IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 3, 2019, accepted August 8, 2019, date of publication August 22, 2019, date of current version September 12, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2936918

Data Communication Optimization for the Evaluation of Multivariate Conditions in Distributed Scenarios

FERNANDO LEÓN-GARCÍA[1], FRANCISCO J. RODRÍGUEZ-LOZANO[2], JOAQUÍN OLIVARES[2], AND JOSE M. PALOMARES[2], (Senior Member, IEEE)

[1]Advanced Informatics Research Group, Universidad de Córdoba, 14071 Córdoba, Spain
[2]Department of Electronic and Computer Engineering, Universidad de Córdoba, 14071 Córdoba, Spain

Corresponding author: Fernando León-García (fernando.leon@uco.es)

This work was supported in part by the Spanish Grant through the National Program Proyectos de I+D Retos Investigación" del Programa Estatal de I+D+i Orientada a los Retos de la Sociedad under Grant RTI2018-098371-B-I00, and in part by the Spanish Grant through the National Program Programa Operativo FEDER Andalucía 2014-2020 under Grant UCO-1263405.

ABSTRACT The current technological landscape is characterized by the massive and efficient interconnection of heterogeneous devices. Sensor networks (SNs) are key elements of this paradigm; they support the local loop, the collection and early manipulation of information. Among the applications of SNs, event detection is a well-explored topic in which strategies such as collaboration, self–organization, and others have been developed in depth. In this topic, the simplest and also most used event concept approach is the threshold-based event, which is usually integrated as part of the local sensor process. This paper addresses a different perspective by discussing the evaluation of multivariate Boolean conditions with distributed variables. We propose a new algorithm (Data Retaining Algorithm for Condition Evaluation, DRACE) that reduces packet traffic while preserving time accuracy in event calculation on an adaptive approach. To facilitate understanding of DRACE, a case study is presented in the context of a logical simile titled The Problem of a Proper Defense. The algorithm supports parameters that affects the compromise between accuracy and traffic savings. To analyze its performance, 9000 executions of the algorithm have been performed. 9 configurations tested on a repository of 1000 triads of signals randomly generated. Focusing on the most accurate configuration, 99% of executions are error-free, and the number of packets is reduced by 40% on average, being between 30 and 50% in 68% of cases.

INDEX TERMS Wireless sensor networks, event detection, information filtering.

I. INTRODUCTION AND MOTIVATION

Nowadays, the technological landscape ranges from large servers in the cloud to small processors in objects across our environments. Classic network technologies become obsolete to face the challenges posed by the massive distribution of hosts and their high heterogeneity [12]. The research topic *internet of things* (IoT) represents the most global vision of this reality. However, the scientific community has adopted other research topics that refer to more specific approaches, such as pieces of the same puzzle (e.g. *[wireless] sensor networks* ([W]SN), *edge computing* or *fog computing*) [32].

Adopting IoT terminology, at the opposite edge of the cloud there are things characterized by high heterogeneity and severely constrained resources [29]. Regardless of the goal of the deployment, from the vast variety of scientific proposals related to this issue, two trends in technological development are revealed: I) the optimization of resources of both the devices and the network; and II) the approximation (as far as possible) of the process to the sources of information, as opposed to the methodological tradition of processing everything in the cloud [3].

As part of the ecosystem of things, the wireless sensor networks (WSN) provide an inexpensive way to capture information from the environment [12]. The two most common uses of WSNs are: 1) to collect data from the environment and route it to one or more sink nodes [7], [13], [15]; and 2) to detect and report events [9], [24], [31]. Due to the scarcity of resources available in the sensors, both applications require optimization, which is the first trend noted

The associate editor coordinating the review of this article and approving it for publication was Yi Fang.

above. The detection of events, moreover, involves bringing processing closer to the sensors.

WSN event detection is a broad topic that has been approached from many perspectives. The scientific proposals can be classified according to many considerations. An important consideration is the concept of event itself, which in its broadest sense is a relevant occurrence circumscribed in time and place. However, the detection process requires a technical definition in practice that allows us to classify event detection systems according to criteria such as architecture or method [1], [14], [17], [31], [35].

Threshold detection event is one of the most simple and common methods [17]. Threshold values are suitable for a lot of applications, for example, fire detection, detection of flooding, or generally other applications in which a sensor can detect a critical boundary of the measured value [14]. The current trend in event detection focuses on the use of increasingly sophisticated techniques such as artificial intelligence, data mining, or fuzzy logic [16]. This results in a lack of interest in threshold-based detection techniques, which are widely used in industry, distributed control networks, alarm systems, or in the early stages of more complex computational methods, such as rule-based systems or complex event processing [4].

The general conception of threshold-based event detection refers to local evaluation architectures, in which sensors evaluate the condition with their own signals and notify when it is met [14]. However, the use of thresholds to define events covers more complex scenarios for which it is worthwhile to propose strategies to improve the process. The use of non-static thresholds or the satisfaction of conditions that depend on multiple network distributed variables are examples of this potential complexity.

The motivation behind this paper is to contribute to the development of this topic, introducing an efficient strategy to deploy networks with a double objective: to collect the data generated by the sensors, and to detect and notify in time threshold-based events expressed as conditions with multiple distributed variables. In order to easily illustrate the intrinsic aspects of this proposal and to favour the understanding of the motivation of this work, a warlike simile is introduced in the following sub-section that incorporates all the relevant aspects to be addressed from another logical context.

A. THE PROBLEM OF A PROPER DEFENSE

A warlord controls a fortress that is surrounded by a moat with 3 drawbridges. His spies reveal to him that his enemies are settling in 3 camps, and that they are trying to recruit soldiers to attack all the bridges simultaneously, one for each army formed. The information is so accurate that it reveals which gate will be attacked by each army at the time of the attack, which will start the next morning to get 10,000 soldiers for the cause counting all camps. However, the reputation of the military is so well known that gathering such a number of soldiers is not being easy, because in addition to new recruits there are also deserters in all enemy armies, making it impossible to accurately estimate when the battle will take place.

To prepare the defense, the warlord has 3000 soldiers, a group of engineers, and 150 defensive devices distributed among the gates, including traps, ballistas, oil boilers, etc. In his experience, one defensive machine equals 50 soldiers. Assuming that each soldier brings a force of 1, the warlord has a defensive force of 10500 against an offensive force of 10000, so he thinks that a good organization will give him victory. Thus, the experienced military decides to proceed as follows. The soldiers will be sent to the different gates in groups according to the force of the expected attack. For this part of the plan, the military requires to know the exact proportion of the enemy armies the same day of the attack, since the soldiers can be mobilized quickly. The engineers, however, will be in charge of redistributing the mechanical traps, deploying a defensive force proportional to the expected offensive one. For this task, on the other hand, the warlord needs to be informed of the state of the enemy armies, because engineers cannot install or uninstall more than 1 defensive devices in each gate per day, so their work cannot be done at the last moment.

To solve the problem of information, a scout is sent to each enemy camp with a command and a carrier pigeon. The command states: ''Hide at the edge of the camp and count the number of soldiers daily. If this is different from what I know, send me the pigeon with the data by nightfall. The pigeon will be returned with or without new orders.''

As planned, the military was rigorously informed of the composition of the enemy armies, and preparations were initiated at each gate in just measure. However, after the first 10 days one of his explorers was captured. Apparently, bored due to the delay of the battle, the soldiers of the enemy camp noticed the regular flight of the bird and combed through the nearby forest. It was necessary to convene a meeting of councillors to decide on another strategy that would allow the warlord to be properly informed and to protect the explorers as far as possible.

Once the advisors had met and the problem was exposed, even without knowing how to proceed, a conclusion was reached unanimously: to achieve this, the warlord could not be so well informed. As one of them said, ''you can't have the cake and eat it''.

B. CONTEXTUALIZATION

Translated into an easily understandable logical context, the Problem of a Proper Defense raises the need to consider the ''price'' of information. By replacing pigeons by network packets, explorers by sensors, and the castle by the sink node, the proposed scenario corresponds to a sensor network with two simultaneous purposes: the detection of the threshold-based event expressed by inequality $v_1(t) + v_2(t) + v_3(t) \geq 10000$ (for moving the soldiers), and the continuous monitoring of involved variables (for preparing the defensive elements).

Sensor networks are often very limited in energy, so traffic avoidance is a priority. However, if the network is designed to detect an important condition, such as an alarm, a conflict arises between the accuracy of the detection and the traffic required. Hence the ''price'' of information.

A common solution is to evaluate the condition locally and send a notification when the event takes place. However, if some tracking of system status is not available, an unexpected alarm notification may be useless. This situation happens when the response to the event requires some preparation. For example, a network of fire detection sensors with a fire suppression system that strategically mobilizes water reservoirs. In this case, the system requires a certain resource to be applied proportionally to the imminence of the event.

The cost of maintaining precision in anticipation of an event is not affordable in terms of energy or network traffic. For this reason, the motivation of this work is to provide a logical vision and propose a simple algorithmic solution. With this solution, the sensors participate in an adaptive tracking process, and the accuracy is proportional to the event occurrence estimation. Thus, if the event is far from being fulfilled, the sensor network will remain in a latent mode, in which consumption and traffic are not a problem.

This work involves a new threshold-based event approach, proposing the scenario of multivariate conditions. We present an algorithm that significantly reduces network traffic, which is widely evaluated by simulation and random signals. The proposal involves the concept of *latent sensor network*, in which the sensors remain conveniently silent as long as the condition in question remains far from being met. This concept may be important in wide coverage applications for alarm purposes, such as fire detection.

The rest of the document is organized as follows. Section II (Background) deals with the theoretical framework of the work, it describes the current panorama of event detection and traffic reduction in WSN, specifically delving into threshold-based events; Section III (Mathematical Approach and Algorithmic Proposal) details the mathematical model behind the Problem of a Proper Defense described above, also details the proposed algorithm (DRACE) to address the problem. Section IV (Case Study) uses DRACE to solve the proposed problem, describing in detail the algorithm operation. Section V (Experiments) addresses an experimental methodology for analyzing the DRACE performance, describing the configurations, data, and metrics. Finally the results are discussed. Section VI (Conclusions) notes the contributions of the work and concludes on the basis of the results. Finally, Section VII outlines the direction of future efforts to continue this work.

II. BACKGROUND

In (W)SNs, event detection is perhaps the most widespread application, and saving traffic is one of the most common optimization goals. The publications on both topics are very varied, and previous considerations are required to specify the scope of the contributions.

In its broadest definition, an event is something that happens in a space and time. This simple, general and ambiguous definition raises more questions than answers: ''*is an event model available or is it an anomaly with respect to a normal regime?*''; ''*is it deterministic?*''; ''*does the event occur punctually in a space, or is it a phenomenon that spreads?*''; ''*is immediate detection required or is it enough to detect it in post–processing?*''. The answers to these and other questions give rise to a wide range of approaches. Among other things, they determine aspects such as collaboration between nodes, redundancy in detection, convenience of clustering, or the suitability of applying machine learning techniques.

In terms of traffic savings, the variety of contributions is related to the part of the system architecture being optimized. Considering a general architecture, from sensing to processing there are numerous processes to be considered. Duty cycle scheme, sampling, in–node processing, queueing/packing/sending policies, routing, clustering, priorities, remote processing, etc. As in event detection, the literature caseload is vast.

This work aims to contribute to both fields by means of a data queuing strategy that optimizes traffic when the event to be detected can be modeled as a multivariate Boolean condition. The proposed scenario fits into different research topics, such as data gathering schemes, event detection, adaptive sampling, or traffic saving among others. And its application is interesting in different areas, like industrial control, domotics, surveillance, or alarm systems.

Since a comprehensive review of the state of the art is not the aim of this work, the following subsections narrow the scope and describe important contributions that have provided inspiration and background for this paper.

A. DISTRIBUTED EVENT DETECTION

Although event detection is a concept that transcends the underlying technology, sensor networks have been the main technological framework in the development of the topic.

Event detection in sensor networks is a widely explored and developed research topic. Relevant aspects and techniques to detect events are usually classified according to the criteria described in [17], summarized in [28]. Situational dependence, criticality of application, numerous and diverse data sources, and network topology are key aspects to be considered. Regarding the methods, they are usually classified to: statistical, probabilistic, or artificial intelligence/machine learning based methods.

According to the conceptual framework exposed above, this work is a traffic–aware model–based method. In this work, a mathematical expression, which provides a Boolean result, has to be evaluated accurately in terms of time triggering. Since the result of the proposed model is Boolean and defined by mathematical expressions, proposals related to threshold–based event detection are considered as background to this work.

Threshold–based approaches are classified within the statistical model–based methods. Essentially, it consists of

reporting when a parameter exceeds a given threshold. This model is considered to be a technique with very low computational complexity. Therefore, it is the most widely used method for simple event detection [9], [17].

There are many examples of the use of threshold–based events for the detection of alarm conditions in specific situations, such as wildfires [5], [22], volcanic activity [34], or perimeter invasion [18].

The idea of comparing a value with a threshold is a well–established mechanism. Most related publications normally use threshold–based events, processed at the sensor level. After that, the measured magnitude is compared with a static or dynamic threshold. Finally, the result is sent to a network that exploits other WSNs features, such as redundancy, dynamism, or collaboration. For example, [19] uses threshold–based events along with spatial correlation and redundancy to provide fault–tolerant distributed detection of events. Another example is [6], which proposes a dynamic double threshold technique for detecting abnormal events by aggregation.

Both previous proposals are typical examples in which the threshold is used to detect significant variations on a single variable, not dealing with any composition of multiple variables. This latter fact rises a question about the complexity of threshold–based event models: "*What happens if the threshold is applied to a multivariate function?*" The issues arising from this question have not been addressed from the point of view of sensor constraints and are the main focus of this work.

B. SAMPLING SCHEMES

The sampling and notification of distributed variables has been approached from the automatic control. Magnitude-based sampling [25], Lebesgue sampling [2], or Send–on–Delta [26] are different names for the same basic principle: successive samples of a signal are not triggered by time criteria, but by signal variation. There is an extensive literature on the subject, with proposals that integrate predictive methods in the sampling trigger criterion, trying to maximize sample reduction or adapting the process to the intended objectives [27], [36].

These sampling strategies provide an interesting approach to traffic aware methods by reducing the number of data needed for monitoring tasks. The potential of such methods when contextualising the use of data in processes with Boolean results was demonstrated in [21], in which a precision, recall, and traffic savings analysis is presented when evaluating conditional expressions with signals sampled by magnitude criteria. The study also proposes metrics to measure the effect of delays in the detection of transitions in the resulting signal, as a consequence of the magnitude–based sampling scheme. Here, the proposal is based on the conclusions of that study, and proposes an adaptive algorithm that takes advantage of the potential traffic savings by minimizing the introduced error.

C. TRAFFIC AWARE STRATEGIES IN WSNS

One of the challenges of the WSNs is the avoidance of network collapse. WSNs typically deploy many nodes that communicate using low-bandwidth wireless technology. This combination entails a problematic scenario due to potential congestion. For this reason, contributions related to traffic avoidance techniques are of great interest. There are two key concepts that appear frequently in the related proposals: compression and prediction.

Compression techniques are applied to achieve a more efficient use of resources, especially in cases of scarcity. Reference [30] delves into the different approaches to compression in WSNs, classifying them into: a) sampling compression, when the number of sensor sample acquisitions is reduced; b) data compression, with which the number of bits in the data stream is reduced; and c) communications compression, which aims at reducing the number of accesses to the network.

On the other hand, prediction is a powerful tool that is often related to compression, because one can avoid reading/processing/sending what can be predicted with some certainty. Reference [8] approaches this issue from two perspectives: 1) where the prediction procedure is carried out (essentially, sink node, sensor node, or both); and 2) which prediction technique is used (statistic, probabilistic, machine learning, etc.).

According to the previously described concepts, the purpose of this work can be classified as belonging to the communication compression topic. Furthermore, the proposed method is related to forecasting techniques on time signals at a sink node. Some representative examples in this scope are [10], [23], [37]. Nevertheless, all these works deal with reducing network traffic by filtering signal data in the sensor domain (usually, integer or real numbers). None of these proposals take into account the final use of the data for event detection. Therefore, these works reduce the transmissions to the sink with a criterion exclusively focused on minimizing the error in the magnitude of the involved data. However, the most relevant fact for an event detection system is not the amount of error in the magnitude, but the final Boolean response. Thus, the compression may be more aggressive when the signal is far away from the threshold, and hence, traffic savings would be larger. Moreover, as the proposals are focused on minimizing one data stream, they are not suitable for detecting events based on multiple variables.

D. DISCUSSION ABOUT THE BACKGROUND

In the previous subsections, the most relevant scientific background has been introduced. The classification of the techniques in several different scopes has been included. All these methods have provided an outline of the scientific field related to this work.

The definition of "event" according to threshold–based detection is stable in all the revised articles. It only applies the Boolean evaluation on only one variable, either in sensor

nodes or in head–cluster ones. There has not been found any reference in which the Boolean evaluation based on thresholding had been applied to functions composed of several different variables. This is a new challenge that this work aims to address.

There are many mechanisms to reduce data sent through the network. However, that reduction of data is achieved without any knowledge about the final use of that data. Therefore, the data sent by the sensors is *decontextualized*. In this work, the final use of the data is known in advance. Therefore, a completely different approach may be applied in this work.

The main background is the D2R–TED model [21]. This model provides the mathematical foundations to allow the data management of multivariate functions with threshold–based event detection. Results shown in that previous work are promising. However, in that work, no algorithm for dynamic adjustment of the threshold was proposed.

III. MATHEMATICAL APPROACH AND ALGORITHMIC PROPOSAL

This section deals with the algorithmic proposal of this work. Mathematical expressions and algorithms requiring arrays and matrixes are used. For a better understanding, the notation is specified here:

A. NOTATION FOR PSEUDOCODE AND EQUATIONS

Array declaration:

- $x \in D^n / n \in \mathbb{N}$ is an array of n elements defined in $D + \{\varnothing\}$, so that $x[i] \in D + \{\varnothing\}$ is the ith element, $i \in \{1, 2, \ldots n\}$.
- $x = \{c\} \times n, c \in D, n \in \mathbb{N}$ is an array of n values equal to c, so $x \in D^n$.
- $x = \{f(e), \forall e \in A\}, A \in {D_1}^n, f : D_1 \rightarrow D_2, n \in \mathbb{N}$ generates an array of n elements defined in D_2 by mapping A elements with f function. If the generator array contains null values, these are transmitted to the generated array. So $x \in D_2 + \{\varnothing\}^n$.
- $x = \{e \in A / c(e)\}, A \in D^n, f : D \rightarrow \{true, false\}$ generates an array with those elements of A for which f is true. So $x \in D^m, m \leq n$.
- All elements of an array whose value has not been explicitly declared are initialized at null value. In other words, $x \in D^n \equiv \{\varnothing\} \times n$, with the difference that the first declaration specifies the mathematical domain of the elements.

Array operation:

- If $x \in D^n$, $x[-1]$ is the last non-null element in the array, or null if there is none.
- If $x \in D^n$, $\#x$ is the number of non-null value elements contained in x, so $\#x \in N + \{0\} \leq n$.
- If $x \in D^n$, $x \leftarrow v$ sets the value v at the position of the first null element contained in x. If there are no null values, all values are shifted to the left, discarding the first and leaving the last void, where v is set.
- If $x \in D^n$, IndexOf(x, v) is the position of the first element of x whose value is v, resulting in $\varnothing$ if none exist. So IndexOf$(x, v) \in \{1, 2, \ldots n\} + \{\varnothing\}$.
- If $x \in R^n$, Max(x) and Min(x) return the highest and lowest value contained in x, respectively. If the array is empty, both functions will return null.

Matrix declaration:

- $x \in \mathrm{M_{n\times m}}(D)/n, m \in \mathbb{N}$ is a matrix of n rows and m columns of elements defined in $D + \{\varnothing\}$, so that:
  - $x[i][j] \in D + \{\varnothing\}$ is the element at the row $i \in \{1, 2, \ldots n\}$ and column $j \in \{1, 2, \ldots m\}$.
  - $x[i][*]$ or simply $x[i]$ is the array of m elements (one per column) corresponding to row $i \in \{1, 2, \ldots n\}$, so $x[i] \in \{D + \{\varnothing\}\}^m$.
  - $x[*][j]$ is the array of n elements (one per row) corresponding to column $j \in \{1, 2, \ldots m\}$, so $x[*][j] \in \{D + \{\varnothing\}\}^n$.
- $x \in \mathrm{M}_{n\times *}(D)/n \in \mathbb{N}$ is a matrix of n rows and undefined number of columns of elements defined in $D + \{\varnothing\}$.

Indexing precedence in matrixes:

- If $x \in \mathrm{M_{n\times m}}(D^h)/n, m, h \in \mathbb{N}$ is a matrix of n rows and m columns of arrays of h elements defined in D. So that $x[r][c][i] \in D + \{\varnothing\}$ is the ith element of the array in row $r \in \{1, 2, \ldots n\}$ and column $c \in \{1, 2, \ldots m\}$ of the matrix, where $i \in 1, 2, \ldots h$.

Open-ended arrays and matrixes:

In order to specify algorithms that operate undefined length time series, this nomenclature is proposed to declare open sized structures.

- $x \in D^*$ is an unspecified size empty array of elements defined in D. The length of the array is increased by inserting elements at the end with the $\leftarrow$ operator. In this case the array elements never shift to the left, they just grow.
- $x \in \mathrm{M_{n\times *}}(D)/n \in \mathbb{N}$ is a matrix of n rows and undefined columns of elements defined in $D + \{\varnothing\}$. initially the number of columns is 0, and the way to increase it is to add elements to the vectors row with $\leftarrow$ operator. The number of columns in the matrix is determined by the number of elements in the longest row vector. And they increase in size using the $\leftarrow$ operator, setting the rest of the elements of the new column to $\varnothing$. For example:
  1) $x \in \mathrm{M}_{2\times *}(D)$
  2) $\#x[1]$ is 0
  3) $x[1] \leftarrow 1$
  4) $x[*][1]$ is $\{1, \varnothing\}$
- $x \in \mathrm{M}_{*\times m}(D)/m \in \mathbb{N}$ is a matrix with undefined rows and m columns of elements defined in $D + \{\varnothing\}$. In this case the rows are created by adding elements to the column vectors.
- Matrixes must have at least one dimension defined.

B. MATHEMATICAL APPROACH TO THE PROBLEM OF A PROPER DEFENSE

From the mathematical point of view, the Problem of a Proper Defense can be modelled from a set of parameters and variables, such as those proposed below.

$TDS \in \mathbb{N}$ Total Defense Soldiers.
$TDD \in \mathbb{N}$ Total Defense Devices.
$E \in \mathbb{N}$ number of explorers (one for each enemy camp).
$X \in \mathbb{N}$ Ith day since the implementation of the strategy in which the battle occurs. Therefore:
$X = \text{Min}\left(\left\{i \in \mathbb{N} / \sum AS[*][i] \geq 10^4\right\}\right)$.
$\lambda \in \mathbb{R}$ Defensive factor of defense devices.
$\varepsilon \in \mathbb{N}$ Work pace of engineers.
$AS \in M_{E\times *}$ Number of Attack Soldiers per camp and day. So that $AS[i][d]$ is the number of soldiers in camp i at day d.
$AS' \in M_{E\times *}$ Last known value of number of enemies in camp i on day d. So that $AS'[i][d]$ is information that is known day d about the number of soldiers in camp i.
$DD \in M_{E\times *}(\mathbb{N})$ Number of defensive devices per gate and day. So that $DD[i][d]$ is the number of defensive devices in gate i at day d.

The temporal granularity of the story is daily, so the independent variable d represents the day number.

According to military numbers, there are 3000 soldiers and 150 defensive devices, so $TDS = 3000$ and $TDD = 150$. The strength of each defense device in battle is 50 times the defensive strength of a soldier, then $\lambda = 50$. It is also considered that engineers cannot manipulate more than 1 defensive device per day, then $\varepsilon = 1\frac{units}{\text{day}}$. These two parameters model the requirement to track monitored time series, as defensive devices are key to winning the battle, for which they must be in place on the day of the fight. In other words, increasing λ or ε means increasing the margin of error in the placement of war machinery.

Assuming battle takes place on day X, the defense of the castle requires the victory of the defensive forces at all gates, (1) expresses this condition considering the Boolean values true $= 1$ and $false = 0$. Fig. 1 is a geographical representation of the military forces according to this model.

$$\text{V}(X) = \begin{cases} true, & \text{if} \sum_{i=1}^{E} (DS[i][X] + \lambda \cdot DD[i][X] \geq AS[i][X]) = E \\ false, & \text{otherwise} \end{cases} \quad (1)$$

Although the above expressions define the state of the conflict on day X, the evolution of the defensive forces depends on the information received by carrier pigeons (AS'), which does not necessarily correspond to the actual data (AS). With regard to the movement of soldiers, the warlord intends to distribute them proportionally to the offensive forces on the day of the attack (day X). For this action to take place it is necessary that $\sum(AS'[*][X]) \geq 10^4$. Equation (2) expresses this fact.

FIGURE 1. Artistic representation of the problem of a proper defense, designed using Inkarnate® web tool (http://www.inkarnate.com).

$$DS[i][d] = \begin{cases} \dfrac{TDS \cdot AS'[i][d]}{\sum(AS'[*][d])} & \text{if } \sum(AS'[*][d]) \geq 10^4 \\ 0 & \text{otherwise} \end{cases} \quad \forall i \in 1, 2, ..E \quad (2)$$

With respect to defensive devices, their relocation is decided on the basis of the daily information received. Algorithm 1 models this process assuming that defensive devices were not operational before implementing this strategy.

So much for the model that describes the actions taken in the castle. The problem of the cost of information must be addressed on the side of the explorers. The number of pigeon flights has to be reduced at least as long as the battle is not imminent. The council solution is described below.

C. THE COUNCIL SOLUTION

After a long discussion, the councillors agreed to proceed as follows:

On the one hand, explorers will continue to record the count of enemy soldiers on a daily basis, but it is neither appropriate nor necessary to send it every day. Each explorer will retain its data until it sufficiently differs from the last value of the last scroll sent. In this way, the information will be sent when: a) a variation greater than a threshold is recorded, or b) no more values fit on the scroll.

On the other hand, the key question of this strategy is which threshold would be appropriate in each case and how the explorers would know. It was concluded that it should be the military command in the castle who established the thresholds with the latest known information, in addition to preparing the defenses. After all, it is there where the conflict is seen as a whole.

In order to tackle this task an information channel from the castle to each explorer was required. For this purpose, pigeon

Algorithm 1 Update_DD(d)

Require: ε, DD, TDD, AS'

1: works $= \varepsilon$
2: done $= \{0\} \times E$
3: **if** $d = 1$ **then**
4: to_do $= \left\{ \frac{AS'[i][d]}{\sum(AS'[*][d])} \cdot \text{TDD} \right\} \forall i \in \{1, 2, \ldots E\}$
5: free_devices $= TDD$
6: **else**
7: to_do $= \left\{ \frac{AS'[i][d]}{\sum(AS'[*][d])} \cdot \text{TDD} - DD[i][d-1] \right\} \forall i \in \{1, 2, \ldots E\}$
8: free_devices $= TDD - \sum DD[*][d-1]$
9: **end if**
10: **while** works > 0 **and** Max($\{|x|, \forall x \in$ to_do$\}$) > 0 **do**
11: **if** free_devices > 0 **then**
12: $i =$ IndexOf(to_do, Max($\{|x|, \forall x \in$ works$\}$))
13: **else**
14: $i =$ IndexOf(to_do, Min(to_do))
15: **end if**
16: $x = \frac{\text{to\_do}[i]}{|\text{to\_do}[i]|}$
17: done$[i]$ = done$[i] + x$
18: to_do$[i]$ = to_do$[i] - x$
19: free_devices = free_devices $- x$
20: works = works $- 1$
21: **end while**
22: **for** $i \in \{1, 2, \ldots E\}$ **do**
23: **if** $d = 1$ **then**
24: $DD[i] \leftarrow$ done$[i]$
25: **else**
26: $DD[i] \leftarrow DD[i][d-1] +$ done$[i]$
27: **end if**
28: **end for**

return flights would be used. In this way, explorers who send information can receive a new threshold as a response if appropriate.

Regarding the calculation of thresholds, the following procedure was outlined:

1) To facilitate the calculation of thresholds, they were limited to a series of predefined values, allowing fine-tuning the surveillance process of enemy camps according to a scale of granularity in the accounts. The proposed scale was 1, 25, 100, 200, and 300. All explorers would have threshold 1 initially.
2) For each explorer, a data record for each scale value would be prepared. Each scroll received would be copied value by value to all the records following this rule: for the record corresponding to threshold X, only those values that differ a minimum of X from the last recorded value would be noted.
3) Every day, after all information has been received and recorded in accordance with this rule, defense devices preparation would be coordinated with the most recent information available. Also, if the battle condition were met, soldiers would be proportionally placed at each gate in preparation for the imminent conflict; if not, all explorers who have sent pigeons would be candidates for changing their threshold value.
4) The calculation of thresholds for candidates is an iterative process in which the battle condition is evaluated by changing known information for expected information, with the intention of finding a safe threshold setting so that the castle would not be surprised in battle. The calculation would have the following steps:
   a) For each record of each explorer, the maximum expected variation in the next value would be forecast using the last 10 values, if available.
   b) For each candidate, each maximum expected variation per threshold added to the last known value would suppose a possible configuration. For non-candidate explorers there would only be one configuration, corresponding to their current thresholds.
   c) Starting from the combination of configurations with higher thresholds, the battle condition would be re-evaluated. If the condition were met, one of the configurations would be replaced by the one of the next lower threshold. This process would be repeated until: a) a combination is found that does not meet the condition, or b) the thresholds can no longer be lowered.

Two aspects of the strategy were discussed at the end, as it was felt that there were many options on the table. At the one hand, the method of forecasting the maximum variation to be expected. At the other hand, the selection of the candidate to reduce the threshold in the iterative process of searching for the best combination of thresholds.

Finally, a simple method was decided by consensus:

1) The maximum expected variation is K times the standard deviation (σ) of the queue values, and to be on the safe side he proposed $K = 3$.
2) The best candidate to be reduced will be the one with the greatest variation present in the forecast.

D. ALGORITHMIC NOTATION

Let us define the variables that complete the model according with the strategy of the council.

$DPM \in \mathbb{N}$ Maximum number of data per message (scroll).
$\Delta = \{\delta_0, \delta_1, \ldots \delta_{n-1}\}, n \in \mathbb{N}$ Set of possible thresholds.
Q Number of values considered to forecast the maximum variation for each threshold record.
K Multiplier factor of the standard deviation when predicting the next maximum value.

For better understanding, all notations are summarized in Table 1.

Algorithm 2 (Explorers) consists of a read and send loop on a daily basis. The reading and annotation of the collected value is always done (L: 4). Sending only occurs if required (L: 5), depending on the δ value and the free slots in the scroll

Algorithm 2 Explorer

Require: DPM, $\Delta[0]$
Initialisation :
1: $\delta = \Delta[0]$
2: scroll $\in \mathbb{N}^{\text{DPM}}$
3: last $\in \mathbb{N} = \varnothing$
LOOP Process (daily)
4: scroll $\leftarrow$ Count_Soldiers()
5: **if** last $= \varnothing$ **or** #scroll $=$ DPM **or** $|\text{scroll}[-1] - \text{last}| \geq \delta$ **then**
6: last $=$ scroll$[-1]$
Reporting interval
7: Send_Pigeon(scroll)
8: **while** #scroll > 0 **do**
9: scroll $\leftarrow \varnothing$
10: **end while**
Out of reporting interval
11: Wait_For_Pigeon($\&\delta'$)
12: **if** $(\delta' \neq \varnothing)$ **then**
13: $\delta = \delta'$
14: **end if**
15: **end if**

TABLE 1. Summary of notations.

TDS	Total Defense Soldiers.
TDD	Total Defense Device.s
E	Number of Explorers.
X	Day of the battle.
λ	Defensive factor of defense devices.
ϵ	Work pace of engineers.
AS	Number of Attack Soldiers per camp and day.
AS'	Last known value of numbers of enemies per camp and day.
DD	Number of defensive devices per gate and day.
DPM	Maximum number of data per message.
Δ	Set of possible threshold.
Q	Number of values considered for forecast (queues lenght).
K	Multiplier factor of the standard deviation.

(packet). After each sending, there is a wait for a possible message with a new value of parameter δ (L: 11).

Algorithm 3 (Castle) consists of a daily iterative process with three well-differentiated parts. The first part is called the *reporting interval* (L: 7–20). In this part, the packets coming from the sending explorers are processed, adding the containing data in queues, corresponding to the possible values of δ (Δ). These queues are required by the forecasting method. The second part (L: 21–28) uses the most up-to-date information available to organize the defenses, finalizing the algorithm if the battle condition is met (in which case the soldiers would be mobilized to the gates). The third part (L: 29–63) is carried out if the battle condition is not met. This part is an iterative process in which the best δ value is determined for each explorer. In this process, the aim is to maximize the δ-values without incurring the risk of suffering the attack without having foreseen it. For this purpose, the algorithm uses the queues filled in the first part to assess the risk of meeting the battle condition with each δ and explorer.

Algorithm 3 Castle

Require: E, Q, Δ, K
Initialisation :
1: i_delta $= \{1\} \times E$
2: AS$' \in \text{M}_{E\times *}(\mathbb{N})$
3: queue $\in \text{M}_{E\times \#\Delta}(\mathbb{N}^{\text{Q}})$
4: fc_dev $\in \text{M}_{E\times \#\Delta}(\mathbb{R})$
5: day $= 1$
LOOP Process (daily)
6: candidates $\in \mathbb{N}^{E}$
Reporting interval
7: **while** Received $scroll_e$ from explorer e **do**
8: candidates $\leftarrow e$
9: **for** $x \in$ scroll$_e$ **do**
10: **for** $i_\delta \in \{1, 2, \ldots \#\Delta\}$ **do**
11: $\delta = \Delta[i_\delta]$
12: **if** $\#queue[e][i_\delta] = 0$ **or** $|queue[e][i_\delta][-1] - x| \geq \delta$ **then**
13: $queue[e][i_\delta] \leftarrow x$
14: **if** $\#queue[e][i_\delta] =$ Q **then**
15: fc_dev$[e][i_\delta]$ $=$ Std_Deviation(queue$[e][i_\delta]$)
16: **end if**
17: **end if**
18: **end for**
19: **end for**
20: **end while**
Out of reporting interval
21: **for** $e \in \{1, 2, \ldots E\}$ **do**
22: AS$'[e] \leftarrow$ queue$[e][1][-1]$
23: **end for**
24: Update_DD(day)
25: **if** $\sum$(AS$'[*][$day$]) > 10^4$ **then**
26: Prepare_To_Battle()
27: **return**
28: **end if**
29: $\delta_{\text{conf}} \in \mathbb{N}^{E}$
30: dev $\in \mathbb{R}^{E}$
31: c $\in \text{M}_{E\times 2}(\mathbb{R})$
32: **for** $e \in \{1, 2, \ldots E\}$ **do**
33: **if** $e \in$ candidates **then**
34: $i_\delta = \text{Max}(\{1, \#\text{fc\_dev}[e]\})$
35: $\delta_{\text{conf}}[e] = i_\delta$
36: dev$[e]$ $=$ fc_dev$[e][i_\delta]$
37: **else**
38: $i_\delta =$ i_delta$[e]$
39: $\delta_{\text{conf}}[e] = 0$
40: **end if**
41: **if** fc_dev$[e][i_\delta] \neq \varnothing$ **then**
42: c$[e][1] =$ AS$'[e][-1] +$ K $\cdot$ fc_dev$[e][i_\delta]$
43: c$[e][2] =$ AS$'[e][-1] -$ K $\cdot$ fc_dev$[e][i_\delta]$
44: **else**
45: c$[e][1] =$ AS$'[e][-1]$
46: c$[e][2] =$ AS$'[e][-1]$
47: **end if**
48: **end for**

Algorithm 3 *(Continued.)* Castle

while $\sum(\delta_{\text{conf}}) > \#\text{candidates}$ **and** $\exists\, I \in \{1,2\}^E / \sum(\{c[e][I[e]], \forall e \in \{1,2,\ldots E\}\}) \geq 10^4$ **do**
  $i = \text{Max}(\{dev[j], \forall j \in \{h \in \text{candidatos}/\delta_{\text{conf}}[h] > 1\}\})$
  $e = \text{IndexOf}(\text{dev}, i)$
  $\delta_{\text{conf}}[e] = \delta_{\text{conf}}[e] - 1$
  $i_\delta = \delta_{\text{conf}}[e]$
  $\text{dev}[e] = \text{fc\_dev}[e][i_\delta]$
  $c[e][1] = AS'[e][-1] + K \cdot \text{fc\_dev}[e][i_\delta]$
  $c[e][2] = AS'[e][-1] - K \cdot \text{fc\_dev}[e][i_\delta]$
end while
for $e \in$ candidates **do**
  if $\text{i\_delta}[e] \neq \delta_{\text{conf}}[e]$ **then**
    $\text{i\_delta}[e] = \delta_{\text{conf}}[e]$
    $\text{Send\_Pigeon\_Back}(\Delta[\text{i\_delta}[e]])$
  end if
end for
$\text{day} = \text{day} + 1$

E. GENERALISATION: DRACE ALGORITHM

Generally speaking, the algorithm consists of collecting data before sending it. How long the data is accumulated depends on the condition satisfaction forecast. In a network context, the castle and the explorer play the roles of sink node and sensor node, respectively. Due to this retaining mechanism, the updating of the data in the sink node is exposed to opportunistic periods of misinformation, resulting in a less accurate time signal when convenient. For all of this, the algorithm is called ***DRACE***, acronym of Data Retaining Algorithm for Condition Evaluation.

An interesting aspect of the algorithm is that the data sending criterion is expressed in magnitude by means of δ. Each possible δ value is associated with a queue of Q elements. Due to the filling process, the higher the δ, the greater the time interval represented in the associated queue (as can be observed in Fig. 2, which illustrates the queues filling process). However, the duration of this interval depends on the behavior of the signal, and will adapt to it as the queue is updated.

The decision to increase or decrease δ comes from limiting the uncertainty of the next value for the current δ-queue. The prediction method takes into account the standard deviation, which will increase as δ increases. Consequently, the further away the signal in question is from causing a change in the state of the condition, the greater δ will be which means longer data retaining time in the sender. This magnitude-time relationship is dynamic, and is defined by predictions of the data in queues.

Algorithms 2 and 3 are written in terms of the military simile that represents an application of the proposal. However, the algorithm is applicable to any multivariate condition in a network whose variables are distributed. To do this, some parts of the algorithm that can be adapted are described below.

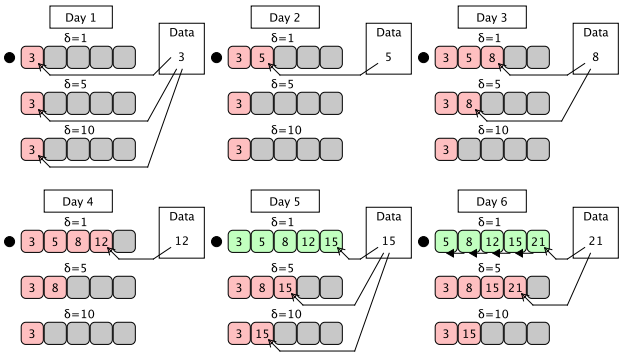

FIGURE 2. Filling process of the queues. The diagram represents an example of the process of filling 3 queues of 5 elements, corresponding to $\delta = 1$, $\delta = 5$ and $\delta = 10$. From left to right and from top to bottom the incorporation of successive data is illustrated. Green elements belong to a complete queue, red elements belong to an incomplete queue, grey elements symbolize absence of value. The black circle indicates the current δ value.

The sending rule The proposed algorithm uses a well-known and simple sending rule: Send-on-Delta [26] (Alg. 2 L: 5, and Alg. 3 L: 12). This is a magnitude-based sampling technique like others that can be considered for performance improvement.

The forecasting technique The forecast is used to predict the maximum expected increment of time signals, based on the data stored in the queues. In this proposal, the standard deviation multiplied by a factor K is used (Alg. 3, L: 15). This part of the algorithm can be modified to use more sophisticated prediction techniques.

The best candidate selection heuristic In each duty cycle, reporting senders are suitable to be reconfigured with another δ value. The restriction to reconfigure only informants is due to the convenience of allowing sensors to deactivate the reception of network packets. In this way, only the sensors that send information will remain listening for a short period as they are susceptible to response. The way the new configuration is decided involves a process that reduces the δ value of the best candidate in each iteration. In the proposed example all signals participate equally in the satisfaction of the condition, so the candidate with the highest expected increase is wanted (Alg. 3, L: 50-51). This part of the algorithm should be adapted to the condition to be evaluated.

The condition The conditional expression is embedded in the castle algorithm(Alg. 3, L: 25, 49). In addition, the execution of the algorithm ends in the satisfaction of the condition, as an adaptation of the proposed military simile. To apply the algorithm in another context where the sink node reports the state of the condition, this code should check if the condition changes state, not if it is true.

TABLE 2. Parameter values M, A, and T for each AS[e] signal.

		Parameters		
		M_e	A_e	T_e
AS[e]	e=1	4	200	400
	e=2	5.5	350	200
	e=3	7	500	100

Finally, regarding the update of defensive devices, algorithm 1 represents a process of relocation of resources ballasted by parameter ε, which establishes a limit of units per day. By replacing defensive devices with units in which the evolution of a reactive procedure can be measured, this algorithm introduces the up-to-date degree of information required.

IV. CASE STUDY

This section illustrates how the algorithm works. For this, a parametric configuration of the model has been chosen as an example, the influence of parameter variation is addressed in Section V. The model described in the previous section is incomplete without enemy recruitment data. These are the AS[e]$\forall e \in \{1, 2, \ldots E\}$ signals. In this example, data is generated with the function defined in (3), which corresponds to the sum of a line without offset and slope M, and a non-shifted sinusoid of amplitude A and period T. Each AS[e] time series has been generated with the combination of M, A, and T shown in Table 2. Fig. 3 represents the evolution of the three signals, the sum of all, the threshold of the condition from day 0 to the day of the battle, and the value of each AS[e] signal on the day of the attack (X). Both (3) and the parameter values of Table 2 have been chosen as an example, in order to add variability to the case study data.

$$f(x) = M \cdot x + A \cdot Sin(\frac{2 \cdot \pi}{T} \cdot x) \tag{3}$$

The strategy of the council (DRACE algorithm) is applied to this scenario with the parameters summarized below (values according to subsections III-B and III-C).

TDS 3000.
TDD 150.
λ 50.
ε 1.
Δ {1, 25, 50, 100, 200, 300}.
DPM 30.
Q 10.
K 3.

The result is illustrated in Fig. 4 and 5. Fig. 4 shows the information obtained in the castle during the strategy operation. The colour grading of the AS′[e] signals reveals the degree of retaining that is set for each of them. The higher the δ, the greater the data retaining and the longer the disinformation period in the castle. The presence of longer constant sections in the parts of higher δ values reveals this fact.

Both edges of the lines are characterized by lower δ values, due to different reasons. In the first few days, $\delta = 1$ due to the queuing process. In the days prior to the bout, δ is minimum due to the forecast of condition compliance. The intermediate period is used to avoid traffic, and it can be noticed here how signal AS′[3] is subject to greater retaining. This is because signal AS′[3] has greater variability and, therefore, the queues corresponding to higher δ values are filled, which does not happen in AS′[1] and AS′[2]. Fig. 6 confirms this fact, by displaying the number of messages of each signal AS′[e] in relation to the corresponding δ value.

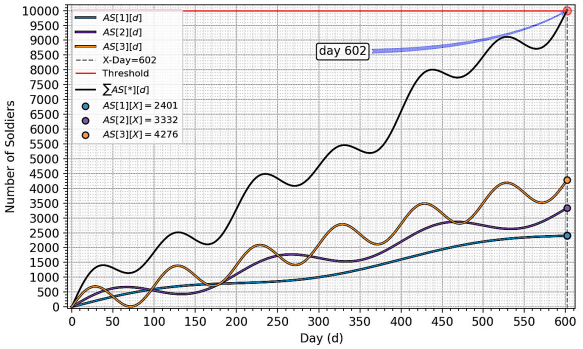

FIGURE 3. Evolution of the number of soldiers in enemy camps ($AS[e][d] = M_e \cdot d + A_e \cdot Sin(\frac{2 \cdot \pi}{T_e} \cdot d)$). The total is represented in black, and the threshold of the attack condition, in red. For this configuration, the battle takes place on day 602, in which the total reaches 10009 soldiers.

Another important aspect of Fig. 4 is the number of sent pigeons. Using the strategy of the council, the number of pigeons is reduced from 1768 (necessary to carry out the original idea of the general) to 352, only 20%, 307 of which correspond to pigeons from explorers (87%), and 45 to pigeons from the castle (13%) (reconfiguring δ values).

The evolution of the battle preparations is illustrated in Fig. 5. It can be observed how, in spite of the strict restriction that parameter ε supposes in the distribution of the defensive devices (Algorithm 1), the received information is enough for the correct preparation of the gates. The battle day is also correctly notified and, therefore, soldiers can be placed for defense. As a result, the victory is for the warlord, who wins at all gates with a narrow margin of strength, as described in (1). Specifically the victory margins are 68 for gate 1, 116 for gate 2, and 155 for gate 3.

V. EXPERIMENTS

This section presents a battery of experiments to analyze the performance of the DRACE algorithm. Starting from an initial configuration, each experiment addresses a variation of the parameters to highlight their influence on the performance. Each experiment consists of a pool of executions of the algorithm using the same set of randomly generated pseudo-realistic signals. Two metrics are extracted from each execution, revealing traffic savings and average accuracy and recall in condition evaluation. Finally, each experiment is shown in a scatter plot that represents both metrics, where each point corresponds to an execution. The following subsections deal with the initial configuration and its variations,

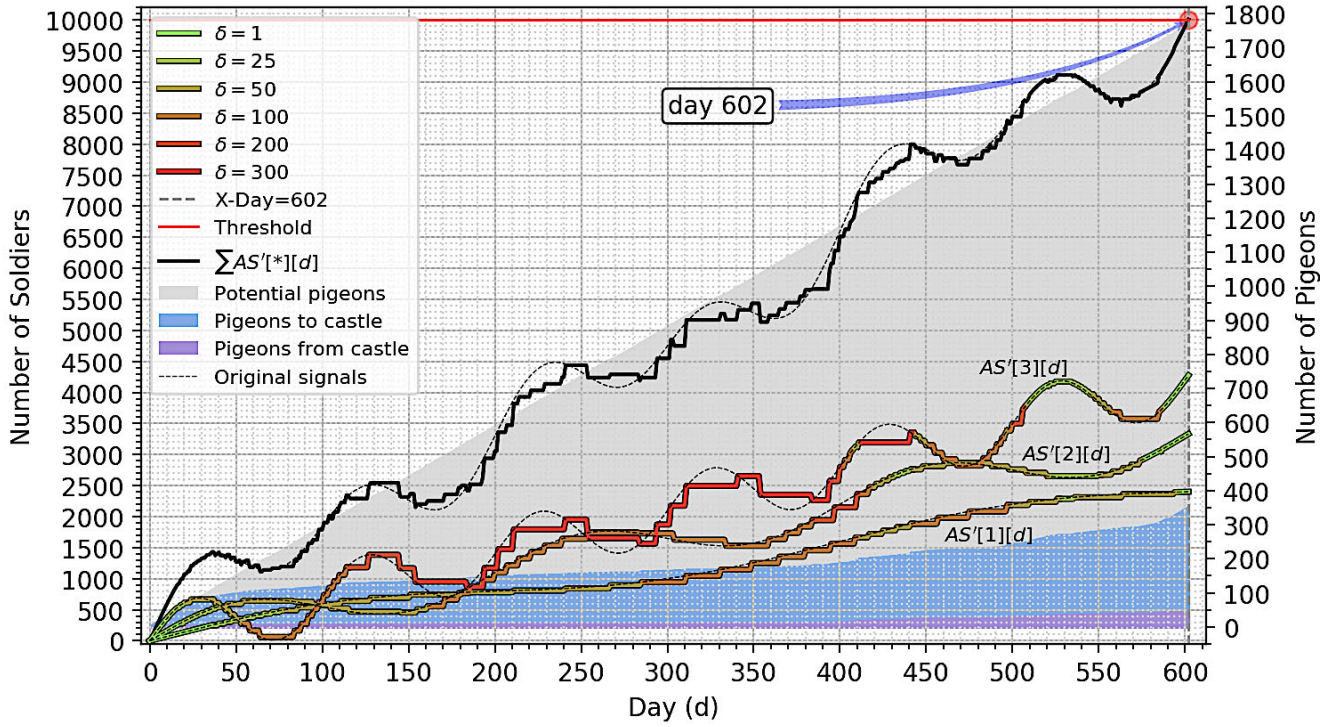

FIGURE 4. Evolution of the AS′[e] signals, which represent the most up-to-date information known at the castle on a daily basis. These signals are coloured on a scale from green to red, representing the δ value of the corresponding explorer at the time of sending. The total is represented in black and the threshold of the condition in red. As a reference, the original signals are shown in dotted lines. In gray, the cumulative number of pigeons that would have been sent if the DRACE algorithm were not applied. The blue area represents the cumulative number of pigeons sent from explorers, and the purple area the cumulative number of pigeons sent from the castle, both add up to the number of pigeons sent using the DRACE algorithm.

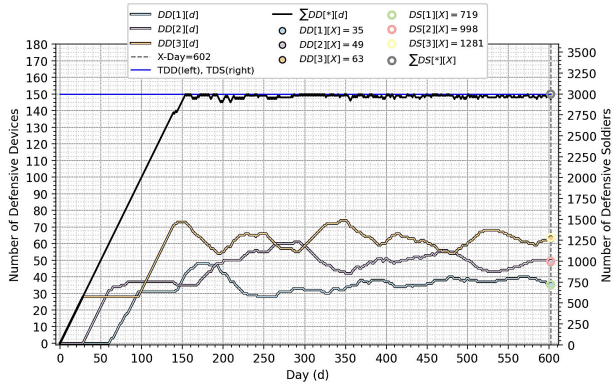

FIGURE 5. Evolution of the defense preparations carried out in the castle. DD[e] signals correspond to the number of defensive devices present in each gate, and evolve according to Algorithm 1. The marks represent the final number of soldiers (DS[e][X]) and devices DD[e][X] defending each door. It is important to note that soldiers are not distributed at the gates until the day of battle. The blue line represents the total number of soldiers (TDS) and devices (TDD) available for defence (on both axes).

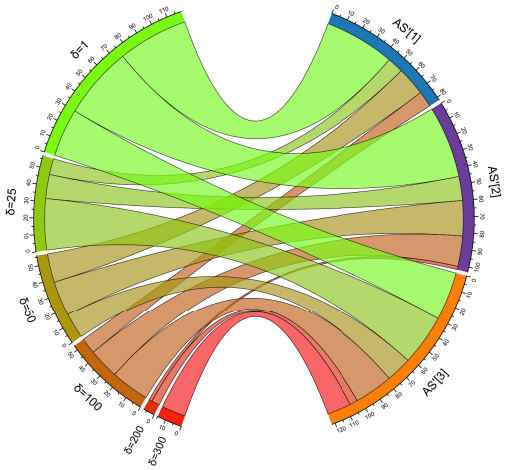

FIGURE 6. Number of pigeons received from each enemy camp (AS′[e] signals) differentiated according to δ value of the corresponding explorer at the time of sending. The diagram has been generated with the R library ***Circlize*** **[11].**

the signal generation, and metrics. A final subsection presents and discusses the results.

A. DRACE CONFIGURATION PARAMETERS

The DRACE application scenario requires a multivariate condition and a series of configuration parameters described in Section III. Section IV deals with a case study as a demonstrator, in which it is required to accurately detect when a condition is satisfied, only once. However, a realistic DRACE application scenario consists of the continuous evaluation of the status of a condition. For this, Algorithm 3 never returns, and the condition satisfaction test (Alg3, L:25, 49) goes on to

check whether the condition state has changed, as discussed in Subsection III-E. This is the initial configuration:

Condition Expression $C[t] = \sum_{i=1}^{3} (S[i][t]) \geq 0.3t$ the state of the condition at instant t. It depends on the value of 3 sensor signals, given by matrix $S \in M_{3\times1000}(\mathbb{R})$, which is chosen from a group of pseudo-realistic random signals of length 1000 limited between 0 and 100. This procedure is addressed in the following subsection. In this case, the threshold is the increasing diagonal that covers from minimum to maximum on both axes.

Forecasting and candidate selection Both are maintained as in the case study (Section IV). Concerning the multiplier factor of the standard deviation, the experiment is performed with $K = 1$, $K = 2$, and $K = 3$.

Δ-Scale $\Delta = \{0.01, 1, 2.5, 5, 10\}$. The values of the scale have been selected by intuition, pretending a valid configuration that remains constant in all experiments.

Data per message (DPM) This parameter defines the maximum size of the retaining buffer in the sensors. $DPM = 30$ in all experiments.

Size of forecasting queues (Q) The experiment is performed with $Q = 5$, $Q = 10$, and $Q = 20$.

With the rest of the parameters remaining constant, the values of Q and K give rise to 9 parameter combinations. Each combination is an experiment, whose results are discussed later.

B. RANDOMLY GENERATED PSEUDO-REALISTIC SIGNALS

SysGpr is a tool for generating pseudo-realistic random signals [20]. The underlying algorithm uses statistical distributions to generate successive increments of the resulting signal. For each sample, the parameters of these statistical distributions are randomly altered using the same mechanism with another configuration. This concept gives rise to a recursive algorithm with a set of nesting levels. As a result it generates continuous but not necessarily monotonous signals. With proper normalization, these signals are indistinguishable from measurements of many actual phenomena, as the paper concludes by 3 validation methods.

In this paper, each combination of DRACE parameters shown in Subsection V-A is an experiment. Each experiment involves 1000 executions, and each execution requires a different $S \in M_{3\times10^3}(\mathbb{R})$ matrix. Considering that the same dataset is used in all experiments, let us define this dataset as $D \in \left(M_{3\times10^3}(\mathbb{R})\right)^{10^3}$, and it is obtained as follows:

1) Let be $R \in M_{10^4\times10^3}(\mathbb{R})$ a repository of 10^4 signals generated with SysGpr. Each signal has 10^3 samples between 0 and 100 each. Fig. 7 illustrates the configuration used. These parameters have been selected by trial and error, to obtain a set of signals whose increments are reasonably limited in relation to the range 0-100, without detriment to diversity. Fig. 8 represents some of the signals obtained as a result of this procedure.
2) Iteratively, while $\#D < 10^3$:
   a) 3 signals are randomly selected from R, resulting in an S matrix.
   b) $C = \left\{\sum S[*][x] \geq 0.3x, \forall x \in \{1, 2, \ldots 10^3\}\right\}$ is calculated.
   c) Considering true $= 1$ and *false* $= 0$, if $450 \leq \sum C \leq 550$ then $D \leftarrow S$, and the signals forming S are removed from R. This condition ensures a balanced result in terms of *true* and *false* samples.

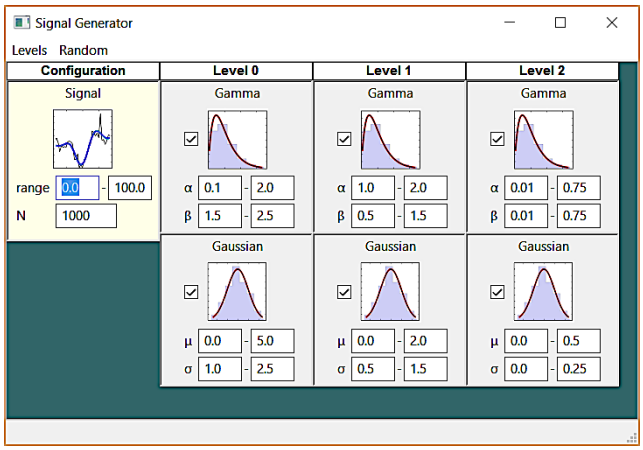

FIGURE 7. SysGpr configuration for the generation of a repository of 10000 pseudo-realistic signals of 1000 samples in the range 0-100. Each level delimits the configuration parameters of the statistical distributions available to generate increases in the previous level.

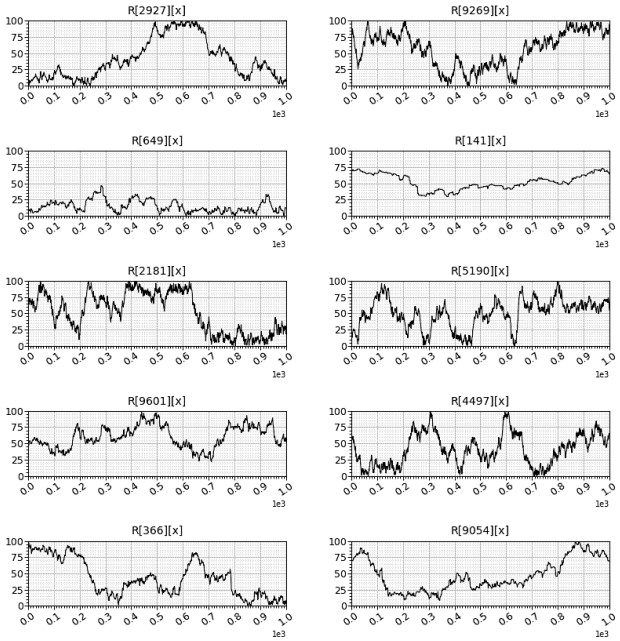

FIGURE 8. Set of signals randomly selected from the R repository.

C. METRICS

As a result of each execution there are two Boolean signals. The first, $C = \left\{\sum S[*][x] \geq 0.3x, \forall x \in \{1, 2, \ldots 10^3\}\right\}$,

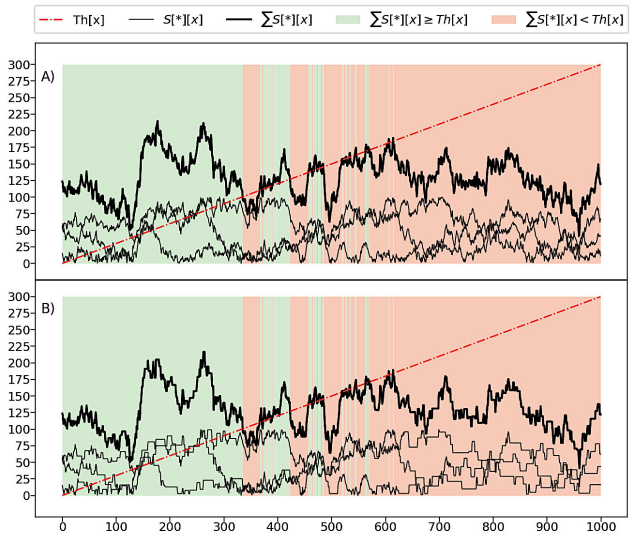

FIGURE 9. Example of execution with $K = 2$ and $Q = 10$. The signals of S are represented by a thin stroke, and the sum, by a thick stroke. The threshold is represented by a red dotted line. B) and A) show the evaluation of the condition with and without DRACE, respectively.

represents the actual state of the condition, evaluated with the original signals. The second, $C' = \left\{\sum S'[*][x] \geq 0.3x, \forall x \in \{1, 2, \ldots 10^3\}\right\}$, is the state of the condition evaluated with the information obtained by applying the DRACE algorithm, exposing the sensors to data retaining. Fig. 9 shows an example of execution, the upper part (A) shows the evaluation of the condition with the original signals, and the lower part (B) with the signals under retaining (DRACE). C and C' are represented with background colors, green for *true* and red for *false*. As expected, the result of the condition on both charts is identical. However, B) represents signals with constant segments. These segments correspond to periods without data updates, as a consequence of the traffic reduction strategy of the DRACE algorithm.

As a consequence of retaining data with DRACE, the number of packets required by S' data is lower than that required by S data. However, DRACE is not an error-free technique. An out-of-forecast value can cause an undetected condition change, which introduces discrepant samples in C and C'. This effect is avoided as long as the forecast is guaranteed, but this is detrimental to traffic savings, because wider margins are required in calculations, which incur in more cautious δ configurations. It is reasonable to assume that the two effects are opposite and that a compromise solution is required. This experimentation is intended to clarify this point empirically.

Paper [21] delves into the effect of data resolution on the result of conditional expressions. The methodology proposes metrics to discrepancies between the obtained and actual Boolean signals in terms of precision and recall. For this purpose, instead of evaluating discrepancies on a sample-by-sample basis, transitions from *true* to *false* and vice versa are considered. In this calculation the variable τ is introduced, which models a tolerable delay margin between the actual transition and the detected transition. In this experiment these metrics are applied without delay tolerance ($\tau = 0$), which simplifies the calculation.

Equations (4), (5), (6), and (7) describe the calculation of transition masks. These masks are binary data arrays with 1s only in those samples where there is a transition from *false* to *true* for C_t and C'_t, and from *true* to *false* for C_f and C'_f. Note that symbols $\oplus$ and $\odot$ correspond to the logic operations XOR and XNOR, respectively.

$$C_t = \left\{\begin{cases} C[i] & \text{if } i = 1 \\ (C[i] \oplus C[i-1]) \cdot C[i] & \text{if } i > 1 \end{cases}, \quad \forall i \in \{1, 2, ..10^3\}\right\} \tag{4}$$

$$C_f = \left\{\begin{cases} |C[i] - 1| & \text{if } i = 1 \\ (C[i] \oplus C[i-1]) \cdot C[i] & \text{if } i > 1 \end{cases}, \quad \forall i \in \{1, 2, ..10^3\}\right\} \tag{5}$$

$$C'_t = \left\{\begin{cases} C'[i] & \text{if } i = 1 \\ (C'[i] \oplus C'[i-1]) \cdot C'[i] & \text{if } i > 1 \end{cases}, \quad \forall i \in \{1, 2, ..10^3\}\right\} \tag{6}$$

$$C'_f = \left\{\begin{cases} |C'[i] - 1| & \text{if } i = 1 \\ (C'[i] \oplus C'[i-1]) \cdot C'[i] & \text{if } i > 1 \end{cases}, \quad \forall i \in \{1, 2, ..10^3\}\right\} \tag{7}$$

Transition precision is defined as the proportion of detected transitions that are real, and transition recall as the proportion of actual transitions that are detected. Equations (8) and (9) describe their calculation from the transition masks, respectively.

$$\text{TP} = \frac{\sum_{i=1}^{10^3} \left(C'_t[i] \odot C_t[i]\right) + \sum_{i=1}^{10^3} \left(C'_f[i] \odot C_f[i]\right)}{\sum_{i=1}^{10^3} \left(C'_t[i] + C'_f[i]\right)} \tag{8}$$

$$\text{TR} = \frac{\sum_{i=1}^{10^3} \left(C'_t[i] \odot C_t[i]\right) + \sum_{i=1}^{10^3} \left(C'_f[i] \odot C_f[i]\right)}{\sum_{i=1}^{10^3} \left(C_t[i] + C_f[i]\right)} \tag{9}$$

As a unified measure of the accuracy of the result, the metric used is the harmonic mean of precision and recall, usually denoted by F1-score, the calculation of which is defined in (10).

$$\text{F1score} = 2 \cdot \frac{\text{TP} \cdot \text{TR}}{\text{TP} + \text{TR}} \tag{10}$$

With regard to traffic savings, the Traffic Saving Ratio (TSR) is defined as the proportion of transmitted packets in relation to those that would have been sent without DRACE. For experimentation, counters are introduced in Algorithm 3 to obtain the number of data packets (from the

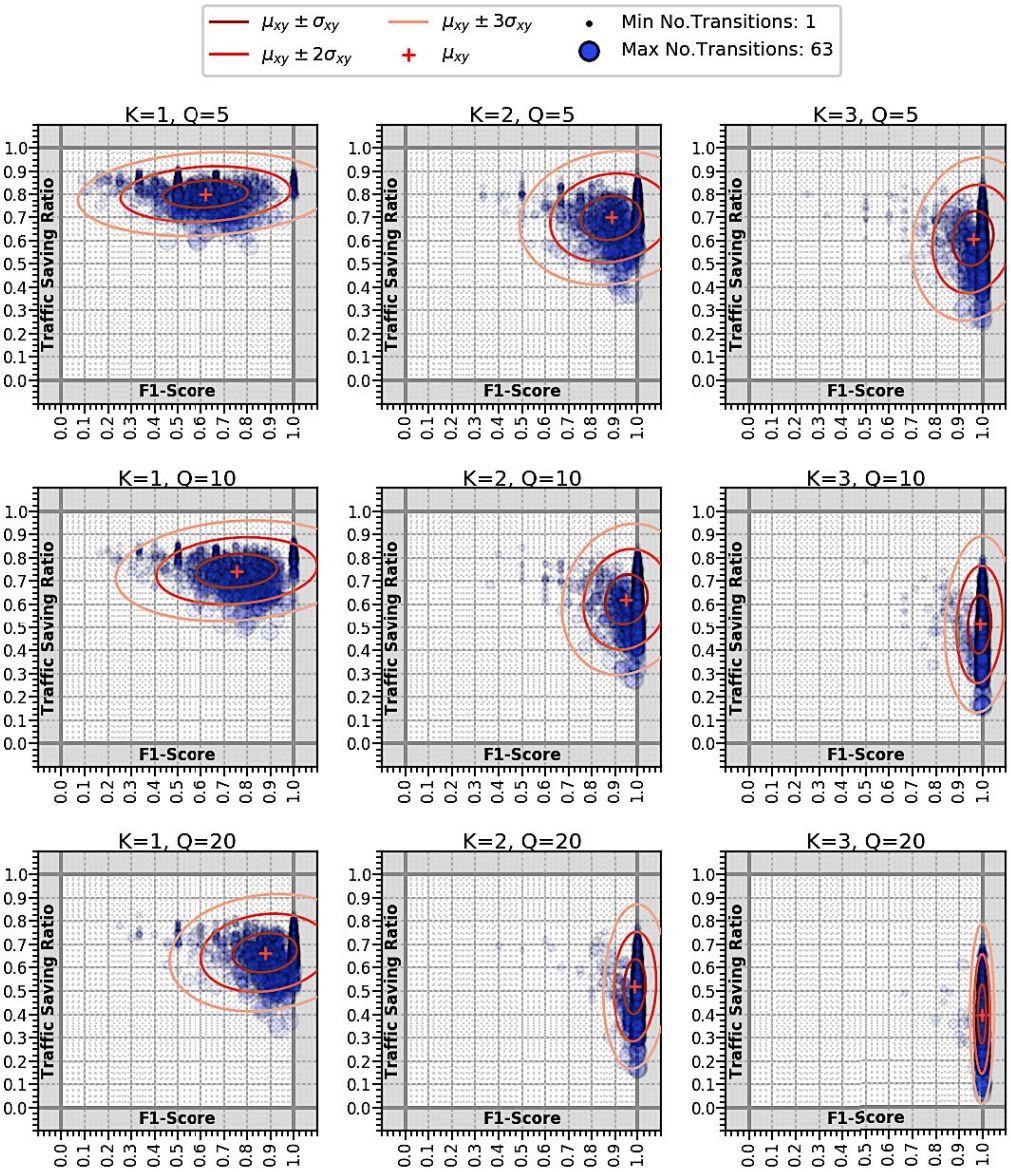

FIGURE 10. Matrix of scatter diagrams with the results of the experimentation. The results of the experiments for increasing Q values are shown from top to bottom. From left to right for increasing K values. In each diagram, 10^3 executions of DRACE are represented by semi-transparent blue marks, proportionally sized to the number of transitions detected in that execution. In red tones, the mean population (μ_{xy}) and iso-contours of the Gaussian distribution for 1σ, 2σ, and 3σ are represented, covering the 68, 95 and 99 percent of the points, respectively.

sensors, line 7) and configuration packets (from the sink node, line 61) transmitted. NDP and NCP are the variables reflecting these scores, respectively. The number of packets required to transmit the original signals (S) is calculated assuming that all samples that differ from the previous one are sent. Equation (11) defines the TSR calculation, assuming the logic expression returns 1 if *true* and 0 if *false*.

$$\text{TSR} = 1 - \frac{\text{NDP} + \text{NCP}}{\sum_{i=1}^{3}\left(1 + \sum_{j=2}^{10^3}(S[i][j] \neq S[i][j-1])\right)} \quad (11)$$

F1-Score and TSR are the metrics that model the incidence of the two opposite effects described on a 0-1 scale for each execution. F1score = 0 corresponds to an execution in which all detected transitions are erroneous, and F1-score = 1 the opposite. TSR = 0 corresponds to an execution in which traffic is not avoided, and TSR = 1 in which all the traffic is avoided (unreal case). TSR can acquire negative values, which are interpreted as sending more traffic than necessary without DRACE.

D. RESULTS AND DISCUSSION

As described above, DRACE has two effects, traffic savings and inaccuracy in condition detection. To analyse the performance of the algorithm, Subsection V-C describes two metrics to measure the incidence of both in each execution.

Concerning the design of the experiment, there are two important aspects: the DRACE parameter set and the test data. Subsection V-A presents the set of experimental parameters. $\Delta = \{0.01, 1, 2.5, 5, 10\}$ and DPM $= 30$ remain constant, while K and Q are vary, resulting in 9 experimental parameterizations ($K \in \{1, 2, 3\} \times Q \in \{5, 10, 20\}$). This decision is taken because it is reasonable to assume that their configuration has a greater impact on the effects explained, as they are related to the precision and margins of the forecast. Regarding the test data, large amounts of pseudo-realistic signals have been generated to represent as many different scenarios as possible, as described in Subsection V-B.

According to this methodology, 9 experiments are performed, one per each parametric configuration. Each experiment consists of 1000 executions in which DRACE is applied to the triads of random signals of $D \in \left(\mathrm{M}_{3\times 10^3}(\mathbb{R})\right)^{10^3}$, obtaining TSR and F1-score in each case.

The results are shown as a scatter plot matrix in Fig. 10. Each graph contains 10^3 marks, one for each triad of signals tested. The size of the marks corresponds to the number of transitions obtained. The experiments present an average of 15 transitions, with 1 being the minimum and 63 the maximum. Each mark is semi-transparent to allow intuition of data frequency by overlap. Also, population mean (μ_{xy}) and confidence ellipses ($\mu_{xy} \pm \alpha \cdot \sigma_{xy}, \forall \alpha \in \{1, 2, 3\}$) are represented in each scatter plot. These ellipses represent the iso-contour of the Gaussian distribution, containing 68%, 95%, and 99% of the samples, with $\alpha = 1$, $\alpha = 2$, and $\alpha = 3$ respectively [33].

Suppose G[i][j] is the graph of row i and column j. The diagonal formed by G[3][3], G[2][2], and G[1][1] represents the transition from the most reliable configuration for forecasting to the least reliable one. As expected, the more reliable the prediction, the more accurate the condition evaluation and the lower the traffic saving ratio.

It can be observed how in extreme cases (G[1][1] and G[3][3]), the dispersion is reduced in the favourable metric and increased in the unfavourable one. This question is related to the signals behaviour and how they interact with the threshold. DRACE takes advantage of opportunity periods when the condition is far from a change of state. The more periods of opportunity with more distance to the change of state of the condition, the more cautious can be the configuration of the forecast without affecting the reduction of traffic. In diagrams G[1][1] and G[3][3], this casuistry is dispersed with respect to the metric that the experiment is not conditioning.

K is a multiplication factor of the maximum deviation forecasted, while Q sets the sample size on which the forecasting is performed. Therefore, both variables influence the probability of success in prediction. K by introducing a post-forecast margin, and Q taking into account a wider range of data and, consequently, considering a longer signal history. The effects of both parameters are similar, as evidenced by the similarity of the G[3][2] and G[2][3], G[3][1] and G[1][3], and G[2][1] and G[1][2] diagrams.

There is an inversely proportional relationship between the number of transitions and traffic saving ratio, this can be appreciated particularly clearly in the diagrams in column 3. This is due to the fact that as transitions increase, so do the changes in condition statuses, which statistically reduces the opportunity periods for retaining data.

Finally, considering the wide variety of cases represented, some dispersion in both axes was expected in all cases. However, the most accurate configurations present a strong sample convergence at F1-score $= 1$, avoiding, in 68% of the cases, between 30 and 50% of the traffic. Although this experiment does not demonstrate the relevance of using DRACE in all scenarios, it seems reasonable to conclude that it is convenient to consider its application, since for most of the test scenarios using random signals, it can avoid a large considerable portion of the traffic without committing errors.

VI. CONCLUSION

This paper deals with the evaluation in time of multivariate conditions when variables are distributed. When the multivariate condition is the expression of a threshold-based event, early detection requires continuous monitoring of all involved variables. Algorithm DRACE is proposed from the perspective of a centralized duty cycle solution. This algorithm regulates a data retaining strategy in the data sources, attempting to avoid network packets without affecting the result of the event detection.

DRACE is based on data forecasting to take advantage of periods in which a change in condition status has a remote probability. During these periods and depending on this calculation, it reconfigures the data sources in order to retain the information proportionally. Data retaining is a consequence of the well-known Send-on-Delta mechanism. It is therefore a magnitude-based technique, not a time-based one. This aspect provides robustness, since possible out-of-forecast data are sent even in a regime of strong data retaining, as long as its magnitude is a trigger condition for sending.

In an effort to illustrate the motivation and promote understanding of its operation, DRACE is presented on a warlike simile that deals with all intrinsic aspects of the problem in another context. This contextualized statement is at the disposal of the scientific community, and we hope that it will be motivating for other authors to propose alternative solutions.

To analyze the performance of DRACE, 9000 executions of the algorithm have been performed. 9 configurations tested on a repository of 3 million data organized in 1000 triads of signals of 1000 samples from 0 to 100 each. The data has been generated with SysGpr, a validated technique that synthesizes continuous pseudo-realistic signals suitable for testing purposes. The traffic savings and the accuracy of the event detection are measured in each execution. The metric used to measure accuracy is based on transitions in the resulting Boolean signal, allowing an analysis based on the harmonic mean of precision and recall.

The experiments reveal the need for a compromise solution between accuracy and traffic savings in the parameter configuration. Nevertheless, the obtained results are promising. Focusing on the most accurate configuration, 99% of executions are error-free, and the number of packets is reduced by 40% on average, being between 30 and 50% in 68% of cases. If some degree of inaccuracy is tolerable, DRACE can reduce traffic by more than 50% with an event detection accuracy greater than 90% in most cases.

VII. FUTURE WORK

This work is part of a research line related to dynamic management of network resources for the efficient deployment of distributed intelligence models. Reference [21] is a previous work which addresses the reduction of the information domain to optimize network traffic, avoiding congestion. DRACE algorithm is an important contribution in this topic, providing an adaptive mechanism based on data retaining. However, there are many aspects not yet addressed for which it will provide an adequate grounding. Some of them are discussed here.

- The proposed procedure causes that all the data is sent (Alg. 2, L: 4). Possible errors in the condition evaluation are due to the fact that the information may not be available on time. This feature allows the sink to produce statistics such as those described in the experimentation. Based on these statistics, it is possible to develop machine learning models in order to seek the optimal DRACE configuration at runtime for a preset level of accuracy or traffic saving ratio.
- DRACE does not reduce the bit size of data. The DPM (Data Per Message) parameter configures the number of data per network packet, but these are sent unaltered. For applications in which the payload must be used to the maximum, it is possible to consider the reduction of bits per data [21], in addition to their retaining. This feature could be especially useful in multimedia sensor network applications where it is required to detect events in a massive data stream, such as video surveillance.
- Parameter Δ corresponds to a scale of magnitudes (δ) that affect the retaining method by means of the Send-on-Delta concept [26]. This scale should be established according to the behaviour and margins of the signals to be monitored. The in-depth analysis of the optimal configuration of this parameter is beyond the scope of this paper, but it would be interesting to address it in the future. In the same way, the restriction of a single Δ for all inputs should be reconsidered, especially in systems that require heterogeneous information.
- As discussed in Subsection III-E, the forecasting technique and the next candidate selection heuristic for seeking the best δ-configuration are subject to reconsideration for improving algorithm performance.

REFERENCES

[1] C. Antonopoulos, S.-M. Dima, and S. Koubias, "Event identification in wireless sensor networks," in *Components and Services for IoT Platforms*. Cham, Switzerland: Springer, 2017, ch. 10, pp. 187–210.
[2] K. J. Astrom and B. M. Bernhardsson, "Comparison of Riemann and Lebesgue sampling for first order stochastic systems," in *Proc. 41st IEEE Conf. Decis. Control*, Dec. 2002, pp. 2011–2016.
[3] H. F. Atlam, R. J. Walters, and G. B. Wills, "Fog computing and the Internet of Things: A review," *Big Data Cogn. Comput.*, vol. 2, no. 2, p. 10, 2018.
[4] C. Bădică, L. Braubach, and A. Paschke, "Rule-based distributed and agent systems," in *Proc. Int. Workshop Rules Rule Markup Lang. Semantic Web* in Lecture Notes in Computer Science, vol. 6826, 2011, pp. 3–28.
[5] M. Bahrepour, N. Meratnia, and P. Havinga, "Automatic fire detection: A survey from wireless sensor network perspective," CTIT, Univ. Twente, Twente, The Netherlands, Tech. Rep. WoTUG-31/TR-CTIT-08-73, 2007.
[6] R. Bi, H. Gao, and Y. Li, "Probabilistic threshold based monitoring using sensor networks," in *Proc. Int. Conf. Wireless Algorithms, Syst., Appl.*, in Lecture Notes in Computer Science, vol. 8491, 2014, pp. 246–255.
[7] G. Campobello, A. Segreto, and S. Serrano, "Data gathering techniques for wireless sensor networks: A comparison," *Int. J. Distrib. Sensor Netw.*, vol. 2016, pp. 1–17, Mar. 2016.
[8] G. M. Dias, B. Bellalta, and S. Oechsner, "A survey about prediction-based data reduction in wireless sensor networks," *ACM Comput. Surv.*, vol. 49, no. 3, p. 58, 2016.
[9] N. Dziengel, "Distributively observed events in wireless sensor networks," Ph.D. dissertation, Freie Universität Berlin, Berlin, Germany, 2015.
[10] D. Goldsmith and J. Brusey, "The spanish inquisition protocol—Model based transmission reduction for wireless sensor networks," in *Proc. IEEE SENSORS*, Nov. 2010, pp. 2043–2048.
[11] Z. Gu, L. Gu, R. Eils, M. Schlesner, and B. Brors, "Circlize implements and enhances circular visualization in R," *Bioinformatics*, vol. 30, no. 19, pp. 2811–2812, 2014.
[12] S. Hammoudi, Z. Aliouat, and S. Harous, "Challenges and research directions for Internet of Things," *Telecommun. Syst.*, vol. 67, no. 2, pp. 367–385, 2018.
[13] M. Hasnain, M. H. Malik, and M. E. Aydin, "An adaptive opportunistic routing scheme for reliable data delivery in WSNs," in *Proc. 2nd Int. Conf. Future Netw. Distrib. Syst.*, 2018, Art. no. 32.
[14] F. Hu and Q. Hao, Eds., *Intelligent Sensor Networks: The Integration of Sensor Networks, Signal Processing and Machine Learning*, 1st ed. Boca Raton, FL, USA: CRC Press, 2013.
[15] N. Ilyas, M. Akbar, R. Ullah, M. Khalid, A. Arif, A. Hafeez, U. Qasim, Z. A. Khan, and N. Javaid, "SEDG: Scalable and efficient data gathering routing protocol for underwater WSNs," *Procedia Comput. Sci.*, vol. 52, no. 1, pp. 584–591, 2015.
[16] K. Kapitanova, S. H. Son, and K. D. Kang, "Using fuzzy logic for robust event detection in wireless sensor networks," *Ad Hoc Netw.*, vol. 10, no. 4, pp. 709–722, 2012.
[17] M. C. Kerman, W. Jiang, A. F. Blumberg, and S. E. Buttrey, "Event detection challenges, methods, and applications in natural and artificial systems," in *Proc. 14th Int. Command Control Res. Technol. Symp.*, Mar. 2009, pp. 1–19.
[18] Y. Kim, D. Kim, P. K. Chong, J. Kang, E. Kim, and S. Seo, "Design of a fence surveillance system based on wireless sensor networks," in *Proc. 2nd Int. Conf. Autonomic Comput. Commun. Syst.*, 2008, Art. no. 4.
[19] J. W. Ko and Y. H. Choi, "A grid-based distributed event detection scheme for wireless sensor networks," *Sensors*, vol. 11, no. 11, pp. 10048–10062, 2011.
[20] F. León, F. J. Rodríguez-Lozano, J. M. Palomares, and J. Olivares, "SysGpr: Sistema de Generación de Señales Sintéticas Pseudo-realistas," *Revista Iberoamericana Automática Informática*, vol. 16, no. 3, pp. 369–379, 2019.
[21] F. Leon-Garcia, J. M. Palomares, and J. Olivares, "D2R-TED: Data—Domain reduction model for threshold-based event detection in sensor networks," *Sensors*, vol. 18, no. 11, p. 3806, 2018.
[22] Y. Li, C. Ai, C. Vu, Y. Pan, and R. Beyah, "Delay-bounded and energy-efficient composite event monitoring in heterogeneous wireless sensor networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 21, no. 9, pp. 1373–1385, Sep. 2010.

[23] D. J. McCorrie, E. Gaura, K. Burnham, N. Poole, and R. Hazelden, "Predictive data reduction in wireless sensor networks using selective filtering for engine monitoring," in *Wireless Sensor and Mobile Ad-Hoc Networks: Vehicular and Space Applications*. New York, NY, USA: Springer-Verlag, 2015.
[24] U. I. Minhas, I. H. Naqvi, S. Qaisar, K. Ali, S. Shahid, and M. A. Aslam, "A WSN for monitoring and event reporting in underground mine environments," *IEEE Syst. J.*, vol. 12, no. 1, pp. 485–496, Mar. 2018.
[25] M. Miskowicz, "The event-triggered sampling optimization criterion for distributed networked monitoring and control systems," in *Proc. IEEE Int. Conf. Ind. Technol.*, Dec. 2003, pp. 1083–1088.
[26] M. Miskowicz, "Send-on-delta concept: An event-based data reporting strategy," *Sensors*, vol. 6, no. 1, pp. 49–63, 2006.
[27] M. Miśkowicz, *Event-Based Control and Signal Processing*. Boca Raton, FL, USA: CRC Press, 2015.
[28] A. Nasridinov, S.-Y. Ihm, Y.-S. Jeong, and Y.-H. Park, "Event detection in wireless sensor networks: Survey and challenges," in *Mobile, Ubiquitous, and Intelligent Computing*, vol. 274. Berlin, Germany: Springer-Verlag, 2014.
[29] T. Park, N. Abuzainab, and W. Saad, "Learning how to communicate in the Internet of Things: Finite resources and heterogeneity," *IEEE Access*, vol. 4, pp. 7063–7073, 2016.
[30] M. A. Razzaque, C. Bleakley, and S. Dobson, "Compression in wireless sensor networks: A survey and comparative evaluation," *ACM Trans. Sensor Netw.*, vol. 10, no. 1, p. 5, 2013.
[31] M. Shahul, S. Thomas, and R. Vijayakumar, "A review on event detection techniques in wireless sensor network," *Int. J. Innov. Res. Sci., Eng. Technol.*, vol. 6, no. 7, pp. 1–5, 2017.
[32] I. U. Din, M. Guizani, S. Hassan, B.-S. Kim, M. K. Khan, M. Atiquzzaman, and S. H. Ahmed, "The Internet of Things: A review of enabled technologies and future challenges," *IEEE Access*, vol. 7, pp. 7606–7640, 2019.
[33] B. Wang, W. Shi, and Z. Miao, "Confidence analysis of standard deviational ellipse and its extension into higher dimensional Euclidean space," *PLoS ONE*, vol. 10, no. 3, 2015, Art. no. e0118537.
[34] G. Werner-Allen, K. Lorincz, M. Welsh, O. Marcillo, J. Johnson, M. Ruiz, and J. Lees, "Deploying a wireless sensor network on an active volcano," *IEEE Internet Comput.*, vol. 10, no. 2, pp. 18–25, Mar./Apr. 2006.
[35] G. Wittenburg, N. Dziengel, S. Adler, Z. Kasmi, M. Ziegert, and J. Schiller, "Cooperative event detection in wireless sensor networks," *IEEE Commun. Mag.*, vol. 50, no. 12, pp. 124–131, Dec. 2012.
[36] X.-M. Zhang, Q.-L. Han, and B.-L. Zhang, "An overview and deep investigation on sampled-data-based event-triggered control and filtering for networked systems," *IEEE Trans. Ind. Informat.*, vol. 13, no. 1, pp. 4–16, Feb. 2017.
[37] S. Zöller, C. Vollmer, M. Wachtel, R. Steinmetz, and A. Reinhardt, "Data filtering for wireless sensor networks using forecasting and value of information," in *Proc. Conf. Local Comput. Netw. (LCN)*, Oct. 2013, pp. 441–449.

FERNANDO LEÓN-GARCÍA received the B.Sc. degree in computer engineering and the M.Sc. degree in distributed renewable energies from the Universidad de Córdoba, Spain, in 2006 and 2012, respectively, where he is currently pursuing the Ph.D. degree in computer science with the Advanced Informatics Research Group (GIIA). He was an Embedded Systems Engineer with the private sector in Sevilla, Spain, for five years. Since 2012, he has been a part of the Research Group of Computer and Automatic Engineering Projects (PRINIA), and later, he has been a Senior Researcher with GIIA. His research interests include the Internet of Things, wireless sensor networks, embedded systems, fog computing, and communication compression techniques.

FRANCISCO J. RODRÍGUEZ-LOZANO was born in El Porvenir De La Industria, Spain. He received the B.Sc. degree in computer engineering and the M.Sc. degree in distributed renewable energies from the Universidad de Córdoba, Spain, in 2016 and 2017, respectively, where he is currently a member of the Advanced Informatics Research Group (GIIA). His research interests include image processing, computer vision, machine learning, high-performance computing, and robotic systems.

JOAQUÍN OLIVARES was born in Elche, Spain. He received the B.S. and M.S. degrees in computer science and the M.S. degree in electronics engineering from the Universidad de Granada, Spain, in 1997, 1999, and 2003, respectively, and the Ph.D. degree from the Universidad de Córdoba, Spain, in 2008, where he has been an Associate Professor with the Electronic and Computer Engineering Department, since 2001. He is also the Founder and the Head of the Advanced Informatics Research Group, Universidad de Córdoba. His research interests include the Internet of Things, wireless communications, embedded systems, computer vision, and high-performance computing.

JOSE M. PALOMARES (M'05–SM'11) was born in Motril, Spain, in 1975. He received the B.Sc., M.Sc., and Ph.D. degrees in computer engineering from the Universidad de Granada, Granada, Spain, in 1996, 1998, and 2011, respectively. Since 2000, he has been a Lecturer and an Assistant Professor with the Universidad de Córdoba, Córdoba, Spain, where he is currently an Associate Professor and is also the Co-Founder of the Advanced Informatics Research Group (GIIA). His research interests include image and video processing, real-time systems, wireless sensor networks, and computer architecture. He has been a member of the ACM, since 2005.