Energy-balancing method to detect denial of service attacks in wireless sensor networks

Quentin MONNET
Lab. LACL, Université Paris-Est
LACL (EA 4219), UPEC
F-94010 Créteil, France
quentin.monnet@lacl.fr

Lynda MOKDAD
Lab. LACL, Université Paris-Est
LACL (EA 4219), UPEC
F-94010 Créteil, France
lynda.mokdad@u-pec.fr

Jalel BEN-OTHMAN
Lab. L2TI, Université Paris 13
L2TI (EA 3043), UP13
F-93430 Villetaneuse, France
jbo@univ-paris13.fr

***Abstract*—The use of sensor networks 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 the network against denial of service attacks with the use of traffic monitoring agents on some nodes. But if the control nodes go down or get compromised, they leave the network unprotected. To better fight against attacks, we try to enhance this solution by introducing an energy-aware and secure method to select these monitoring nodes (called *cNodes*) in a clustered wireless sensor network. Our election process is done in accordance to their remaining reserves: nodes with the higher residual energy are selected. We discuss limitations of this deterministic process concerning security and cluster coverage, and suggest as a workaround to designate new control nodes (called *vNodes*). Those *vNodes* are responsible for monitoring the *cNodes* by periodically enquiring about their remaining energy and ensuring that they do not lie during the election process (in attempt to keep their *cNode* role). Finally, we present some experimental results obtained with the ns-3 simulator in order to analyze the impact of our proposal on the energy repartition in the network.

***Index Terms*—Wireless sensor networks; Reliability, availability, and serviceability; Energy-aware systems; Simulation**

INTRODUCTION

Traffic regulation or pollution measurement in water are example activities which require the constant presence of measuring agents over wide — and sometimes hard to access — areas. In such cases, as it is not feasible to send people on site to run measurements, wireless sensor networks are used. They are made of small devices, sometimes dropped on the spot by helicopter, tasked with gathering data on their physical environment. Sensors are able to exchange data through wireless communications. Useful data is typically centralized by a base station, which acts as an interface between the network and the user.

As they are often used in hostile environments with no human assistance, sensors are generally able to self-organize and to form a consistent network. But they also embed cheap hardware, as it may be hard, if possible at all, to fetch them once their life cycle is over. Consequently, they have restricted resources: low computational capabilities and low available memory. They also have limited energy [1] in a single-use battery.

Wireless sensor networks are used for many applications [2], [3], some of them being crucial. For instance there is a lot at stakes when sensor networks are used for watching forests for fires, for measuring the nuclear activity degree in sensitive areas, or for military operations over battlefields. In this context, bringing security guaranties — including availability — to the network becomes essential.

Based on former studies (see Section I), the present one relies on the use of monitoring nodes (or "*cNodes*") to protect a wireless sensor network again various denial of service attacks. Actually, it focuses on the election process of those control nodes. Our approach consists in taking energy into account at this step, 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.

The rest of the paper is organized as follows: first we present a quick overview of related work in Section I. After we introduce a new *cNode* selection method in Section II, we describe and discuss the simulation of the algorithm that we have done using ns-3 in Section III. Last Section summarizes the main contributions of the paper and gives some directions of future work.

I. RELATED WORK

For sensitive operations involving the deployment of a wireless sensor network, all security aspects of the network must be reviewed. WSN-addressed protocols to provide data privacy [4] and authentication [5] have been subject to deep research investigation, and led for example to the proposal of mechanisms for secure data aggregation without persistent cryptographic operations [6] or for authenticated broadcast

such as μTESLA [7]. But cryptography is of no use if the network is down: in this paper we focused on resistance against denial of service (DoS) attacks.

Indeed there are many existing attacks able to compromise the good working of a wireless sensor network [8]. Several mechanisms have been proposed to detect it and to provide countermeasures [9], [10]. Many consist in the implementation of trust based mechanisms [11], [12] with agents applying set of rules [13] on traffic to attribute a trust value to each of the nodes in the network. In particular, Lai and Chen [14] proposed to elect control nodes to monitor the traffic in clustered networks and to detect and react to denial of service attacks. Clustered networks are partitioned into clusters *via* algorithms such as *LEACH* [15]. One cluster head (CH) per cluster is responsible for gathering data from its peers, for aggregating and sending it to the base station. The CHs are the only nodes to use long range (and expensive) transmissions to reach the base station. Clustering enables to preserve energy for the sensing nodes and offers an easier management of the nodes. In our case, all nodes of a cluster can reach their cluster head directly (*1-hop transmission*), as on Figure 1.

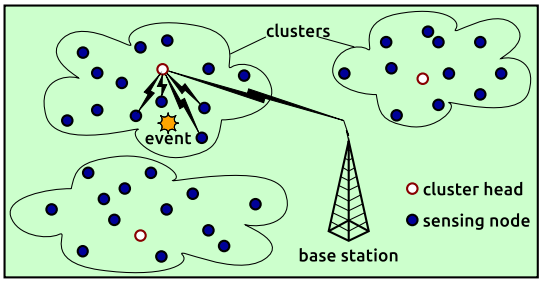

Figure 1. Clustered wireless sensor networks scheme

Control nodes are elected among the non-cluster head nodes of a cluster. We will call them *cNodes* from now on. They are responsible for listening to the traffic and detecting nodes whose emitted traffic exceeds a given threshold value. These abnormal behaviors are reported to the cluster head. 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. Although the method is efficient for detecting rogue nodes, the authors do not give details of the election mechanism to choose the *cNodes*. Also, there is no mention in their study of renewing the election in time, which causes the appointed *cNodes* to endorse a heavier energy consumption on a long period.

In a first attempt to bring load balancing to this solution, we propose in other papers [16], [17] to reiterate the election periodically. Simulations show a better load repartition among the cluster, but at this time our focus was not on designing an energy efficient election process for the *cNodes*. This is the purpose of the current study.

II. CNODES SELECTION MECHANISM

Using control nodes to watch over the network traffic allows the detection of various types of denial of service attacks. This is achieved with agents in the *cNodes* applying specific rules on overheard traffic. Each rule is used to fight against one kind of attack: jamming, tampering, black hole attacks, and so on [13]. Due to limited size for this paper, we can not described in details the attacks and the associated rules. We will only treat one example in the rest of this study: 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 [18]. In wireless sensor networks, it is also used to drain the energy of neighbor nodes. *cNodes* are responsible for listening to the traffic of their surrounding nodes: if a sensor is to generate more traffic in the network than a predetermined threshold, it is considered as potentially compromised and trying to flood the network. A report is sent to the cluster head. On reception of reports coming from multiple *cNodes*, the CH considers that traffic from the suspicious node must no more be considered. Information about distrust is passed on to normal nodes which stop listening to the packets coming from the attacker. We work under the following assumptions: firstly, the cluster head is not a compromised node (the use of *cNodes* to detect a malicious CH is described in [14], but it is not considered in this paper). Secondly, we do not consider the case of several malicious nodes cooperating with one another.

Electing the *cNodes* is not an easy task. In [17] we expose and compare 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. In that previous study however we do 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 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.

A. 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 first step, each node evaluates its residual energy and sends the value to the cluster head;
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 which 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 (based on LEACH [15] model for instance), 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 *cNode* (in other words, it can survey several neighbor *cNode*).

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 (regarding to the model) than 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 a 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 to get elected at one point in the time.

Thus, the interest of *vNodes* can be summarized as follows: a compromised node can not 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 to process to its normal sensing activity (requests to *cNodes* must not occur often, otherwise

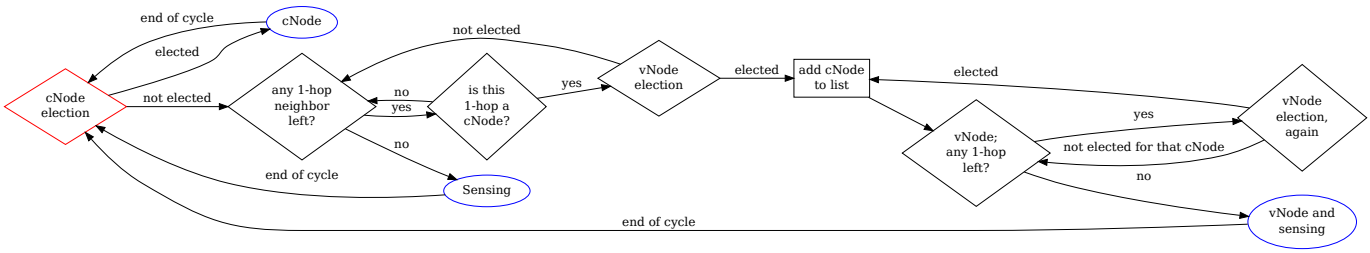

Figure 2. State machine of the (non-CH) nodes

it will drain too much power from the *vNodes*). The state machine of the nodes is presented in Figure 2.

B. 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, that 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 do not fade, possibly for all cycles. Figure 3 illustrates this problem.

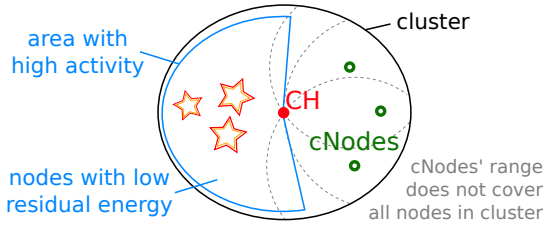

Figure 3. Illustrative scheme: *cNodes* are elected inside the area with less activity (thus with more residual energy) and do not cover nodes from the opposite side of the network.

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 II-A needs to be modified. The correct version is as follows:

1) During 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[1];
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[2]. If needed, it selects some additional nodes to cover all the cluster;
5) The CH returns to selected nodes a message to assign them the role of *cNode*.

Note that some clustering algorithms (such as HEED [19] 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 probability that the nodes declare themselves elected. Instead we prefer nodes to broadcast their residual energy in order to enable surveillance by the *vNodes*.

C. 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 literature, most of them more advanced than this one (see Section I). The *cNodes* could implement several other trust mechanisms (by lowering a score on bad behaviour for each node for instance). As more complex mechanism would create additional overhead, we prefer to limit to this simple method in this study.

III. Selection in practice: results from simulation

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

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 [20] 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 behaviour 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*,

[1]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.

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

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.

Rakhmatov and Vrudhula's diffusion model refers to the chemical reaction happening inside the battery electrolyte, and is summarized by equation (1).

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

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;
- $\beta = \frac{\pi\sqrt{D}}{w}$, 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. To obtain data about detection rate or false positive values of the *cNodes* scheme, the reader is redirected to our previous work [16], [17]. When we implemented our solution, we set the parameters of the simulation as detailed in Table I.

Table I
PARAMETERS USED FOR SIMULATIONS

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

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 (excluding cluster head) as well as the standard deviation. Average residual energy per minute in the batteries of the nodes is displayed on Figure 4. 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 an 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 5.

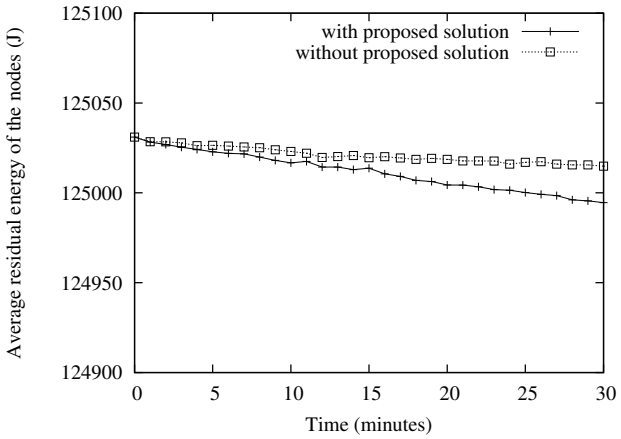

Figure 4. Average residual energy of the nodes (excluding cluster head)

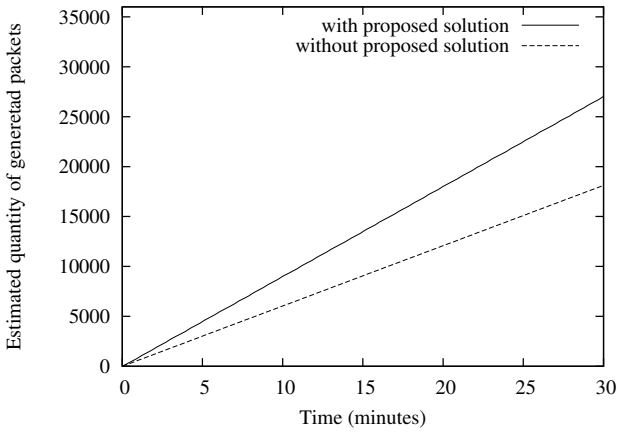

Figure 5. Estimated number of generated packets during the simulation

Standard deviation of residual energy value in the nodes at each minute of the simulation is presented on Figure 6. 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 seven first 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

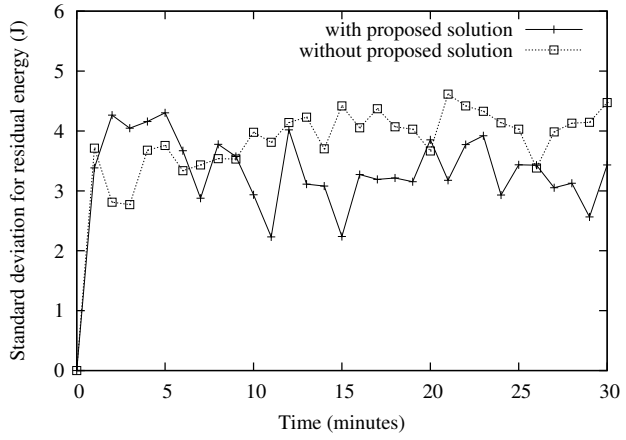

Figure 6. Standard deviation for residual energy of the nodes

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 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.

Conclusion

cNodes are used in clustered wireless sensor networks to monitor traffic of the nodes and to detect denial of service attacks (*e.g.* flooding, black hole attacks). In this paper, we have proposed a new method to dynamically elect those *cNodes*, based on their residual energy. The aim of the proposed selection algorithm is to provide a better load balancing in the cluster.

We have addressed several issues related with the use of a deterministic selection. Compromised nodes trying to systematically take over the *cNode* role are forced to abandon it for later cycle, or get 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. Working with clusters ensures a good scalability of the solution. It is also flexible, as *cNodes* can endorse various trust-based model, and monitoring rules can be set to fight against several types of denial of service attacks. And the use of *vNodes* is resilient to a small percentage of compromised *vNodes* (depending on parameters set by user).

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 a better load repartition between sensors.

Future works include improvements of our solution by adding monitoring of the cluster head, as well as modeling a cluster with areas of different activity levels. Also we would especially like to study the impact of the percentage of designated *vNodes* on global energy consumption.

References

[1] T. Bernard and H. Fouchal, "Slot scheduling for wireless sensor networks," *Journal of Computational Methods in Science and Engineering*, vol. 12, no. 1, pp. 1–12, Nov. 2012.

[2] C. Ramassamy, H. Fouchal, and P. Hunel, "Classification of usual protocols over wireless sensor network," in *Proceedings of the 2012 IEEE International Conference on Communications (ICC'12)*, Ottawa, Canada, Jun. 2012, pp. 622–626.

[3] H. Fouchal and Z. Habbas, "Distributed backtracking algorithm based on tree decomposition over wireless sensor networks," *Concurrency and Computation: Practice and Experience*, vol. 25, no. 5, pp. 728–742, Apr. 2013.

[4] S. Ozdemir and Y. Xiao, "Secure data aggregation in wireless sensor networks: a comprehensive overview," *Computer Networks*, vol. 53, no. 12, pp. 2022–2037, Aug. 2009.

[5] M. A. Simplicio, Jr, B. T. de Oliveira, P. S. L. M. Barreto, C. B. Margi, T. C. M. B. Carvalho, and M. Naslund, "Comparison of authenticated-encryption schemes in wireless sensor networks," in *Proceedings of the 36th Annual IEEE Conference on Local Computer Networks*, Bonn, Germany, Oct. 2011, pp. 454–461.

[6] K. Wu, D. Dreef, B. Sun, and Y. Xiao, "Secure data aggregation without persistent cryptographic operations in wireless sensor networks," *Ad Hoc Networks*, vol. 5, no. 1, pp. 100–111, Jan. 2007.

[7] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. D. Tygar, "SPINS: Security Protocols for Sensor Networks," *Wireless Networks*, vol. 8, no. 5, pp. 521–534, Sep. 2002.

[8] J. Zheng and A. Jamalipour, *Wireless sensor networks: a networking perspective*. Wiley-IEEE Press, Oct. 2009.

[9] S. K. 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.

[10] H. Sedjelmaci, S. M. Senouci, and M. Feham, "An efficient intrusion detection framework in cluster-based wireless sensor networks," *Security and Communication Networks*, vol. 6, no. 10, pp. 1211–1224, Oct. 2013.

[11] M. Momani and S. Challa, "Survey of trust models in different network domains," *International Journal of Ad Hoc, Sensor and Ubiquitous Computing*, vol. 1, no. 3, pp. 1–19, Sep. 2010.

[12] M. C. Fernández-Gago, R. Román, and J. 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)*, Istanbul, Turkey, Jul. 2007, pp. 25–30.

[13] M. R. Rohbanian, M. R. Kharazmi, A. Keshavarz-Haddad, and M. Keshtgary, "Watchdog-LEACH: a new method based on LEACH protocol to secure clustered wireless sensor networks," *Advances in Computer Science: an International Journal*, vol. 2, no. 3, pp. 105–117, Jul. 2013.

[14] G. H. Lai and C.-M. Chen, "Detecting denial of service attacks in sensor networks," *Journal of Computers*, vol. 4, no. 18, Jan. 2008.

[15] 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*, Stockholm, Sweden, 2002, pp. 368–372.

[16] M. Guechari, L. Mokdad, and S. 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, Jun. 2012.

[17] P. Ballarini, L. Mokdad, and Q. Monnet, "Modeling tools for detecting DoS attacks in WSNs," *Security and Communication Networks*, vol. 6, no. 4, pp. 420–436, Apr. 2013.

[18] J. Rehana, "Security of wireless sensor network," Helsinki University of Technology, Technical report, 2009.

[19] O. Younis and S. Fahmy, "HEED: a Hybrid, Energy-Efficient Distributed clustering approach for ad-hoc sensor networks," *IEEE Transactions on Mobile Computing*, vol. 3, no. 4, pp. 366–379, Oct. 2004.

[20] 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)*, San Jose, CA, USA, Nov. 2001, pp. 488–493.