timed-transition provided with as many outgoing arcs as the number of neighbors of that node. The complete sensing functionality of a WSN can be modeled by combining several such GSPN modules.

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. From a modeling point of view, a distinction must be made between the case of statically elected cNodes (as in [13]) and that of dynamically eligible cNodes (as in [14]). 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.

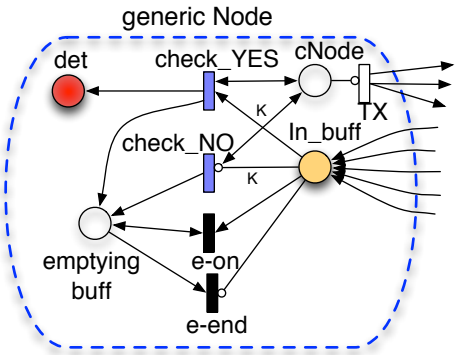
GSPN models for both *static* and *dynamic* cNodes are depicted in Figure 14(a) and Figure 14(b), respectively. A cNode detects an attack whenever the overheard traffic throughput (*i.e.*, number of overheard packets per observation period) exceeds a given threshold ρ_{attack}. Place "InBuff" (Figure 14(a)) represents the input buffer of a node, where packets received/overheard from neighbor 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 14(a) and Figure 14(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 a DoS detection, otherwise (*i.e.*, "checkNO" enabled) no tokens are 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 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 14(b)) is a simple extension of that for static cNodes obtained by adding an auxiliary place "cNodes" and an auxiliary exponentially distributed timed-transition "TX". This is needed



(a) GSPN model for a *statically elected* cNode



(b) GSPN model for a *dynamically eligible* cNode

Figure 14: GSPN components representing cNodes behavior in a WSN with DoS detection mechanisms

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 14(b) behaves exactly as that of Figure 14(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 14(b) behaves exactly as that of Figure 13).

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 3×3 WSN

grid-topology equipped with DoS detection functionalities.

GSPN model of DoS detection with static cNodes Figure 15 illustrates a complete GSPN model for a 3×3 grid topology representing an example of DoS detection with static election of cNodes (as in [13]). 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.*,

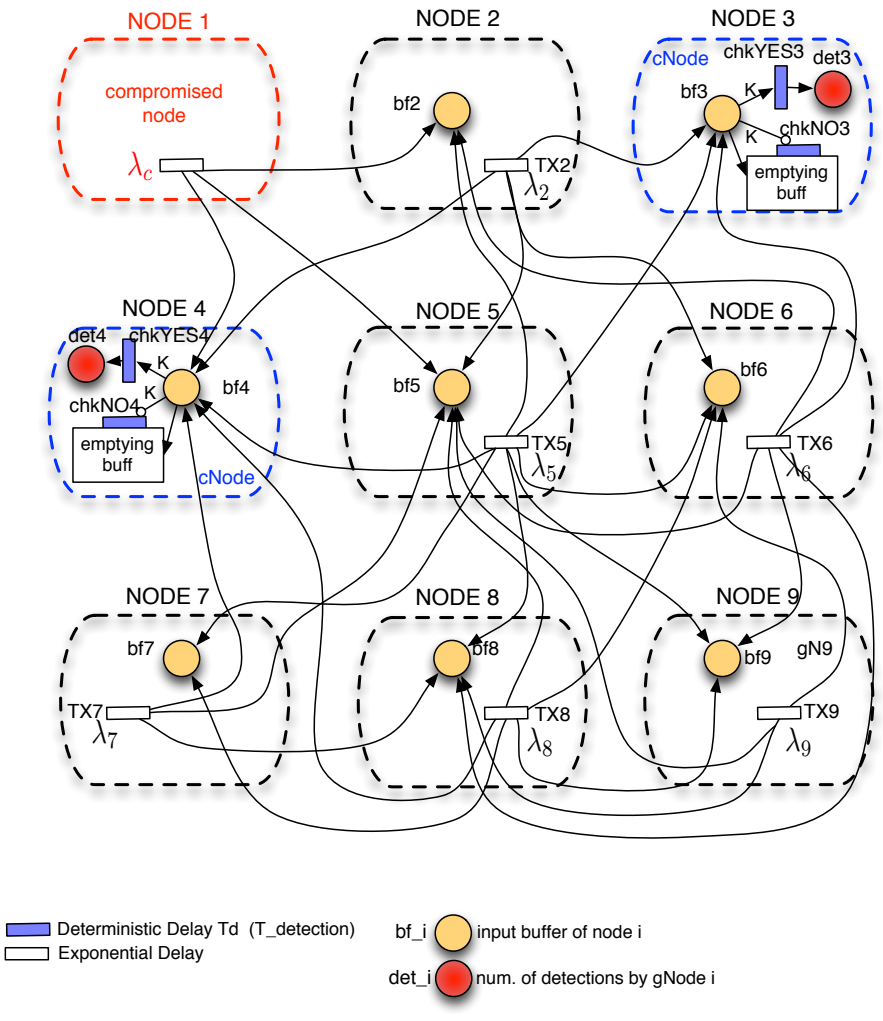


Figure 15: GSPN model of a 3×3 grid-topology with 1 (fixed) compromised node and 2 static cNodes

the content of that box corresponds to the subnet responsible for emptying the "inBuff" place as depicted in Figure 14(a) and Figure 14(b)).

This model can be used to study the performances of DoS detection with static cNodes in many respects, such as: measuring the expected number of detected attacks within a certain time bound, or also, for example, assessing the average energy consumption of cNodes. In the next subsection 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.

GSPN model of DoS detection with dynamic cNodes Figures 16 and 17 illustrate the GSPN model of a 3×3 grid topology for the case of DoS detection with dynamic election of cNodes (as in [14]). For simplicity the model has been split into two parts: the actual network topology part (Figure 16) and the cNodes random election mechanism (Figure 17). The network model (Fig-

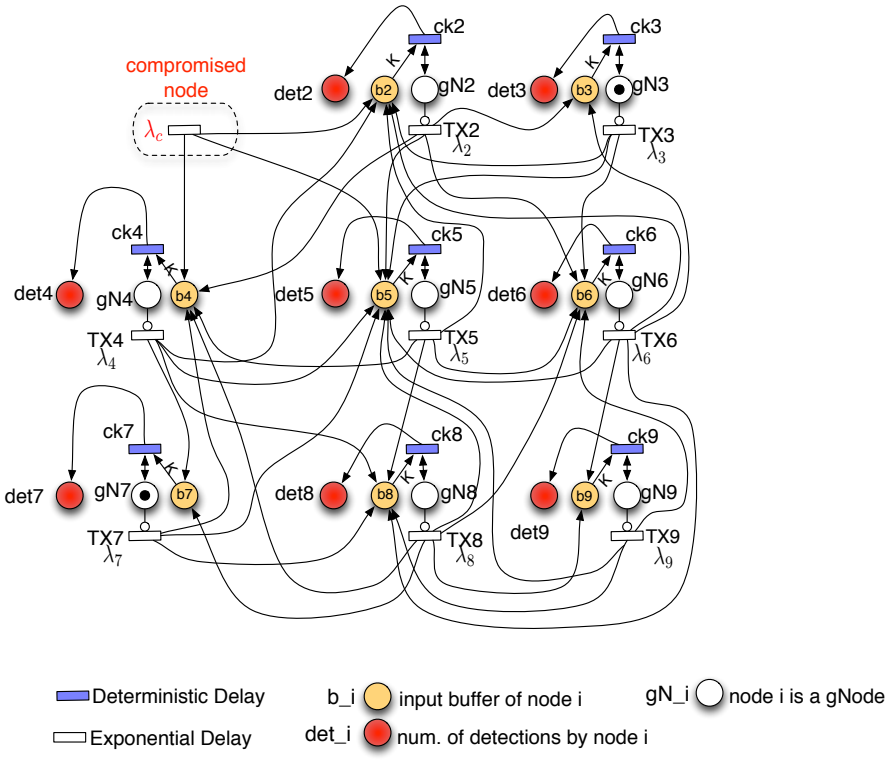


Figure 16: GSPN model: the traffic part in a 3×3 topology with 1 (fixed) compromised node and 2 randomly elected cNodes

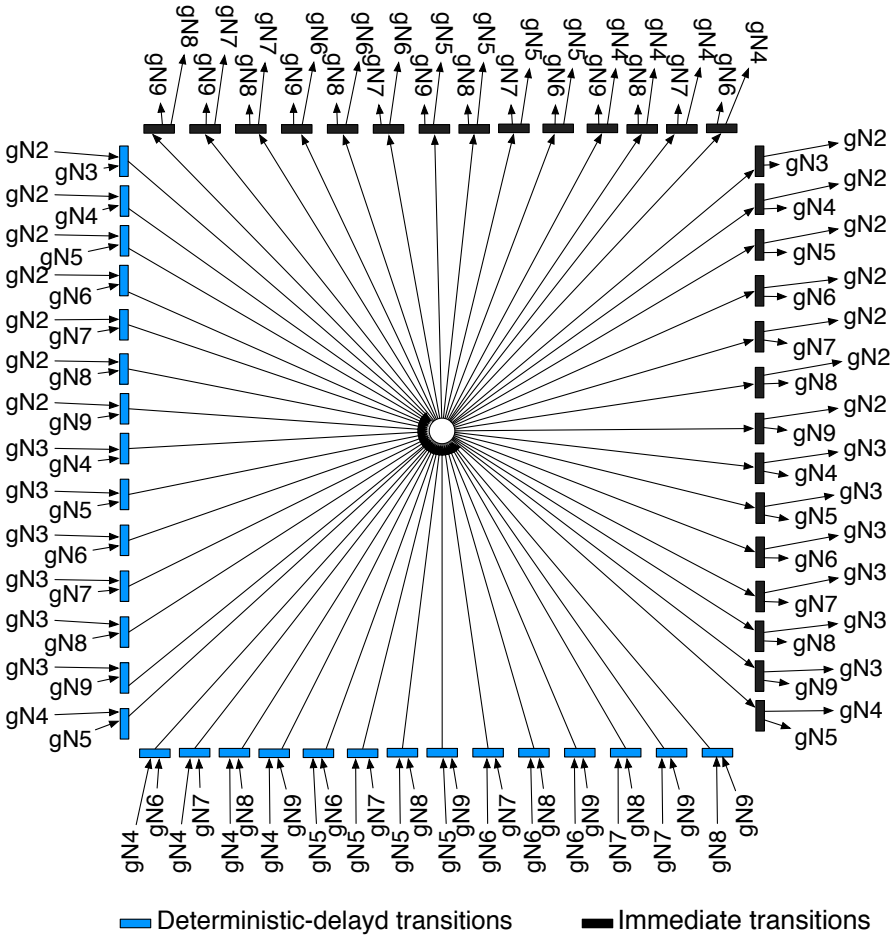


Figure 17: GSPN model: the random election policy part: 2 cNodes are elected out of 8

ure 16) 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 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 14(b)). The cNodes election component (Figure 17), 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 17 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 17. 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.

4.2.3 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 [15] for formal details.

HASL model checking Model checking 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 [48]) 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 timed executions 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 following syntax and which represent the measure to be assessed.

$$Z ::= \mathrm{E}(Y) \mid Z + Z \mid Z \times Z$$
$$Y ::= c \mid Y + Y \mid Y \times Y \mid Y/Y \mid \mathrm{last}(y) \mid \min(y) \mid \max(y) \mid \mathrm{int}(y) \mid \mathrm{avg}(y)$$
$$y ::= c \mid x \mid y + y \mid y \times y \mid y/y$$

The informal meaning of an HASL expressions Z 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.*, $\mathrm{last}(y), \min(y), \max(y), \mathrm{int}(y), \mathrm{avg}(y)$) on top of which a path variable Y is built have intuitive meanings. In particular: $\mathrm{last}(y)$ indicates the last value of expression y along an accepted synchronized path; $\min(y)/\max(y)$ indicates the minimum (maximum) of y along a path; $\mathrm{int}(y)$ the integral of y along a path; $\mathrm{avg}(y)$ the average of y along a path.

The HASL statistical model checking procedure works as follows:

- It takes a GSPN model and an HASL formula
- It iteratively generates 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.

4.2.4 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 subsection. 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:

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

The LHA in Figure 18 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 15, Figures 16 / 17). It consists of 2 locations and refers to the 4 data-variables described above. 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

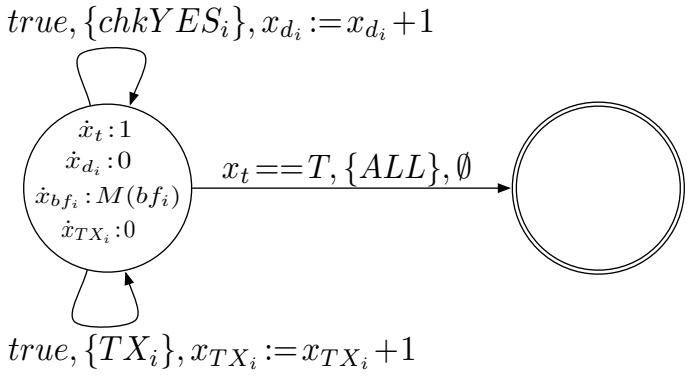


Figure 18: An LHA for assessing relevant measures of DoS GSPN models

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_{bf_i} is incremented with rate proportional to the number of tokens in the input buffer of cNode i (*i.e.*, $\dot{x}_{bf_i} = M(bf_i)$); 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 counter variables x_{d_i} and x_{TX_i} on occurrence of the associated events in the GSPN model. For example transition $l_1 \xrightarrow{true, \{chkYES_i\}, x_{d_i} := x_{d_i}+1} l_1$ indicates that, 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 stops 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 18 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 18 to evaluate relevant measures of the DoS GSPN models.

- $Z_1 \equiv \mathrm{E}(\mathrm{last}(x_{d_i}))$: the expected number of detected attacks by cNode i after T time units
- $Z_2 \equiv \mathrm{E}(\mathrm{last}(x_{d_i} + x_{d_{i'}}))$: the sum of attacks detected by cNode i and i' after T time units
- $Z_3 \equiv \mathrm{E}(\mathrm{last}(x_{TX_i}))$: the expected value of packets transmitted by node i

after T time units

- $Z_4 \equiv \mathrm{E}(\mathrm{avg}(x_{bf_i}))$: the expected cumulative flow of packets received by node i within T time units

5 Energy-based designation of the cNodes

5.1 cNodes selection mechanism

Electing the cNodes is not an easy task. In subsection 3.1.3 we exposed and compared three ways to elect them:

- pseudo-random election by the base station;
- pseudo-random election by the cluster head;
- pseudo-random election by the nodes themselves.

We assumed that election should be random so that compromised nodes would not be aware of which node could control the traffic. We did not consider the remaining energy during the cNodes election. But monitoring the traffic implies to keep listening for wireless transmission without interruption. Hence cNodes will have a greater energy consumption than normal nodes. Given that preserving energy is an essential issue in the network, we now prefer to ensure load balancing rather than assuring a pseudo-random election, and thus to consider the residual energy of the nodes during the election. This choice also raises new issues and makes us define a new role for the nodes in the cluster[2].

5.1.1 Using vNodes to ensure a secured deterministic election

The issue with energy measurement is that no agent in the network is able to measure the residual energy of a given node N, but the node itself. The neighbor nodes of N may record messages sent from N and compute a rough estimate, but as they know neither the initial amount of energy of N (at the network deployment) nor the energy N spent for listening, estimates can not be used to obtain values precise enough so as to reliably sort the nodes according to their residual energy.

So the only way to get the residual energy of a node is to ask this node. The election algorithm we propose is described as follows:

1. During the first step, each node evaluates its residual energy and sends the value to the cluster head;

[2]The recursive k-clustering is used in this section in the same way as in former section. And yet for simplicity we will only mention "clusters", as the solution is not dependent of the depth of recursive clustering.

2. Having received the residual energy of all nodes in the cluster, the cluster head picks the n nodes with the highest residual energy (where n is the desired number of cNodes during each cycle) and returns them a message to assign them the role of cNode.

It is a deterministic selection algorithm that eliminates any random aspect from the process. The rule is simple: nodes possessing the highest residual energy will be elected. Given that the cNode role implies consuming more energy (cNodes listen to surrounding communications most of the time), rotation of the roles is theoretically assured. But the deterministic aspect is also a flaw that may be exploited by compromised nodes. This is a crucial issue: we can not neglect compromised nodes as the whole cNodes mechanism is deployed in the sole purpose to detect them!

More precisely, the problem may be stated as follows. Compromised nodes will be interested in endorsing a cNode role, as it enables them:

- to reduce the number of legitimate cNodes able to detect them;
- to advertise the cluster head about "innocent" sensing nodes to have them revoked.

When a pseudo-random election algorithm is applied, a compromised node (or even several ones) can be elected during a cycle, but it will loose its role further in time, for later cycles. Even with a self-election process, compromised nodes can keep their cNode role as long as they want, but they can not prevent other (legitimate) nodes to elect themselves, too. With deterministic election, however, they can monopolize most of the available cNode roles. They only have to announce the highest residual energy value at the first step of the election to get assured to win. If there are enough compromised nodes to occupy all of the n available cNode roles, then they become virtually immune to potential detection.

To prevent nodes from lying when announcing their residual energy, we propose to assign a new role to some of the neighbors of each cNode. Those nodes—we call them vNodes, as for *verification* nodes—are responsible for the surveillance of the monitoring nodes. Once the cNodes election is over, each neighbor to a cNode decides with a given probability whether it will be a vNode for this cNode or not. A given node can act as a vNode for several cNodes (in other words, it can survey several neighbor cNodes).

If this role consumes too much energy, it is not worth deploying vNodes: we should rather use pseudo-random election for the cNodes. So vNodes must not stay awake and listen most of the time, as cNodes do. Instead they send, from time to time, requests to the cNode they watch over, asking it for its residual energy. They wait for the answer, and keep the value in memory.

Once they have gathered enough data, vNodes try to correlate the theoretical model of consumption of the cNode they survey and its announced consumption, deduced from broadcast messages (during elections) and answers to requests from vNodes. Four distinct cases may occur:

1. The announced consumption does not correlate (at all) with the theoretical model: there is a high probability the node is compromised and seeks to take over cNode role. It is reported to the cluster head.

2. The announced consumption correlates *exactly* with the theoretical model: the node is probably a compromised node trying to get elected while escaping to detection (in other words, the rogue cNode adapts its behavior regarding to the previous point). It is easy to detect the subterfuge as values received from the rogue node and the ones computed by the vNodes are exactly the same. It is reported to the cluster head.

3. The announced consumption correlates roughly with the theoretical model, but does not evolve in the same way (with regard to the model) as the real consumption locally observed by the vNodes (local (in time) evolution of the announced consumption does not "stick" to the one of the surrounding vNodes, which should roughly rise or decrease during the same periods). The node is probably compromised, trying to escape detection by decreasing its announced energy with random values. It is reported to the CH.

4. The announced consumption correlates roughly with theoretical model, and evolves in the same way as the traffic observed by vNodes. Whether the node is compromised or not, it has normal behavior and is allowed to act as a cNode.

If a given vNode is in fact a malicious node, it could lie about integrity of the cNode it watches. To prevent that, the cluster head must receive multiple reports (their number exceeding a predetermined threshold) from distinct vNodes before actually considering a cNode as compromised. To some extent, this also makes the scheme resilient to errors from the vNodes.

In that way, nodes are allowed to act as cNodes only if they announce plausible amounts of residual energy. Assuming that this role consumes more energy than sensing only, the nodes elected as cNodes will sooner or later see their residual energy drop below the reserve of normal sensing nodes, which implies that they will not get re-elected at the next election. Note that the cases 2 and 3 make a compromised node decrement its announced energy as the time goes by. Even if inconsistency may be noticed and the compromise detected, this simple behavior ensures that the rogue node will stop being elected at some point in time.

Thus, the interest of vNodes can be summarized as follows: a compromised node cannot ensure the takeover of the cNode role at each cycle without cheating when announcing residual energy, and hence being detected by the vNodes. Detecting rogue cNodes, or forcing them to give up their role for later cycles, are the two purposes of the vNodes. The vNode role does not prevent a node from processing to its normal sensing activity (requests to cNodes must not occur too often, or too much power will be drained from the vNodes). The state machine of the nodes is presented in Figure 19.

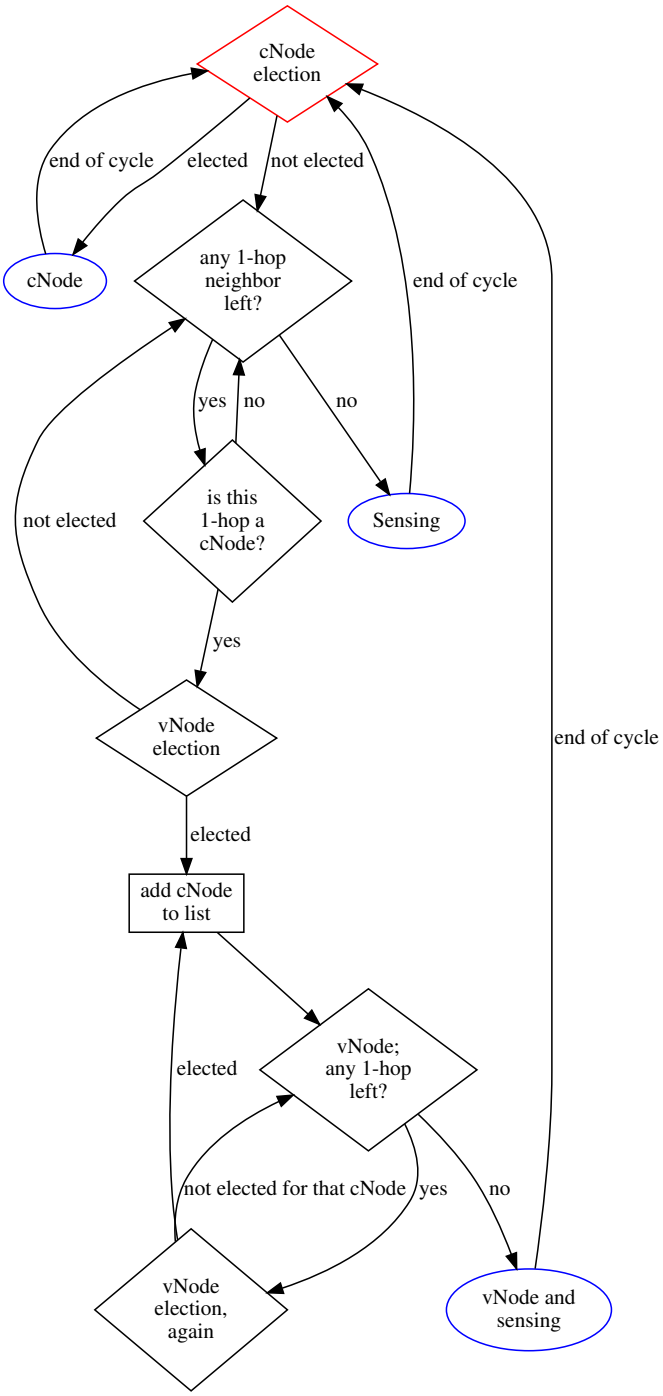


Figure 19: State machine of the (non-CH) nodes

5.1.2 Cluster coverage in case of heterogeneous activity

Deterministic election of the cNodes does not only introduce a flaw that compromised nodes could try to exploit. There is a second problem, independent from the nodes' behavior, which could prevent the detection of compromised nodes. If a region of the network happens to produce more traffic activity than the other parts of the network, the energy of its nodes will be drawn faster. In consequence, none of the n nodes with the highest residual energy (n being the desired number of cNodes during each cycle) will be located inside this region, and some nodes may not be covered for surveillance as long as traffic does not fade, possibly for all cycles. Figure 20 illustrates this problem.

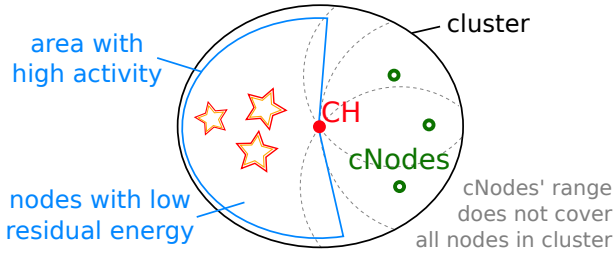


Figure 20: Illustrative scheme: cNodes are elected inside the area with less activity (thus with more residual energy) and do not cover nodes from the opposite side of the network

To address this issue we need to ensure that every node in the network is covered by at least one cNode. So the election process we presented in subsection 5.1.1 needs to be modified. The correct version is as follows:

1. During the first step, each node evaluates its residual energy and broadcasts the value;
2. The cluster head listens to all values. Other nodes also register all messages they hear into memory;
3. All nodes send to the CH the list of their 1-hop neighbors[3];
4. The CH picks the n nodes among those with the highest residual energy, such that the n nodes cover all other nodes in range[4]. If needed, it selects some additional nodes to cover all the cluster;
5. The CH returns a message to selected nodes to assign them the role of cNode.

[3]We do not deal with the case of compromised nodes cheating at this step of the process. Indeed they could announce extra virtual neighbors to try to escape from coverage.

[4]The details of the algorithm executed by the cluster head at this step are not given in this study.

Note that some clustering algorithms (such as HEED [9], for example) provide other election mechanisms (for cluster heads, but that can also be used for selecting cNodes) based on residual energy. We do not want to use it because energy only takes part in the process as a factor for the probability that the nodes declare themselves elected. Instead we prefer nodes to broadcast their residual energy in order to enable surveillance by the vNodes.

5.1.3 Observations

cNodes apply a very basic trust based scheme to the cluster: when a sensor node breaks a rule, for example by exceeding a given threshold for transmitted packets, it is considered as untrustworthy. There are many other trust based schemes in the literature, most of them more advanced than this one (see Section 2). The cNodes could implement several other trust mechanisms (by lowering a score on bad behavior for each node, for instance). As more complex mechanism would create additional overhead, we prefer to limit ourselves to this simple method in this study.

5.2 Selection in practice: results from simulation

We have undertaken simulation of our second proposal regarding the energy consumption in order to compare it with the previous model (using pseudo-random election for cNodes). We used the ns-3 framework to proceed[5].

In the new proposal, the vNodes are to model the theoretical consumption of the cNodes they watch over. We have chosen to use Rakhmatov and Vrudhula's diffusion model [49] to compute the consumption. This choice was driven by several reasons:

- It provides a pretty accurate approximation of real consumption, taking into account chemical processes internal to the battery such as rate capacity effect and recovery effect.
- It is one of the models already implemented in ns-3. So in our case it is an absolutely perfect theoretical model. It remains "theoretical" as vNodes use this model to compute the expected behavior of cNodes according to the few packets they sometimes hear. Meanwhile, real cNodes consumption computed by ns-3 core takes into account every packet actually sent or received by cNodes, also including packets that vNodes can not hear (because of distance or sleep schedule). So the values computed by vNodes and ns-3 core will not always be the same, which allows us to use the model.

[5]To perform the simulations in this section we switched to ns-3, third major version of the network simulator tool. There were two reasons for this: the first one was that we found it more practical to implement the different applications for the nodes (*e.g.,* vNode application) through ns-3 architecture; the second reason was that we have realized this work later: as development and support for ns is progressively moving toward the third major version, so did we.

Rakhmatov and Vrudhula's diffusion model refers to the chemical reaction happening inside the battery electrolyte, and is summarized by the following equation:

$$\sigma(t) = \underbrace{\int_0^t i(\tau)\,\mathrm{d}\tau}_{l(t)} + \overbrace{\int_0^t i(\tau)\left(2\sum_{m=1}^{\infty}\exp^{-\beta^2 m^2 (t-r)}\right)\mathrm{d}\tau}^{u(t)}$$

where:

- $\sigma(t)$ is the apparent charge lost from the battery at t;
- $l(t)$ is the charge lost to the load ("useful" charge);
- $u(t)$ is the unavailable charge ("lost in battery" charge);
- $i(t)$ is the current at t;
- $\beta = \dfrac{\pi\sqrt{D}}{w}$, where D is the diffusion constant and w the full width of the electrolyte.

In practice, computing the first ten terms of the sum provides a good approximation (this is also the default behavior of ns-3, by the way).

We launched several simulation instances and chose to focus on the energy consumption and load balancing in the cluster. When we implemented our solution, we set the parameters of the simulation as detailed in Table 4.

We obtained the residual energy values for each node at each minute of the simulation. From this data we draw the average residual energy of the nodes

Table 4: Simulation parameters

Parameter	Value
Number of nodes	30 (plus 1 CH)
Number of cNodes	4
Probability for vNodes selection	33 %
Delay between consecutive elections	1 minute
Simulation length	30 minutes
Cluster shape	Squared box
Cluster length	Diagonal is 2×50 meters
Transmission range	50 meters
Location of the nodes	CH: center; others: random
Mobility of the nodes	Null
Average data sent by normal nodes	1024 bytes every 3 seconds
Data sent by vNodes (per target cNode)	1024 bytes every 5 seconds

(excluding cluster head) as well as the standard deviation. Average residual energy per minute in the batteries of the nodes is displayed in Figure 21.

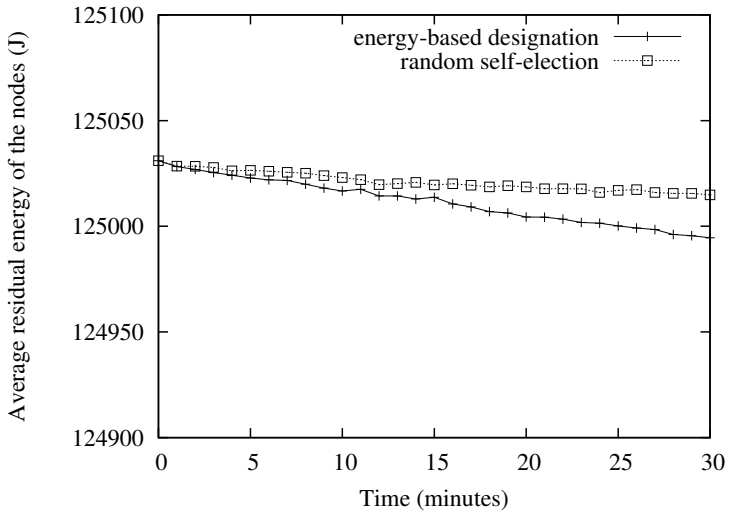


Figure 21: Average residual energy of the nodes (excluding cluster head)

Increasing values at $t = 11$ minutes and $t = 15$ minutes with the use of the proposed solution traduce the recovery effect of the batteries. As expected, our proposal causes increased global energy consumption. This is due, of course, to the new vNode role. vNodes have to wake up periodically to send requests to neighbor cNodes and to wait for an answer: this is energy-consuming. The estimated overhead for our solution appears on Figure 22.

Standard deviation of residual energy value in the nodes at each minute of the simulation is presented on Figure 23. During the first minutes of simulation, our solution creates a higher disproportion in load balancing due to the introduction of vNodes (there are more nodes assuming demanding functions). But after the first seven minutes or so, the standard deviation with our method falls below the standard deviation of previous method. This is the consequence of a better load repartition over the nodes with our solution. The difference between standard deviation with and without our simulation may look small: this is due to the model of the simulation we implemented. Given that we have a good pseudo-random numbers generator, when the number of elections get high, all nodes will roughly assume cNode role the same number of times in simulation *not* using our solution. As sensing nodes all have the same activity, a correct repartition of the cNode roles over the time leads to a good energy balance. But in a situation where sensing nodes have different activity levels—for instance, if there is an area in the cluster when measured events occur much more often than in the other parts of the cluster—the consumption would not be equilibrated between all the nodes with the previous method; whereas our

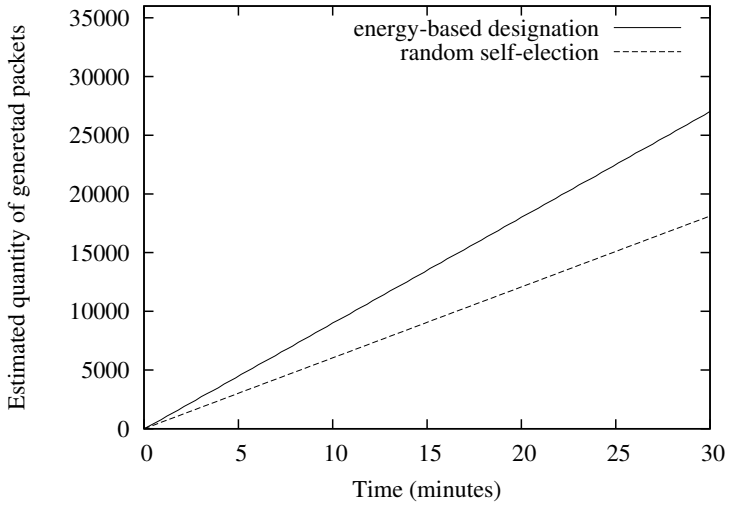


Figure 22: Estimated number of generated packets during the simulation

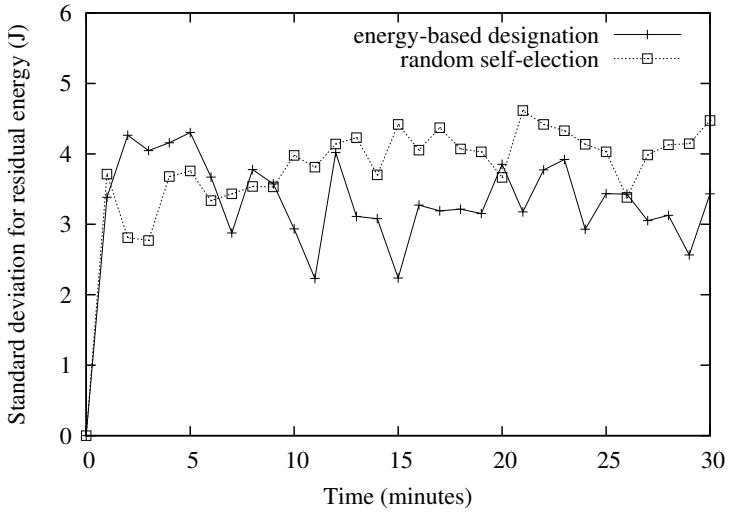


Figure 23: Standard deviation for residual energy of the nodes

solution would deal well with this case, since cNodes are elected according to residual energy. Thus simulations show that the use of vNodes leads to a higher energy consumption, but electing cNodes on residual energy provides a better load repartition in the cluster.

6 Conclusion

Detection of DoS attacks is a fundamental aspect of WSN management. We have considered a class of DoS detection mechanisms designed to operate on clustered wireless sensor networks: cNodes are used to monitor traffic of the nodes and to detect denial of service attacks (*e.g.,* flooding, black hole attacks). In the literature two basic election approaches have been proposed: static versus dynamic election. In this chapter, we have proposed two distinct election algorithms related to the dynamic approach to elect those cNodes, in an attempt to provide a better load balancing in the network.

The first one is a self-election similar to the process involved in LEACH clustering algorithm. With this solution we presented different modeling approaches for obtaining models of WSNs with DoS functionalities. First we 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. Hence we 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 a model of WSN 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. Expressive performance measures of the DoS GSPN models can be formally written and assessed by means of the recently introduced Hybrid Automata Stochastic Logic. We have then presented numerical results obtained with virtual WSN implementation *via* the ns-2 simulator. They confirm the intuition that cNodes' dynamic allocation guarantees a more uniform energy consumption (throughout the network) while preserving a good detection capability.

The second designation algorithm is based on the residual energy of the sensors. We have addressed several issues related with the use of this deterministic selection. Compromised nodes trying to systematically take over the cNode role are forced to abandon it for later cycle, or be detected, by vNodes. The vNode role is a new role we introduced to survey the cNodes by matching their announced energy consumption with a theoretical model. The issue of areas of the cluster uncovered by cNodes, depending of the activity in the cluster, is addressed by enforcing covering of the whole cluster: the cluster head is to designate additional cNodes if needed. The results we have obtained through simulations show that, even though using our simulation causes a higher global consumption of energy in the cluster, it provides an even better load repartition between sensors.

Working with clusters ensures a good scalability of network management. The detection system is also flexible, as cNodes can endorse various trust-based models, and monitoring rules can be set to fight against several types of denial of service attacks. Future developments of this work should 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 networks (different topologies and scales, areas with different activity levels).

References

[1] Jalel Ben-Othman and Bashir Yahya. Energy efficient and qos based routing protocol for wireless sensor networks. *Journal of Parallel and Distributed Computing*, 70(8):849–857, August 2010.

[2] Thibault Bernard and Hacène Fouchal. A low energy consumption MAC protocol for WSN. In *Proceedings of the 2012 IEEE International Conference on Communications (ICC'12)*, pages 533–537, Ottawa, ON, Canada, June 2012.

[3] William R. Claycomb and Dongwan Shin. A novel node level security policy framework for wireless sensor networks. *Journal of Network and Computer Applications*, 34(1):418–428, January 2011.

[4] Marcos A. Simplicio, Jr, Bruno T. de Oliveira, Paulo S. L. M. Barreto, Cintia B. Margi, Tereza C. M. B. Carvalho, and Mats Naslund. Comparison of authenticated-encryption schemes in wireless sensor networks. In *Proceedings of the 36th Annual IEEE Conference on Local Computer Networks*, pages 454–461, Bonn, Germany, October 2011.

[5] Jalel Ben-Othman, Karim Bessaoud, Alain Bui, and Pilard. Self-stabilizing algorithm for efficient topology control in wireless sensor networks. *Journal of Computational Sciences*, 4(4):199–208, July 2013.

[6] Fei Hu and Neeraj K. Sharma. Security considerations in ad hoc sensor networks. *Ad Hoc Networks*, 3(1):69–89, January 2005.

[7] Wendi Rabiner Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *Proceedings of the IEEE 33rd Hawaii International Conference on System Sciences*, volume 8, Maui, HI, USA, January 2000.

[8] Suat Ozdemir and Yang Xiao. Secure data aggregation in wireless sensor networks: a comprehensive overview. *Computer Networks*, 53(12):2022–2037, August 2009.

[9] Ossama Younis and Sonia Fahmy. HEED: a Hybrid, Energy-Efficient Distributed clustering approach for ad-hoc sensor networks. *IEEE Transactions on Mobile Computing*, 3(4):366–379, October 2004.

[10] Said Fouchal and Ivan Lavallée. Fast and flexible unsupervised clustering algorithm based on ultrametric properties. In *Proceedings of the 7th ACM Symposium on QoS and security for Wireless and Mobile Networks (Q2SWinet'11)*, Miami, FL, USA, November 2011.

[11] Said Fouchal, Quentin Monnet, Djamel Mansouri, Lynda Mokdad, and Malika Ioualalen. A new clustering algorithm for wireless sensor networks. In *Proceedings of the seventeenth IEEE Symposium on Computers and Communications (ISCC'12)*, Nevşehir, Turkey, July 2012.

[12] Yao Liang. Efficient temporal compression in wireless sensor networks. In *Proceedings of the 36th Annual IEEE Conference on Local Computer Networks*, pages 470–478, Bonn, Germany, October 2011.

[13] Gu Hsin Lai and Chia-Mei Chen. Detecting denial of service attacks in sensor networks. *Journal of Computers*, 4(18), January 2008.

[14] Malek Guechari, Lynda Mokdad, and Sovanna Tan. Dynamic solution for detecting denial of service attacks in wireless sensor networks. In *Proceedings of the 2012 IEEE International Conference on Communications (ICC'12)*, Ottawa, Canada, June 2012.

[15] Paolo Ballarini, Hilal Djafri, Marie Duflot, Serge Haddad, and Nihal Pekergin. HASL: an expressive language for statistical verification of stochastic models. In *Proceedings of the 5th international ICST conference on performance evaluation methodologies and tools (VALUETOOLS'11)*, pages 306–315, Cachan, France, May 2011.

[16] Ahmad Yusri Dak, Saadiah Yahya, and Murizah Kassim. A literature survey on security challenges in VANETs. *International Journal of Computer Theory and Engineering*, 4(6):1007–1010, December 2012.

[17] Sahabul Alam and Debashis De. Analysis of security threats in wireless sensor network. *International Journal of Wireless and Mobile Networks*, 6(2):35–46, April 2014.

[18] Soufiene Ben Othman, Abdullah Ali Bahattab, Abdelbasset Trad, and Habib Youssef. Confidentiality and integrity for data aggregation in WSN using homomorphic encryption. *Wireless Personal Communications*, September 2014.

[19] Quentin Monnet, Lynda Mokdad, and Jalel Ben-Othman. Data protection in multipaths WSNs. In *Proceedings of the eighteenth IEEE Symposium on Computers and Communications (ISCC'13)*, Split, Croatia, July 2013.

[20] Hai-Yan Shi, Wan-Liang Wang, Ngai-Ming Kwok, and Sheng-Yong Chen. Game theory for wireless sensor networks: a survey. *Sensors*, 12(7):9055–9097, July 2012.

[21] Ping Guo, Jin Wang, Jiezhong Zhu, and Yaping Cheng. Authentication mechanism on wireless sensor networks: A survey. In *Proceedings of the 2nd International Conference on Information Technology and Computer Science (ITCS'13)*, volume 25, pages 425–431, Beijing, China, July 2013.

[22] Ping Guo, Jin Wang, JieZhong Zhu, YaPing Cheng, and Jeong-Uk Kim. Construction of trusted wireless sensor networks with lightweight bilateral authentication. *International Journal of Security and Its Applications*, 7(5):225–236, September 2013.

[23] Harjot Bawa, Parminder Singh, and Rakesh Kumar. An efficient novel key management scheme for enhancing user authentication in a WSN. *International Journal of Computer Network and Information Security*, 5(1):56–64, January 2013.

[24] Leonidas Kazatzopoulos, Costas Delakouridis, and Christos Anagnostopoulos. WSN location privacy scheme enhancement through epidemical information dissemination. *International Journal of Communication Networks and Information Security*, 6(2):162–167, August 2014.

[25] Chinnu Mary George and Manoj Kumar. Cluster based location privacy in wireless sensor networks against a universal adversary. In *Proceedings of the International Conference on Information Communication and Embedded Systems (ICICES'13)*, pages 288–293, Chennai, India, February 2013.

[26] A. Varshovi and B. Sadeghiyan. Ontological classification of network denial of service attacks: basis for a unified detection framework. *Scientia Iranica*, 17(2):133–148, December 2010.

[27] Shio Kumar Singh, M. P. Singh, and D. K. Singh. A survey on network security and attack defense mechanism for wireless sensor networks. *International Journal of Computer Trends and Technology*, May 2011.

[28] Mohammad Momani and Subhash Challa. Survey of trust models in different network domains. *International Journal of Ad Hoc, Sensor and Ubiquitous Computing*, 1(3):1–19, September 2010.

[29] M. Carmen Fernández-Gago, Rodrigo Román, and Javier Lopez. A survey on the applicability of trust management systems for wireless sensor networks. In *Proceedings of the third international workshop on Security, Privacy and trust in Pervasive and Ubiquitous computing (SECPerU'07)*, pages 25–30, Istanbul, Turkey, July 2007.

[30] Mohammad Reza Rohbanian, Mohammad Rafi Kharazmi, Alireza Keshavarz-Haddad, and Manije Keshtgary. Watchdog-LEACH: a new method based on LEACH protocol to secure clustered wireless sensor networks. *Advances in Computer Science: an International Journal*, 2(3):105–117, July 2013.

[31] Adrian Perrig, Robert Szewczyk, Victor Wen, David Culler, and J. D. Tygar. SPINS: Security Protocols for Sensor Networks. *Wireless Networks*, 8(5):521–534, September 2002.

[32] Muhammad Hasan Islam, Kamran Nadeem, and Shoab A. Khan. Optimal sensor placement for detection against distributed denial of service attacks. In *Proceedings of the 2009 International Conference on Advanced Computer Control (ICACC'09)*, pages 675–679, Singapore, January 2009.

[33] Jun-Won Ho. Distributed detection of node capture attacks in wireless networks. In Hoang Duc Chinh and Yen Kheng Tan, editors, *Smart Wireless Sensor Networks*, chapter 20, pages 345–360. InTech, December 2010.

[34] Sudip Misra, P. Venkata Krishna, Kiran Isaac Abraham, Navin Sasikumar, and S. Fredun. An adaptive learning routing protocol for the prevention of distributed denial of service attacks in wireless mesh networks. *Computer and Mathematics with Applications*, 60(2):294–306, 2010.

[35] Ju-Hyung Son, Haiyun Luo, and Seung-Woo Seo. Denial of service attack-resistant flooding authentication in wireless sensor networks. *Computer Communications*, 33(13):1531–1542, August 2010.

[36] Amrita Ghosal and Sipra DasBit. A lightweight security scheme for query processing in clustered wireless sensor networks. *Computers and Electrical Engineering*, April 2014.

[37] Ameer Ahmed Abbasi and Mohamed Younis. A survey on clustering algorithms for wireless sensor networks. *Computer Communications*, 30(14–15):2826–2841, October 2007.

[38] M. J. Handy, M. Haase, and D. Timmerman. Low energy adaptive clustering hierarchy with deterministic cluster-head selection. In *Proceedings of the 4th IEEE International Workshop on Mobile and Wireless Communications Networks*, pages 368–372, Stockholm, Sweden, 2002.

[39] B. Brahma Reddy and K. Kishan Rao. A modified clustering for LEACH algorithm in WSN. *International Journal of Advanced Computer Science and Applications*, 4(5):79–83, May 2013.

[40] Naveen Chawla and Ashish Jasuja. Algorithm for optimizing first node die (FND) time in LEACH protocol. *International Journal of Current Engineering and Technology*, 4(4):2748–2750, August 2014.

[41] Leonardo B. Oliveira, Adrian Ferreira, Marco A. Vilaça, Hao Chi Wong, Marshall Bern, Ricardo Dahab, and Antonio A. F. Loureiro. SecLEACH — on the security of clustered sensor networks. *Signal Processing*, 87(12):2882–2895, December 2007.

[42] Maryam Mohi, Ali Movaghar, and Pooya Moradian Zadeh. A bayesian game approach for preventing DoS attacks in wireless sensor networks. In *Proceedings of the 2009 International Conference on Communications and Mobile Computing (CMC'09)*, Kunming, Yunnan, China, February 2009.

[43] Giuseppe Anastasi, Marco Conti, Mario di Francesco, and Andrea Passarella. Energy conservation in wireless sensor networks: a survey. *Ad Hoc Networks*, 7(3):537–568, May 2009.

[44] P. Sivakumar, K. Amirthavalli, and M. Senthil. Power conservation and security enhancement in wireless sensor networks: A priority based approach. *International Journal of Distributed Sensor Networks*, May 2014.

[45] Jinat Rehana. Security of wireless sensor network. Technical report, Helsinki University of Technology, 2009.

[46] Marco Ajmone Marsan, Gianfranco Balbo, Giuseppe Conte, Susanna Donatelli, and Giuliana Franceschinis. *Modelling with Generalized Stochastic Petri Nets*. John Wiley & Sons, November 1995.

[47] Paolo Ballarini, Hilal Djafri, Marie Duflot, Serge Haddad, and Nihal Pekergin. COSMOS: a statistical model checker for the hybrid automata stochatic logic. In *Proceedings of the 8th international conference on quantitative evaluation of systems (QEST'11)*, pages 143–144, Aachen, Germany, September 2011.

[48] M. Kwiatkowska, G. Norman, and D. Parker. Stochastic model checking. In M. Bernardo and J. Hillston, editors, *Formal Methods for the Design of Computer, Communication and Software Systems: Performance Evaluation (SFM'07)*, volume 4486 of *Lecture Notes in Computer Science*, pages 220–270. Springer, June 2007.

[49] D. Rakhmatov and S. Vrudhula. An analytical high-level battery model for use in energy management of portable electronic systems. In *Proceedings of the International Conference on Computer Aided Design (ICCAD'01)*, pages 488–493, San Jose, CA, USA, November 2001.