Overcoming the Deficiencies of Collaborative Detection of Spatially-correlated Events in WSN

Martin Peres[1], Romain Perier[1], and Francine Krief[1]

Université de Bordeaux, LaBRI
{martin.peres, romain.perier, krief}@labri.fr

Abstract. Collaborative WSN are known to efficiently correlate sensors' events to detect spatially-correlated events with a low latency. However, because of the drastic reduction of messages sent to the network administrator, it is difficult for him/her to understand in what state the network is. In this paper, we propose a modality-agnostic collaborative detection of spatio-temporally correlated events that drastically lowers power consumption by both reducing and localising communication. This contribution can then be used to expose better auditing capabilities that overcome the problems usually found in collaborative networks. These capabilities can in turn be used by both the administrator and the network itself to, for instance, automatically react to faulty sensors. Experiments validate the interest of our proposal in an intrusion detection scenario.

Keywords: Wireless Sensor Networks, collaborative event-detection, low-latency, energy-efficient, autonomic-networking

1 Introduction

As today's society expects the availability of information in real time on about anything, the problem of collecting and distributing information from remote or poorly-accessible places becomes more and more apparent.

The use of Wireless Sensor Networks (WSN) as a mean to lower deployment cost compared to a traditional wired sensor network is a widely accepted fact. WSN are already used in several fields such as monitoring air-quality, seismic activity, forest fires, structural integrity of a building and also area intrusion detection.

Sensor nodes are small wireless network nodes characterised by their very low processing power and storage capacity (both ROM and RAM). They are also characterised by very limited energy resources. Despite these constraints, sensors are usually expected to be secure and serve data with a relatively low latency and operate over a long period without swapping their batteries.

As the number of sensor networks grows, less and less human resources can be used to administrate them. This problem can be addressed by working towards autonomic and collaborative WSN that would not only lower maintenance but also improve the quality of service and increase power efficiency.

This research has been conducted during the DIAFORUS project [9]. The project's goal is to develop an energy-efficient framework for distributed applications and functions over redundant unattended sensors. As a demonstration of the framework, an intrusion detection application was written using redundant and heterogeneous sensors.

In Sect. 2, we introduce the state of the art concerning distributed and collaborative decision-making in Wireless Sensor Networks. Section 3 describes our proposal and how we believe it can not only reduce power consumption but also improve collaboration and error reporting. Then, Sect. 4 evaluates the proposal in an intrusion detection scenario and Sect. 5 introduces the future work. Section 6 concludes the paper.

2 State of the Art

According to [1], the energy cost of transmitting 1 KB over a distance of 100 m is approximately the same as the cost of executing 3 million instructions by a 100 million instructions per second (MIPS)/W processor. Moreover, with processors' efficiency doubling every 18 months (Koomsley's law [7]), it is clear that processor time is getting cheaper and cheaper while communications are bound by physical properties that limits efficiency improvements.

Based on these facts, it is clear that improving the lifespan of WSNs means diminishing both the number and the average length of communications.

The following describes common data-processing schemes and architectures for WSN and compares them from a number and average communication length point of view.

2.1 The Sink

Wireless Sensor Networks started as simple ad-hoc networks with the information flow going from all the sensors to the gateway. Indeed, each sensor node would collect data from the sensor, average/aggregate it and then send this data to the gateway. The gateway either processes the information itself, stores it or sends it through the Internet to another server. This gateway node is often refered as "the sink" as all the traffic ultimately is routed through him.

In this kind of WSN, sensors' readings are usually stored before being sent in batch to limit the number of packets at the expense of latency.

The average communication length on these architectures is the average route length that links every sensor node to the gateway. As all sensor nodes send data periodically, this kind of network's lifespan depends on how often the sensors' data is sent to the gateway.

2.2 Cluster-based Data Aggregation

A way to lower the average length of communication is to let a data-aggregation cluster node gather data from his surrounding nodes, aggregate it into a single message and send it to the gateway.

This architecture introduces two kinds of communications:

– Short-distance: A one-hop communication
– Long-distance: A multi-hop communication to the sink

Long-distance communications should be avoided as much as possible to lower power consumption in the network. Indeed, they not only consume energy on both the emitter and gateway, but they also consume the energy of every routing node in-between. Moreover, nodes around the route are also consuming more energy because they need to receive and parse at least the MAC header in order to determine if they are the recipient or not. Finally, since the gateway is always the ultimate destination for every packet, the closer a node is to the gateway, the less likely the communication medium is to be available because of the ever-increasing traffic found when getting closer to the gateway. This further increases power consumption.

On the other hand, short-distance communications consume energy on a much more limited zone and more uniformly.

With the cluster-based data aggregation architecture, most communications are local and thus, spatially spread power consumption more evenly.

2.3 Local Event Detection

Instead of addressing the average communication length problem, local event detection addresses the problem of the number of messages.

Both the sink and the cluster-based data aggregation architectures rely on the outside of the network to process data. These architectures are data-oriented as they only acquire and forward sensors' data to the gateway.

If the WSN just feeds an application that is run outside the network, then it is possible to reduce the number of messages by only sending sensors' data conditionally [8].

For instance, in an intrusion detection application, it isn't necessary to constantly send the raw values read by the sensors. Instead, local processing of sensors' data avoids unnecessary transmissions by letting the sensor node detects whether the sensor detects something or not.

Local event detection may be challenging for sensor nodes because of their very tight CPU, ROM and RAM constraints, but it is possible to use digital signal processing units(DSP) [11] as these processors are getting more and more efficient [7].

Awareness of the WSN's application enables sensor nodes to limit the number of messages they emit by filtering the unneeded information [4][10].

2.4 Collaborative Detection

In spatially-correlated sensor networks, collaborative detection builds on the concepts of both local data aggregation and cluster-based data aggregation.

Sensors are first locally correlated to filter unneeded information and then, these information are collected in a cluster in order to correlate all the sensors of a given area [12][14].

This method further limits the message count by avoiding false positives induced by defective or ill-calibrated sensors. Less false positives also means reduced long-distance communications.

This method also works on heterogeneous sensor networks [6][2] and is also known to produce better results than value-fusion as found in cluster-based data aggregation when fault-tolerance is needed [3][13].

This architecture provides both a lower average communication distance and fewer messages in the network because of its aggressive filtering at both the node level and the cluster level.

However, to our knowledge, already-existing collaborative detection networks do not abstract the sensor modality. This severely limits the addition of new sensors at run time since the correlating node needs to be aware of all the sensor types and know how to correlate them.

Moreover, collaborative detection makes it more difficult for the network administrator to know if the network is operating correctly because of the low volume of messages. In case of a detection failure, it also makes it harder for her/him to debug what went wrong because no logs are available on her/his work computer.

Finally, because of the correlation found in collaborative detection, it is difficult for the network administrator to detect faulty sensors.

3 Our proposal

To address the shortcomings of the state-of-the-art collaborative detection, we propose three different reasoning components:

- modality-agnostic collaborative detection of spatio-temporally correlated events
- offline logging capabilities
- sensors reputation management

3.1 Modality-agnostic Collaborative Detection of Spatio-temporally Correlated Events

Our proposal is based on the following assumptions:

1. an area defines a spatial zone where all sensors are correlated and can communicate with each other;
2. area's sensors should all detect an intrusion within a definite maximum time called maximum intrusion time;
3. sensors are noisy and randomly emit false positives;
4. sensors' may not be reachable at all time;
5. sensors are ill-calibrated.

Assumptions 1 and 2 have to be enforced when sensors are deployed. It also means network's lifespan can be improved by using collaborative detection to lower both the number of messages and the average distance of communications.

Assumptions 3 and 4 mean that the network should be fault-tolerant and thus, be using decision-fusion [3].

Assumption 5 raises the problem of sensors correlation. Since non-calibrated sensors' values cannot be averaged, it means that the values read by the sensor can only be interpreted by the sensor node that produced it. Indeed, the value can only be interpreted when put into its context, that is to say, the previous values/trends. Moreover, decision-fusion mandates the sensor to produce a decision whether it detects an event or not.

Thus, instead of reporting a given value, a sensor needs to locally correlate its values then decides if an event is going on or not. If an event is detected, a confidence rating ranging from 1 to 3 is attributed to the detection.

This proposal abstracts the sensor type and has many additional benefits such as:

– hiding away calibration errors from the correlation node;
– correlating sensors of arbitrary type.

Messages between the sensors and the reasoning node are sent using a Publish/Subscriber communication paradigm. This is done to ease the creation of topic of interests sensors can subscribe to. Moreover, publisher do not have to know which sensor nodes are interested in. This allows the creation of an highly-dynamic collaborative behaviour.

In our proposal, individual sensors are required to detect events by themselves and give a confidence rating ranging from 1 to 3. The algorithm that should be followed highly depends on the sensor modality and the event that should be captured. As an example, let's imagine we want to use a binary infrared optical barrier to detect a car driving down a street. The signal produced by the sensor will be a simple square whose length will depend on the speed of the car.

To detect such a square signal, the sensor node can poll the sensor's value periodically. The polling period depends on how wide the optical barrier is and how fast the car is. The maximum speed of the event to detect is one parameter of the area. As soon as the polled sensor detects something, a correlation timer is set. If the value sensor keeps on detecting something for, for instance, more than a tenth of the minimum detection time, then a message (hereinafter referred to as "alert") should be sent to the correlation node with the minimal confidence level. The confidence level should then be increased gradually to 3 as long as the sensor keeps on detecting the intruder.

When the correlation node receives an event from a sensor (alert), it stores both the current timestamp and the confidence level into memory for future reference. Then, it calculates the area criticality level by summing the contribution of all the sensors of the area.

A sensor's contribution to the criticality level is the confidence level of the alert times the age factor. The age factor linearly decreases from 1 to 0 in *correlation_time* seconds. The correlation time should be roughly the same as the maximum time it takes for an event to happen. In a intrusion detection scenario, it is the maximum time it takes for a pedestrian to cross the area.

The age factor is important because alerts should expire after some time. Moreover, older alerts shouldn't influence the correlation as much as fresh alerts do. The choice of a linearly decreasing function is motivated by the simplicity of implementation. It may however need to be adjusted depending on the kind of application you want to use this system in.

Computing the area's criticality level is done following (1).

$$age\_factor(a) = \begin{cases} 0 & \text{if alert\_age(a) > correlation\_time} \\ 1 - \dfrac{alert\_age(a)}{correlation\_time} & \text{otherwise} \end{cases}$$

$$\begin{aligned} alert\_age(a) &= current\_time() - alert\_timestamp(a) \\ contrib\_alert(a) &= confidence(a) * age\_factor(a) \\ criticality &= \sum_{a=0}^{alert\_count} contrib\_alert(a) \end{aligned} \tag{1}$$

When most sensors from an area simultaneously detect an event, the criticality level of the area should be higher than a threshold that depends on the number of sensors in the area and the minimal correlation factor we want to achieve.

When the criticality level goes over the threshold, the correlating node of the area publishes a message telling that an event has been detected by the area. This message, hereinafter referred to as alarm, can then be used by other nodes to trigger automatic actions such as lighting up a light-bulb or an alarm.

Being able to automatically trigger responses to an event enables fully autonomic collaborative WSN.

We also propose that the first node to be added to an area should be "elected" as the correlating node. However, this node may not be available during the whole lifespan of the network for the following reasons:

- energy source depletion;
- hardware malfunction;
- selective jamming on this node.

To overcome these challenges, the correlation node role should be split in two. Both correlation nodes would subscribe to the alerts sent in their area. The only difference between the two nodes would be that the master node (the first one elected) would emit an alarm as fast as possible. On the contrary, the slave correlation node (the second node to be added to the area) would wait for a few seconds before emitting an alarm unless the master node emits the alarm before the expiration of the delay.

This proposal increases reliability by avoiding the single point of failure that was the correlation node. However, a re-election should be made whenever the master correlation node's battery is running low or when the slave correlation node detects that the master isn't behaving correctly. In this case, the slave should elect himself as the master and query a list of potential candidates in order to select a suitable slave node to succeed him. If the slave node becomes unavailable, the master node should re-elect another slave node.

The election of a correlation node should be made according to these criteria:

– energy available (in Joules, not percentage);
– number of hops to the gateway;
– number of routes to the gateway.

Redundancy could then be further improved by electing n correlation nodes with an increasing delay between detection and the emission of an alarm. This works because every node of the area can communicate with each others (assumption 1).

However, redunding the correlation node comes at the expense of power consumption and thus, the network lifespan. Indeed, every correlating node is required to subscribe and receive every alert and alarm sent by the area and process them. As a result, the average power consumption of the correlating nodes is higher than other nodes in the area. This means there is trade-off between power consumption and redundancy.

3.2 Offline Logging Capabilities

Due to the drastic message reduction found at the gateway when using in-network reasoning, auditing the system becomes difficult. Auditing is needed by the network administrator to understand what is going wrong with the system in case of false positives or negative detections.

To address this problem, the correlating node should be required to store the history of what happened in an area in the past hours or days. Given the stringent constraints found on sensor nodes, it is impossible to store all the data. It is thus important to find an efficient way to only keep meaningful data.

The proposed solution stores events. An event is characterised by a local time stamp, a confidence level and the list of sensors contributing to this event and their relative contribution to it.

The system can only store a selected few events. When a new event needs to be stored, a usefulness score is attributed to each event in the history. The event with the lowest score gets replaced by the new event.

An important event is an event that both drove the confidence level close to the alarm threshold and was also a local maximum for a long time. However, when an event gets older, its importance tends to lower. This is why the scoring system (2) depends on the confidence level of the event, how long it was a local maximum and the age of the event.

$$score(e) = \frac{confidence(e) * local\_max\_time(e)}{1 - \frac{age(e)}{max\_storage\_time}} \tag{2}$$

The history can then be queried on-demand by a network administrator using a REST-like protocol such as CoAP [5].

3.3 Sensors Reputation Management

The history is not only useful to the network administrator, it is also useful to the WSN itself. An obvious usage of the history is to juge how useful a sensor usually is at detecting a certain type of event. We refer to this usefulness score as reputation.

We separate reputation in two scores:

- False positive: How often are the events emitted by a sensor not correlated with his surrounding nodes;
- False negative: How often is a sensor not participating in the correlation of real events.

Both reputations are represented by a value ranging from 0 to 1, 0 being the lowest possible reputation and 1 being the best.

$$reputation\_fp(a, s) = \frac{area\_detection\_count\_involving(a, s)}{sensor\_events\_count(s)} \tag{3}$$

False positive reputation is calculated when alerts are deleted from the history. When an alert is deleted, the detection count (sensor_events_count(s)) of the associated sensor is incremented. If the alert has been used to emit an alarm, then the alarm counter of the associated sensor (area_detection_count_involving(a,s)) is incremented. The false positive reputation is just the ratio of alerts that have been correlated over the total alert count for a given sensor. The equation is detailed in (3).

$$reputation\_fn(a, s) = \frac{sensor\_correlated\_count(s)}{area\_detection\_count(a)} \tag{4}$$

False negative reputation is calculated when an event is added to the history. If the event is an alarm, then the alarm count (area_detection_count(a)) is incremented. Then, all the sensors that participated to this alarm have their correlation count incremented (sensor_correlated_count(s)). False negative reputation is the ratio of how many times a sensor was involved in the emission of alarms over the total alarm count, the equation is detailed in (4).

Additionally, the reasoning node could alert the administrator when one sensor gets one of his reputations lower than a certain threshold. It can also be used during the correlation process to weight a sensor contribution according to its false positive reputation. A lower false positive reputation would result in a lower contribution during the event correlation.

4 Evaluation

For evaluating our proposal, we used a physical intrusion detection scenario. The latency of detection should be under 10 seconds and usually around 5 seconds. It was decided that sensor nodes shouldn't correlate values for more than 1 second before sending it to the network. This leaves up to 4 seconds for the network to carry the detection message and its acknowledgement(ACK). The event is re-emitted if no ACK is received by the sensor node that emitted the detection message.

It is however unnecessary to flood the network with alarms when an intrusion is detected. Two alarms can be separated by at least the area's minimum crossing time unless a new sensor has been correlated with an existing alarm. This amendum to the rule is made so as the administrator gets new meaningful information in the timeliest fashion. This proposal enables the administrator to be aware of the current situation while also limiting the number of messages.

The evaluation has been carried out in a custom-made simulated environment based on FreeRTOS. Each simulated sensor node is executed as a FreeRTOS Linux process. When a sensor sends a message, instead of sending it through the radio like it would be done on real nodes, the message is sent using TCP sockets to a program called dispatcher. The dispatcher emulates the communication medium. It forwards the message to the destination node if nodes are within reach of each other.

The dispatcher gets the sensor nodes' location from an XML deployment file. This file not only lists the sensor nodes, their positions and their radio range, it also contains all the sensor-specific configuration such as what sensors (modality) are connected to each sensor node and how (i2c address, gpio line, ...). This deployment file is used to generate the node configuration as C header files that are then compiled and linked with the node's firmware.

In simulation, the values read by the sensors are generated using a simulation environment called Diase. This environment, developed for the ANR DIAFORUS, enables:

– the deployment of a simulated network;
– the creation of intrusion scenarios;
– the monitoring of both sensor nodes and the network.

Monitoring gathers data from the network using a REST protocol for constrained applications called CoAP. It then displays the gathered data into a textual or a graph form.

Figure 1 is a screenshot of Diase running an intrusion scenario.

4.1 Modality-agnostic Collaborative Detection of Spatio-temporally Correlated Events

Distributed and collaborative detection is meant to lower both the number and the average length of communications. We evaluate those metrics in this subsection.

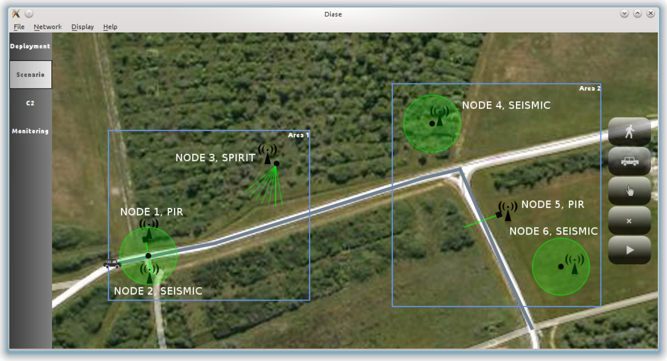

Fig. 1. Diase simulating an intrusion scenario

Sensors are never perfect, if calibrated to detect the smallest event they will be prone to false positives. We choose to model the detection of an event by a sensor using a Bernoulli distribution. When no real intrusion is going on, sensors will have a probability p of wrongly detecting an intrusion and a probability of $1 - p$ of not detecting one. The experience is repeated every second.

This means we have a probability p of getting a false positive. Knowing this, we evaluate how many messages are exchanged in different kinds of Wireless Sensor Networks. A distinction between short-distance (no hops) and long-distance (towards the gateway) communications is also introduced. The number of messages is then compared to the number of values read by the sensor. This comparative study is done for an area of 3 nodes, reading values every second for 30 minutes. The results can be found in Table 1.

WSN type	readings	short-distance	long-distance
Sink	5400	0 (0%)	5400 (100%)
Cluster aggregation	5400	3600 (67%)	1800 (33%)
Local detection	5400	0 (0%)	540 (10%)
Collaborative detection	5400	$\leq$ 540 (10%)	$<$ 180 (3.33%)

Table 1. Comparing WSN data management on a 3-nodes area with a sensor noise probability (p=0.1, f=1Hz)

In the *sink* WSN, all sensors readings are forwarded to the gateway every second to fulfil the 1 second correlation time rule. This only generates long-distance traffic and is the worst possible case.

In the *cluster data aggregation* WSN, 2 of the 3 sensors send their values to the aggregation node. This node aggregates these 2 values with the value of his sensor in one message before sending it to the gateway. This means 67% of the traffic is local and 33% of the traffic is long-distance.

In the *local event detection*, sensors detect intrusions themselves. When they detect a suspicious event, they forward the acquired value to the gateway. As the sensor's probability to detect an event is 10%, then 10% of the values read are forwarded to the gateway and thus, be considered as long-distance communications. No short distance traffic is generated.

In the *collaborative detection* WSN, when sensors detect an event, it is sent to the correlation node. This node then correlates these events that expire after a certain amount of time that depends on the maximum time it takes to cross the area. This means up to 10% of the communications will be local. This value depends on the number of sensors connected to the correlation node. Telling how often a long-distance communication will happen is non-trivial because this highly depends on the algorithm used to correlate the local events.

We now move on to testing the influence of the sensor noise and the correlation time on the number of short-distance and long-distance communications in our proposal. All these communications are false positives, so lower is better. We simulate the worst case scenario, when no sensors are connected to the correlation node.

Correlation	readings	short-distance	long-distance	$\frac{long}{short}$ ratio
c=10s	5400	545 (10%)	8 (0.15%)	1.5%
c=20s	5400	558 (10.3%)	23 (0.43%)	4.1%
c=40s	5400	541 (10%)	32 (0.59%)	6%
c=60s	5400	516 (9.5%)	33 (0.61%)	6.3%
c=90s	5400	525 (9.7%)	40 (0.74%)	7.7%
c=120s	5400	518 (9.5%)	41 (0.76%)	7.8%
c=150s	5400	552 (10.2%)	50 (0.93%)	9%
c=180s	5400	520 (9.6%)	56 (1.03%)	10.7%

Table 2. Message count in DIAFORUS with noisy sensors (f=1Hz, p=0.1) and a correlation time c. Experiment time of 30 minutes.

In Table 2, we evaluate the influence of the correlation time over the number and length of communications. As expected, the number of local detections is around 10%, which is the sensor noise. Concerning the long-distance communications, the longer the correlation time, the higher the number of messages. This can be explained because the longer the correlation time, the higher the probability of correlation between sensors. Areas should then be as small as possible to limit the number of false positives.

In Table 3, we evaluate the influence of the sensor noise over the number and length of communications. As expected, the number of local detections is linear with the sensor noise. Long distance communications count is increasing with the sensor noise.

In tables 2 and 3, the number of long distance communications don't scale linearly with the correlation time and noise probability because of the no-alarm-

Sensor noise	readings	short-distance	long-distance
p=0	5400	0 (0%)	0 (0%)
p=0.002	5400	11 (0.2%)	0 (0%)
p=0.02	5400	94 (1.7%)	0 (0%)
p=0.1	5400	545 (10%)	56 (1.03%)
p=1	5400	5400 (100%)	210 (3.89%)

Table 3. Message count in DIAFORUS with noisy sensors (f=1Hz, p) and a correlation time of 180s. Experiment time of 30 minutes.

re-emission policy that is meant to limit the number of messages. In the case of Table 3, when p=1, we got an average of an alarm every 8.5 seconds even though the minimum intrusion time was set to 10 seconds. The 15% difference is due to the amendum to the no-alarm-re-emission policy that states than an alarm could be re-emitted if a new sensor was correlated with the on-going alarm.

With a relatively small correlation time (60s) and a relative low probability of false positive (2%), no alarms has been emitted during these 30 minutes. In normal situations, both the number and the average length of communications were lowered compared to non-collaborative approaches. Sensors' average noise determining the minimum number of messages that will be sent. In the worst case scenario studied, the number of messages is increased by 3.9% compared to the "Sink WSN" architecture while the average communication distance tended towards 0 hops which is an considerable improvement on large scale networks for both latency and power consumption.

4.2 Sensors Reputation Management

In the previous subsection, we investigated the influence of the average sensor's noise and the maximum intrusion duration on the average communication length and the number of messages. We saw that most of the time, the reasoning node was able to filter false positives as long as sensors aren't too noisy. In the case of a single faulty sensor, it is likely that the administrator will never receive any information concerning this sensor.

We created one scenario to validate both false positives and negatives. It features 2 areas which can be seen in Fig. 2.

Area 1 is composed of sensors 1, 2 and 3 and is meant to test false negatives. An intruder is repeatedly detected by sensors 1 and 2 but is never detected by sensor 3. No sensor noise was added to ease validation. After some time, nodes 1 and 2 are expected to have both a perfect false negative and a perfect false positive reputation while node 3 is expected to have a perfect false positive reputation but the lowest false negative reputation because it never contributed to any alarms.

Area 2 is composed of sensors 4, 5 and 6 and is meant to test false positives. An intruder is repeatedly detected by sensors 5 but is never detected by sensors 4 and 6. Again, no sensor noise was added to ease validation. After some time,

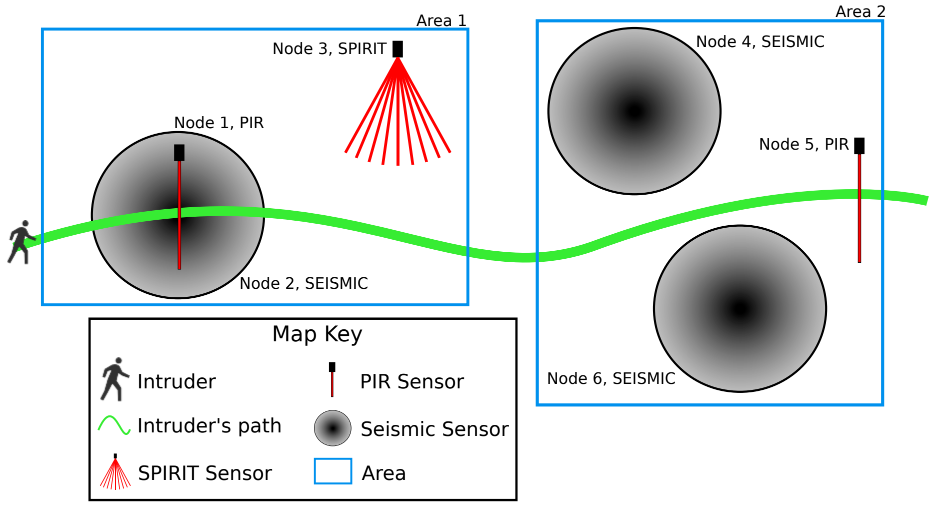

Fig. 2. Scenario validating the reputation. 2 areas, 6 sensors.

nodes 4 and 6 are expected to have both a perfect false negative and a perfect false positive reputation. However, node 5 is expected to have a perfect false negative reputation along with the lowest false positive reputation because none of the alerts it sent led to emission of an alarm.

Node ID	False positive reputation	False negative reputation
1 & 2	1 (18/18)	1 (283/283)
3	NaN (0/0)	0 (0/283)
4 & 6	NaN (0/0)	NaN (0/0)
5	0 (0/9)	NaN (0/0)

Table 4. Results of the reputation experiment found on Fig. 2

Results found in Table 4 are perfectly matching the expected reputation values. These results demonstrate the ability of our proposal to detect both the false positive and the false negative cases.

The scenario has then been changed to simulate the impact of noise (p=0.1, f=1Hz) on the sensors' reputation. The results are shown in Table 5.

With a noise probability (p=0.1, f=1Hz), sensors apparently sent an alert every 10.7s in average. With a correlation time of 20 seconds, it was pretty likely for all sensors to be participating in all alarms that were sent during these 30 minutes. This explains the false negative reputation of 1 for all three nodes.

As expected, the false positive reputation of sensors 1, 2 and 3 is quite low. It would be even lower if not all sensors were faulty or if not all sensors were so noisy because less alarms would have been sent.

Node ID	False positive reputation	False negative reputation
1	0.34 (57/168)	1 (119/119)
2	0.34 (57/167)	1 (119/119)
3	0.34 (57/169)	1 (119/119)

Table 5. Reputation of noisy sensors (p=0.1, f=1Hz) after 30 minutes and correlation time of 20 seconds

5 Future Work

Current research results have been obtained using a simulated network. However, porting this network on real nodes has been an on-going work for the past few months and we are currently preparing for final tests of the real network in the forthcoming weeks to validate the simulation results. The algorithms used by the simulation are the ones used on the real nodes. This means our proposal is feasible on standard sensor nodes.

We also would like to improve the reputation management by allowing the WSN administrator to report false positives and false negatives. This would increase the accuracy of the sensors' reputation.

A study should also be carried on to evaluate the power consumption cost of adding redundancy for the correlation node.

Finally, the influence of sensor density over the number of communications will be studied.

6 Conclusion

In this paper, we demonstrated a modality-agnostic collaborative detection of spatio-temporally correlated events. Correlating sensors' values inside the network instead of forwarding data to the gateway achieves a drastically lower power consumption by both reducing and localising communication. Our contribution enhances the state-of-the-art collaborative networks by abstracting sensors which allows the reasoning node to correlate ill-calibrated heterogeneous sensors. This abstraction also eases the creation of better auditing capabilities that overcome the problems usually found in collaborative networks. These capabilities can be used by both the administrator and the network itself to, for instance, automatically react to faulty sensors.

Acknowledgements

This work is part of research project called DIstributed Applications and Functions Over Redundant Unattended Sensors (DIAFORUS) [9], funded by the French National Research Agency (ANR).

References

1. Akyildiz, I.F., Su, W., Sankarasubramaniam, Y., Cayirci, E.: Wireless sensor networks: a survey. Computer Networks 38, 393–422 (2002)
2. Andersson, D., Fong, M., Valdes, A.: Heterogeneous Sensor Correlation: A Case Study of Live Traffic Analysis (2002)
3. Clouqueur, T., Ramanathan, P., Saluja, K.K., Wang, K.c.: Value-fusion versus decision-fusion for fault-tolerance in collaborative target detection in sensor networks. in proceedings of fourth international conference on information fusion p. pp. (2001), http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.116.6587
4. Croce, S., Marcelloni, F., Vecchio, M.: Reducing power consumption in wireless sensor networks using a novel approach to data aggregation. The Computer Journal 51(2), 227–239 (Mar 2008), http://comjnl.oxfordjournals.org/content/51/2/227
5. Frank, B., Shelby, Z., Hartke, K., Bormann, C.: Constrained application protocol (CoAP). https://tools.ietf.org/html/draft-ietf-core-coap-03, https://tools.ietf.org/html/draft-ietf-core-coap-03
6. He, Y., Li, M., Liu, Y.: Collaborative query processing among heterogeneous sensor networks. In: Proceedings of the 1st ACM international workshop on Heterogeneous sensor and actor networks. p. 25–30. HeterSanet '08, ACM, New York, NY, USA (2008), http://doi.acm.org/10.1145/1374699.1374705
7. Koomey, J., Berard, S., Sanchez, M., Wong, H.: Implications of historical trends in the electrical efficiency of computing. Annals of the History of Computing, IEEE 33(3), 46 –54 (Mar 2011)
8. Krishnamachari, B., Estrin, D., Wicker, S.B.: The impact of data aggregation in wireless sensor networks. In: Proceedings of the 22nd International Conference on Distributed Computing Systems. p. 575–578. ICDCSW '02, IEEE Computer Society, Washington, DC, USA (2002), http://dl.acm.org/citation.cfm?id=646854.708078
9. LaBRI: Diaforus. https://diaforus.labri.fr/doku.php (2010), https://diaforus.labri.fr/doku.php, [Online; accessed 25-May-2012]
10. Lazzerini, B., Marcelloni, F., Vecchio, M., Croce, S., Monaldi, E.: A fuzzy approach to data aggregation to reduce power consumption in wireless sensor networks. In: Fuzzy Information Processing Society, 2006. NAFIPS 2006. Annual meeting of the North American. pp. 436 –441 (Jun 2006)
11. Letian, H., Guangjun, L.: A reconfigurable system for digital signal processing. In: Signal Processing, 2008. ICSP 2008. 9th International Conference on. pp. 439 –442 (Oct 2008)
12. Nasipuri, A.: Collaborative Detection of Spatially Correlated Signals in Sensor Networks, vol. Proceedings of the 2005 International Conference on Telecommunication Systems Modeling and Analysis. Dallas, Texas (Nov 2005)
13. Ould-Ahmed-Vall, E., Heck Ferri, B., Riley, G.: Distributed Fault-Tolerance for event detection using heterogeneous wireless sensor networks. Mobile Computing, IEEE Transactions on PP(99), 1 (2011)
14. Phani Kumar, A.V.U., Reddy V, A.M., Janakiram, D.: Distributed collaboration for event detection in wireless sensor networks. In: Proceedings of the 3rd international workshop on Middleware for pervasive and ad-hoc computing. p. 1–8. MPAC '05, ACM, New York, NY, USA (2005), http://doi.acm.org/10.1145/1101480.1101491