Information Fusion 45 (2019) 38–52

Contents lists available at ScienceDirect

Information Fusion

journal homepage: www.elsevier.com/locate/inffus

Fusing information from tickets and alerts to improve the incident resolution process

Saeed Salah, Gabriel Maciá-Fernández, Jesús E. Díaz-Verdejo*

Department of Signal Theory, Telematics and Communications - CITIC, University of Granada, c/ Periodista Daniel Saucedo Aranda, s/n Granada 18071, Spain

ARTICLE INFO

Keywords:
Quality of service
Data analysis
Network management systems
Alert correlation
Ticket-alert correlation

ABSTRACT

In the context of network incident monitoring, alerts are useful notifications that provide IT management staff with information about incidents. They are usually triggered in an automatic manner by network equipment and monitoring systems, thus containing only technical information available to the systems that are generating them. On the other hand, ticketing systems play a different role in this context. Tickets represent the business point of view of incidents. They are usually generated by human intervention and contain enriched semantic information about ongoing and past incidents. In this article, our main hypothesis is that incorporating tickets information into the alert correlation process will be beneficial to the incident resolution life-cycle in terms of accuracy, timing, and overall incident's description. We propose a methodology to validate this hypothesis and suggest a solution to the main challenges that appear. The proposed correlation approach is based on the time alignment of the events (alerts and tickets) that affect common elements in the network. For this we use real alert and ticket datasets obtained from a large telecommunications network. The results have shown that using ticket information enhances the incident resolution process, mainly by reducing and aggregating a higher percentage of alerts compared with standard alert correlation systems that only use alerts as the main source of information. Finally, we also show the applicability and usability of this model by applying it to a case study where we analyze the performance of the management staff.

1. Introduction

Nowadays, IT Service Management (ITSM) [1] tasks constitute a heavy burden on the management staff of modern telecommunication networks, as these networks become larger and more complex in terms of the diversity and the criticality of the services that they offer to customers. In addition to that, IT management is now a business-oriented service rather than just a process for network/systems management. This means that IT services are adopted according to their contributions to the required business processes. For this reason, network experts are always trying to find efficient strategies to quickly solve network incidents and improve the network uptime to comply with committed Service Level Agreement (SLA).

ITSM adopts the Information Technology Infrastructure Library (ITIL) framework [2], which is a widely accepted industry standard that is defined as the best practice in managing information technology services and providing infrastructure, development and operations for identifying, planning, delivering and supporting IT services for business. Incident Management (IM) is one of the main processes defined in ITIL. Citing the ITIL terminology, an *incident* can be defined as "*an unplanned interruption of an IT service or reduction in the quality of an IT service. Failure of a configuration item that has not yet impacted service is also an incident*" [3].

Typically, *alerts* are the main source of information used by management staff to derive the existence of incidents. Alerts in the network are collected by monitoring systems, which are intended to warn staff in network operation and management centers. The sensitivity of today's monitoring systems leads a huge amount of alerts being triggered per day, which overwhelms management staff. This issue makes it advisable to use and develop additional systems to reduce this quantity of alerts. Alerts correlation [4–7] is the primary technique employed to handle this problem.

Alternately, Incident Ticketing Systems (ITSs), also known as Service Desks as referred to by the ITIL terminology, are a primary tool used by management staff to track and report ongoing and resolved incidents. ITSs store records called *tickets*, which can be created either automatically by an ITS in response to receiving alerts or manually by humans. In the latter case, tickets can be created from two different sources: (1) by the help desk staff in response to receiving customers' calls regarding some problems in the resources, and (2) by the

* Corresponding author.
E-mail addresses: sasalah@staff.alquds.edu (S. Salah), gmacia@ugr.es (G. Maciá-Fernández), jedv@ugr.es (J.E. Díaz-Verdejo).

https://doi.org/10.1016/j.inffus.2018.01.011
Received 14 February 2017; Received in revised form 18 December 2017; Accepted 14 January 2018
Available online 17 January 2018

management staff, from the observed alerts but also from other symptoms and even due to some feedback from other technicians at different locations.

Although many of the records in ITSs contain semantically rich information related to incidents, to the best of our knowledge, only limited efforts have been devoted to the inclusion of this information in the alert correlation procedure [8,9]. As the bulk of the tickets in ITSs are created manually, they are ideal candidates for the addition of further semantic information and human knowledge, both from the management staff and from the users of the services (through help desk staff), into the alert correlation procedure.

It is worth mentioning that in this work we are not concerned with specific algorithms that extract information from natural language in tickets but to show that they contain additional information that can be used to correlate them with alerts. In our experiments we show that the approach works even with a quite simplistic approach. As it will be shown later, both datasets have some specificities that make this correlation hard to be performed. Yet, despite these specificities, both sources of information intersect in several ways, as they contain technical information about ongoing incidents, including human-expert information in tickets. In the case study we present, every ticket is characterized by several tenths of features, and we show that only some of them contain technical information that is useful for the alert-ticket correlation process.

The potential benefits from this are threefold: *(i)* from the IT user perspective, the proposed methodology can enhance the user expectations regarding IT service quality by speeding up the incident resolution process; *(ii)* from the IT management perspective, better event correlation rates would be obtained, *i.e.*, a larger and more reliable reduction in the number of alerts. This last benefit enhances the incident detection accuracy and reduces False Positives (FPs). Finally, (*iii)* from the decision making perspective, managers would receive more accurate information with regard to the real incidents that occur in the network and their descriptions. This potentially improves incident management cycles, as it provides a more accurate feedback on the Quality of Service (QoS) in the incident management process, and shows how the different teams involved in the process behave.

In this article, our main contribution is to show that incorporating the information found in tickets into the alert correlation system significantly improves the correlation of the events, and thus the overall incident resolution process. For this, we propose a methodology to correlate tickets and alerts (events in the system) based on an intentionally simple algorithm, as our purpose is to show that even with a simple correlation criterion, the contribution of tickets to the correlation process is beneficial for both management and decision making processes. The proposed correlation approach is based on the time alignment of the events (alerts and tickets) that affect common elements in the network. Thus, the algorithm groups together all the events related to the same incident in what we call *representative events*, that summarize all the information from the grouped elements. Ideally, at the end of the algorithm execution, a single representative event per incident is provided.

We evaluate the proposal with a real dataset from a company that is in charge of the operation and management of a corporate network that provides services to a regional government. The network serves millions of habitants from many public sectors, such as education, health and civil, among others, and includes a help desk center that generates tickets directly from end users complaints. Furthermore, we show the applicability and usefulness of the proposed solution by applying it in two scenarios: first, in the alert correlation process, especially for reducing and aggregating a larger number of related alerts. Second, in assessing the performance of the management staff in terms of their speed, accuracy and the efficiency achieved by the different management groups in the entire incident resolution process.

The structure of the article is as follows. First, a review of the related work is summarized in Section 2. Then, a basic model for event correlation is presented in Section 3. Based on this model, a complete system for ticket-alert correlation is proposed and discussed in Section 4. A thorough explanations of the proposed ticket-alert correlation model, the arising challenges and the suggested solutions are provided in Section 5. The correlation model is experimentally tested and evaluated through a case study utilizing the dataset taken from a real management company in Section 6. Some applications of the proposed model are listed in Section 7. Finally, in Section 8, we draw various conclusions and provide insights into further works.

2. Related work

Despite the large amount of research effort that has been carried out in the alert correlation field [10–15], this is still an active research area in both NMS and IT security. This is mainly because the efficiency and robustness of the used models and the proposed algorithms vary from system to system, but none of them have thus far succeeded in providing an optimal solution to this problem in terms of reducing the number of alerts to a single alert per incident [16–18].

The information considered during the alert correlation process can derive from many different sources of information [19], *e.g.*, topological information that provides an accurate representation of the monitored network as a set of links and nodes [20,21]. In particular, the representation of the location of nodes and the connectivity and direction of the existing links are of particular relevance. Network topology information usually contains extensive details about the network and equipment structure, such as switches, routers, and servers; configuration parameters, such as IP addresses and their matchings to names, subnets or virtual LANs; and host information, such as OS type and open services.

Many of the alert correlation techniques adopt expert rules and similarity-based correlation methods [22–26], aiming to reduce the total number of alerts by aggregating them using their similarities. The main assumption behind similarity-based techniques is that similar alerts tend to have the same root causes or similar effects on the monitored system. How to define similarity measures is a critical performance issue for such techniques. To answer this question, several similarity measures have been used by many researchers [27–29]. The aim is to define a suitable similarity function for each attribute observed in the alerts because attributes may have different weights and effects on the overall correlation process.

Some authors have applied data mining methods for alert correlation analysis, such as association rules mining [30–33], incremental frequent mining [34], and sequential pattern mining [35,36]. Their aim is to automate the process of finding meaningful activities and interesting features from training datasets and build a knowledgebase that can be used for the alert correlation process in real-time. The main drawback of this approach is the heavy load imposed on the system to build models that dynamically adapt to new conditions.

Alternately, some research efforts, such as those in [37–39], have noted the importance of ticket correlation for incident resolution, claiming that the latter can be extended with advanced functions to enhance the incident resolution process, as the information in the tickets is related to incidents generated by events that have already been identified as network failures, and as such, some related alerts should exist. Other efforts, such as those in [40–42], use ITSs for several purposes, such as studying and characterizing the nature and causes of routing changes and the observed instability. In these references, the authors use simple ticket preprocessing operations to reduce the total number of tickets before correlating them. However, they do not deeply analyze the ITS, and the correlation of the tickets is not targeted at reaching one ticket per incident.

In another research line, in [43] the authors proposed an intrusion detection model that leverages contextual information to create attack prediction models driven by database and graph mining techniques. The proposed approach automatically identifies and queries flows to

generate semantic links among alerts raised in response to suspicious activities. It consists of two main phases: the preprocessing phase, which leverages previous flows to create a flow classification model, called Semantic Link Network (SLN); and the prediction phase, which occurs at run-time and takes incoming flows as inputs, and produces an initial prediction of whether they are suspicious or benign. Despite the main contribution of our work not being targeted at modeling the process associated to an attack, considering context-aware correlation approaches like the one presented in this work has some beneficial outcomes to our approach and might be used as an insight for future improvements or a confirmation on the existence of profitable semantic information.

To the best of our knowledge, despite its potential for obtaining various useful statistical measures to study the nature of incidents and their effects on network stability [44,45], no efforts have been devoted to the use of information from the tickets in the alert correlation process itself (joint correlation), targeted at increasing the percentage of reduction in the number of alerts and the significance of the resulting events.

Finally, despite the availability of a large number of monitoring tools such as Logstash [46], Splunk [47], and Sumo Logic [48], which are mainly designed to assist in storing and analyzing log files from the point of view facilitating the management process, we did not find specific solutions to the challenges described in this article.

In summary, the aim of this work is different than that of the cited references. The proposed method is expected to produce a final set of incidents that more accurately represent the real incidents in the network when compared with standard alert correlation systems that only use the alerts as the main source of information.

3. Basic model for event correlation

Before discussing the ticket-alert correlation process in more detail, we introduce various terminologies in this section and provide a brief overview of the basic event correlation model that we use as a basic building block for the correlation process. As will be shown, this basic model is intentionally simple because its main purpose is just to serve as a basic criterion upon which our ticket-alert correlation proposal is built. As previously discussed, the main goal of this article is showing that even when the underlying correlation algorithm is simple, the contribution of tickets to the alert correlation process is substantial.

In a first step, we consider both the appearance of alerts in the NMS and the generation of tickets in the ITS system as generic *events*. This way, whenever an incident takes place in a monitored network, a set of different events related to that incident appear. Let us denote as I (see Table 1 for a summary of the notation) the set of m different events (e.g., alerts or tickets) that appear as a consequence of an incident occurring in a network: $I = \{e_1, e_2, ..., e_m\}$.

Every event e_i has a different *duration*, spanning from the instant of its creation or appearance, which we call the *event creation time*, $t_{e_i}^{CT}$, to the instant at which this event finishes or is resolved, which we call the *event resolution time*, $t_{e_i}^{RT}$.

In addition, every event e_i is associated with one or more elements of the network, which we call the *affected elements* of that event. For example, in a "node down" alert, the node of the network that has gone down is the affected element of that event. Note that an event could have several affected elements. For example, if a ticket is created due to the failure of several nodes in a network, all of them are really the affected elements for that event. In general, we will say that every event e_i will have a *set of affected elements*, E_i, which is a list of the different *identifiers* of the network elements, applications, services, etc. affected by the incident described in that event. An identifier here could be an IP address, a node name, a service name, etc.

Furthermore, every event e_i is also specified by an *event description*, D_i, which is usually a free text field describing the event, its effects on the network, and/or the root cause of its appearance.

Table 1
List of acronyms and symbols with their descriptions.

Acronym/symbol	Description
FOD	Forward offset delay
BOD	Backward offset delay
SoI	Start of incident
EoI	End of incident
A_R	Representative alert
T_R	Representative ticket
A_P	Set of alerts after preprocessing
T_P	Set of tickets after preprocessing
A_C	Set of alerts after correlation
T_C	Set of tickets after correlation
D_i	Event *ith description*
E_i	Set of affected elements for event *ith*
I	Set of incidents
$t_{e_i}^{CT}$	Event *ith creation time*
$t_{e_i}^{RT}$	Event *ith resolution time*
e_R	Representative event
I_{actual}	Number of real incidents
I_A	Number of incidents extracted from alert dataset
I_T	Number of incidents extracted from ticket dataset
t_{CT}	Ticket creation time
t_{RT}	Ticket resolution time

Normally, when an incident occurs in a network, many different alerts and tickets (events in general) are generated. Here, it is desirable that an ideal event correlation algorithm provides a single event for this incident that contains all of the semantic information extracted from the set of related alerts and tickets. Thus, we are first interested in the correlation of all the events that belong to an incident I so that a single event can represent the entire incident. We refer to this single event as the *representative event* for incident I, e_R. To be coherent with the description of the set I, the duration of the event e_R should span from the earliest event creation time from the events in I to the last event resolution time observed in the set of events for that incident. Fig. 1 shows an example of this definition, where a set of m events belonging to the incident I are represented by a single representative event e_R.

Note that in realistic scenarios, the duration of the representative event might not match the real incident duration, mainly due to the fact that when the incident starts, a delay could occur before the first event appears; and the same could occur when the incident finishes, that is, it would be usual to have a delay between the end of the incident and the closing (resolution) of a ticket. Thus, we define a *Forward Offset Delay, FOD*, as the delay between the real start of the incident (SoI) and the time at which the first event appears. In addition, we define a *Backward Offset Delay, BOD*, i.e., the delay between the real end of the incident

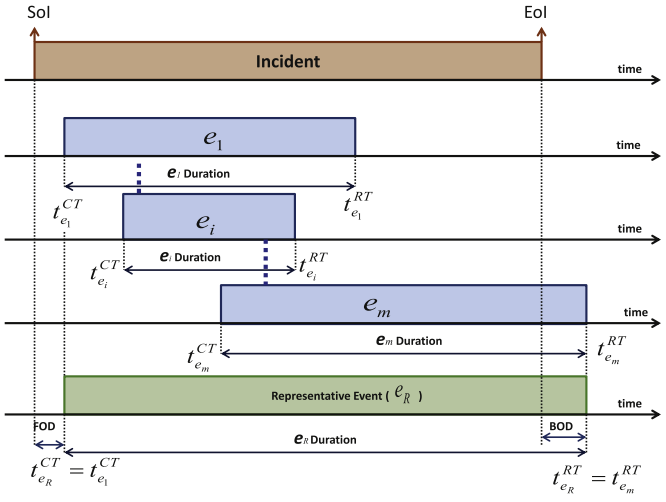

Fig. 1. Correlation of m events belonging to the same incident into a representative event.

(EoI) and the closing time of all the related events. Note that BOD could take a negative value in the case that the last event ends before the incident. As an example, if we consider the events to be tickets that are manually created and closed by a member of the management staff, it could happen that the member believed an incident had finished while it was actually still active. In this case, the member would close the ticket (the end of the event), thus assigning to BOD a negative value. These two measures will be relevant for the proper handling of events that occur near in time, as they are related to the time misalignments between the real incident and its perceived effects. We consider them in the ticket-alert correlation algorithm proposed in Section 5.

3.1. Tickets vs. alerts

Despite the fact that we generally consider both tickets and alerts as *events*, it is important to highlight that there are relevant differences among them and, thus, some challenges appear when trying to combine both.

The main difference between alerts and tickets appears in the nature of the information they contain, mainly due to the own management process. Alerts are automatically triggered by network equipments and systems and, thus, they incorporate automated information. On the other hand, tickets are normally generated by humans, either when a service desk call is received, or when the management staff receives any kind of notification about ongoing incidents. This implies that tickets will incorporate human expert knowledge, while alerts will not. Note that even in the case that tickets are automatically generated when alerts arrive, operators are usually allowed to add expert information.

While it is expected that alerts should be usually associated with tickets, in realistic scenarios we may have alerts without related tickets and *vice versa*. To clarify this point consider an example from the intrusion detection field where a Web server is a victim of a low-rate denial of service (DoS) attack [49,50]. This type of attacks succeeds to defeat application servers only by sending them low-rate traffic in an intelligent way, and can easily bypass detection mechanisms, so no alert is generated by these traffic monitoring systems. In this case, as no alert is created in the system, the incident could remain undetected even when using alert correlation. Yet in this scenario, any Web user might denounce the unavailability of the Web server by calling the service desk. There, an operator would open a corresponding ticket. As a result, we do not expect to have related alerts for every created ticket. In this regard, we have explored real datasets (See Section 6) and found some relevant properties that strongly suggest that they are separated sources with different information.

Furthermore, as tickets and alerts are generated following different processes, they contain different information. Some of the challenges that appear when dealing with the information in both tickets and alerts are:

- Information structure: Information contained in alerts is usually structured. This is due to the fact that it is automatically generated by the monitoring systems. On the other hand, many of the information contained in tickets is not structured, as it is manually generated and maintained by humans.
- Timing information: Alerts are automatically triggered by monitoring systems that are reactive enough to quickly respond and rapidly generate alerts. Thus, the delay between the beginning of an incident and the first alert creation time is almost negligible. Whereas, in ITS, and especially for those tickets that need human intervention, a perceivable and even significant delay in ticket creation can appear. Thus, these timing offsets introduce a real challenge when trying to correlate alerts and tickets. As it will be shown in Section 5, we argue that alerts contain more accurate timing information about incident lifetimes than tickets.
- Semantic information: Since a ticket contains many free text fields, ticket creators and those that manage them feel free to describe the incident, its possible causes and the solutions applied to solve it in more detail. Therefore, tickets are expected to provide better information than alerts with regard to identifying and describing the actual incidents that occur in a network.
- Preprocessing steps: Both datasets may contain duplicated or redundant records. The processes applied to filter and remove these records are different as they present different field types, even including non-structured data in the case of tickets. Furthermore, unlike the alert dataset, the ticketing system is a multipurpose system that can be used not only for incident resolution, but also to register informative data for administrative issues such as scheduled maintenance or system update. This implies different preprocessing criteria and constitutes a big challenge when trying to filter out alerts and tickets not directly related to incidents.

As will be shown in Section 5, applying the basic event correlation model that follows for both tickets and alerts is not straightforward, and the proposed correlation algorithm considers these challenges in its design.

3.2. Basic correlation method

Suppose now that we have a set of m events and that we do not have any information about the incidents they are related to. We are interested in obtaining the same number of representative events as that of the incidents that originated those events. To achieve this in this basic model for event correlation, we assume the following **hypothesis:**

> *Two events that (i) have a similar description or have an affected element in common, i.e., are related to the same network nodes, and (ii) happen simultaneously in time will likely belong to the same incident.*

Mathematically, the first condition, *i.e.*, the similarity in the description or the affected elements of two events e_i and e_j, can be described by the following expression:

$$\{E_i \cap E_j\} \neq \phi \ OR \ \{D_i \cap D_j\} \neq \phi \tag{1}$$

while the second condition, i.e., the simultaneous occurrence of two events e_i and e_j, is given by

$$\left\{t_{e_i}^{CT} \leq t_{e_j}^{CT} \leq t_{e_i}^{RT}\right\} \ OR \ \left\{t_{e_j}^{CT} \leq t_{e_i}^{CT} \leq t_{e_j}^{RT}\right\} = true \tag{2}$$

Note that the conditions $E_i \cap E_j$ and $D_i \cap D_j$ represent the intersection of two text-free fields. Although there exist many alternatives to determine a metric to decide if an intersection is present [51], we have opted for a very simple model to justify that, even in these conditions, the inclusion of tickets provides benefits. In case of the condition $E_i \cap E_j$, as will be detailed in Section 6, every list E_i is built by mining text-free fields and searching node identifiers present in a predefined list. Thus, two fields E_i and E_j intersect when they contain at least one identifier in common. Regarding the condition $D_i \cap D_j$, we have included this condition in the model for the sake of completeness. However, to make our approach as simple as possible, we do not truly apply it in our experiments such that no comparison of free text fields regarding descriptions is conducted at all. That said, a more advanced algorithm could be selected from the proposed solutions for text mining in the literature [51].

In our basic correlation algorithm, if the above two rules are fulfilled by any group of events, we conclude that they are all related to the same incident, and we simply aggregate them into one representative event, e_R, having the following properties (see Fig. 1):

$$t_{e_R}^{CT} = \min_{i \in \{1,m\}} \left\{t_{e_i}^{CT}\right\} \tag{3}$$

$$t_{e_R}^{RT} = \max_{i \in \{1,m\}} \left\{t_{e_i}^{RT}\right\} \tag{4}$$

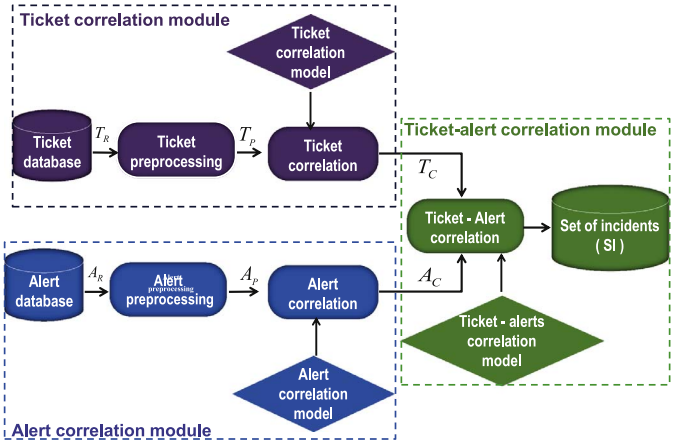

Fig. 2. Proposed architecture for the ticket-alert correlation system.

$$E_{e_R} = \bigcup_{i=1}^{m} E_i \tag{5}$$

$$D_{e_R} = \bigcup_{i=1}^{m} D_i \tag{6}$$

Note that we join all of the descriptions D_i of the different events and the set of event-affected elements, E_i, as we consider that any information in one of the events in an incident will complement the information provided in other events in the same incident.

4. Ticket-alert correlation system

Fig. 2 shows the proposed ticket-alert correlation system. It mainly consists of three modules: a module for ticket correlation, another for alert correlation and the last for ticket-alert joint correlation.

The ticket correlation module is represented in the upper part of the figure. A set of *raw* tickets, T_R, obtained from the ITS is entered as an input to the *ticket preprocessing* phase to normalize and extract only relevant tickets, as will be explained next. The resulting processed set, T_P, is then passed through a *ticket correlation* phase, which, based on the basic model for event correlation explained in Section 3, produces a new set of *representative tickets,* T_C. Every representative ticket, which is ideally expected to represent a single incident, contains the summary of a group of correlated tickets.

The lower part of Fig. 2 represents the alert correlation module. Here, a set of *raw* alerts, A_R, usually triggered by a monitoring system, is entered as input to an *alert preprocessing* phase to normalize and extract only relevant alerts, as will be explained next. The resulting processed set, A_P, is then passed through an *alert correlation* phase, also based on the basic model for event correlation (Section 3), to produce a new set, A_C, which is the final set of *representative alerts*. Every representative alert is expected to ideally represent a single incident, containing a summary of the information provided by a group of correlated alerts.

Finally, the right-hand side of Fig. 2 represents the ticket-alert correlation module. Here, the outputs of the alert and ticket correlation modules, A_C and T_C, are entered as inputs, and the processing is performed according to the correlation model presented in Section 5. The aim is to produce a final set of incidents, I, that will more accurately represent the real incidents in the network compared with methods that only account for alert correlation. In the following subsections, we provide a more detailed discussion regarding each module separately.

4.1. Ticket correlation module

The inputs for this module are the tickets obtained from an ITS database. ITSs are considered essential tools for tracking resolution activities associated with incidents in corporate networks. Each record in an ITS represents a ticket that has information related to an incident. Normally, an incident is perceived by the management staff by observing events generated by monitoring software or by receiving customers' complaints. These events, called tickets, contain information such as *Node IDs* (*affected elements* in our basic event correlation model), which are identifiers of the main network element/s or service/s affected by the incident reported in the ticket; *ticket timestamps*, such as ticket creation and resolution times; and *incident description fields* (*descriptions* in our basic event correlation model), such as an incident summary, a worklog history and a solution description containing all the procedures used to solve the incident. Tickets may also contain fields storing administrative information, such as the management groups involved in the resolution process, along with their contact information, among things.

In this module, the tickets are first introduced in a *ticket preprocessing phase*. The aim is to extract only *relevant tickets*, or tickets related to real network incidents. It is worth noting here that ITSs are used as dual-task systems: they can be used for purposes other than registering incidents' lifecycles. For example, they are normally used to record other administrative and maintenance tasks, e.g., programmed work in the network or availability of a new software release. Therefore, because not all tickets are created as a consequence of actual network incidents, we might say that some tickets are *informative*. Thus, we first normalize the data and obtain in a proper format the different tickets' fields that are relevant from the point of view of incident solving and remove the remaining information in every ticket. Second, we discard *malformed tickets*, that is, tickets having some incoherent values, as well as *informative tickets*. To discern which tickets are informative, a common method is to use a pre-defined list of keywords and pattern-matching techniques.

The output of the ticket preprocessing phase is fed as an input to the *ticket correlation phase.* This process is well studied in our previous work [52]. Essentially, we apply our basic event correlation algorithm (See Section 3) with several adaptations. In a first step, we obtain the main *affected element* of every ticket, usually clearly stated in a field called "Node ID" in ticketing systems, and make the correlation considering all the tickets that have only this affected element. In a second step, we obtain more information regarding *affected elements* from other tickets' fields related to the description of the incident, such as worklogs and solution descriptions. In [52], we show that this leads to a considerable improvement in the incident resolution process in terms of accuracy, timing, and incident description.

4.2. Alert correlation module

Alerts are usually generated by network elements and obtained by management platforms, e.g., syslog or HP OpenView, using management protocols, such as SNMP. Each alert is a short message with a specific textual format defined by equipment vendors and is generated as an external manifestation of a potential failure or disorder occurring in a piece of equipment of the managed network or system. Typically, alerts contain the same relevant information as that described in our basic model presented in Section 3, such as *(i) affected element* identifier, e.g., `node ID` and `interface ID`, *(ii)* the timing information of the alert, i.e., the creation and resolution times, and *(iii)* a description of the fault, i.e., the root cause and the severity of the alert. Moreover, alerts may provide information with different detail levels, such as specific data regarding the status of the devices and their configurations or higher level details with aggregated information gathered from several alerts.

Alerts are first passed to a *preprocessing phase*, with the aim of selecting only *relevant alerts*: alerts related to relevant incidents. It is worth mentioning here that today's monitoring systems trigger a huge amount of so-called *normal-behavior alerts* in response to daily operational tasks that are not really related to real network incidents, *i.e.*, maintenance activities, software updates, etc. Thus, to filter

nonrelevant alerts, we use a pre-defined list of keywords and some pattern matching techniques. In the experimental section we provide more details about this process.

The output of the preprocessing phase is fed into an *alert correlation phase*. Here, a similar approach to that followed in the ticket correlation phase is utilized: we use the basic event correlation model (Section 3) in two steps. First, we only consider alerts that are related to a single *affected element*, and in a second step, we incorporate those alerts that are related to a list of several *affected elements*. These last alerts are normally generated by intermediate network elements that really correlate several of them and generate a new alert with the summarized information.

Note that this module is very similar to the ticket correlation module. It is remarkable to say that traditionally, this is the only module that has been implemented in network alert correlation systems, and a large amount of research effort has been devoted to studying it [5,7,9,10,19]. In our case, we are not as interested in refining this module as in evaluating whether the incorporation of tickets' information would improve the alert correlation process. For this reason, and for ease, we have opted for this implementation.

4.3. Ticket-alert correlation module

This module works with the information provided by both the alert and ticket correlation modules. As previously explained, we are mainly interested in evaluating whether introducing this module would result in a benefit in the correlation process.

Our intuition is that the tickets can introduce relevant information in the procedure, incorporating human knowledge and significance to the events. An example of a scenario revealing this is depicted in Fig. 3. In the first timeline, we can observe the result from an alert correlation process, where three clusters of alerts are summarized in three representative alerts: A_{R1}, A_{R2} and A_{R3}. The second timeline represents the output from the ticket correlation process, where a single representative ticket, T_R, has been obtained. The third timeline represents the duration of the incident that generated the different events (alerts and tickets).

Note that the alerts in this incident appear intermittently, which makes the correlation process consider that they are not overlapped in time and are thus not likely to belong to the same incident. However, the existence of tickets makes it possible to observe the concurrence in time between the three groups of alerts and tickets, thus allowing the correlation of all of them to represent a single incident.

We can further clarify this example by using a real scenario with alerts and tickets taken from the dataset that we analyze below in Section 6. Fig. 4 shows the set of alerts after preprocessing (24 alerts) that are considered in this example. These alerts are triggered by two different network nodes, namely NIX1-FORTIGATE and AVPN-CEIC-039, with 20 and 4 alerts, respectively. The basic alert correlation

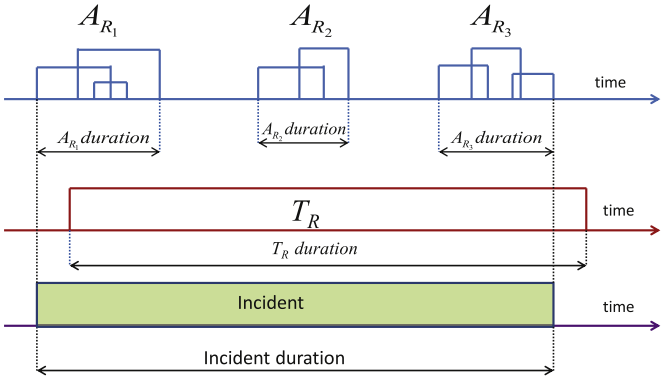

Fig. 3. Example of alert aggregation using the proposed model.

Node _ID	Interface _ID	Description	Creation_Time	Resolution_Time
NIX1-FORTIGATE	CICE-SEV-039	InterfaceDown	Fri Nov 1 01:23:06 2013	Mon Nov 4 15:16:10 2013
NIX1-FORTIGATE	AVPN-SAS-139	InterfaceDown	Fri Nov 1 10:59:07 2013	Fri Nov 1 11:28:09 2013
NIX1-FORTIGATE	AEXT-SAS-117	InterfaceDown	Fri Nov 1 12:29:08 2013	Mon Nov 4 06:49:14 2013
NIX1-FORTIGATE	AVPN-SAS-166	InterfaceDown	Fri Nov 1 13:53:08 2013	Sat Nov 2 14:31:09 2013
NIX1-FORTIGATE	AVPN-SAS-148	InterfaceDown	Fri Nov 1 16:41:07 2013	Fri Nov 1 16:43:09 2013
NIX1-FORTIGATE	AVPN-SAS-148	InterfaceDown	Fri Nov 1 17:59:07 2013	Fri Nov 1 18:01:10 2013
NIX1-FORTIGATE	AVPN-SAS-148	InterfaceDown	Fri Nov 1 19:41:07 2013	Fri Nov 1 19:44:07 2013
NIX1-FORTIGATE	SAS-SEV-076	InterfaceDown	Sat Nov 2 06:41:08 2013	Sat Nov 2 12:04:09 2013
NIX1-FORTIGATE	AVPN-SAS-138	InterfaceDown	Sat Nov 2 14:38:10 2013	Sat Nov 2 14:40:10 2013
NIX1-FORTIGATE	AVPN-SAS-148	InterfaceDown	Sat Nov 2 16:41:09 2013	Sat Nov 2 16:43:09 2013
NIX1-FORTIGATE	AVPN-SAS-148	InterfaceDown	Sun Nov 3 17:50:06 2013	Sun Nov 3 17:53:06 2013
NIX1-FORTIGATE	AVPN-SAS-148	InterfaceDown	Sun Nov 3 18:20:08 2013	Sun Nov 3 18:23:07 2013
NIX1-FORTIGATE	AVPN-SAS-157	InterfaceDown	Mon Nov 4 14:14:48 2013	Mon Nov 4 14:16:14 2013
AVPN-CEIC-039	AVPN-CEIC-039	InterfaceDown	Fri Nov 1 01:26:07 2013	Fri Nov 1 07:31:09 2013
AVPN-CEIC-039	AVPN-CEIC-039	InterfaceDown	Fri Nov 1 01:27:11 2013	Fri Nov 1 01:30:13 2013
NIX1-FORTIGATE	CICE-SEV-039	InterfaceDown	Mon Nov 4 21:44:09 2013	Tue Nov 5 07:25:10 2013
NIX1-FORTIGATE	AEXT-SAS-115	InterfaceDown	Tue Nov 5 00:05:07 2013	Tue Nov 5 00:40:13 2013
NIX1-FORTIGATE	AEXT-SAS-115	InterfaceDown	Tue Nov 5 00:47:10 2013	Tue Nov 5 10:07:21 2013
NIX1-FORTIGATE	AVPN-SAS-157	InterfaceDown	Tue Nov 5 02:44:09 2013	Tue Nov 5 02:47:07 2013
NIX1-FORTIGATE	AEXT-SAS-117	InterfaceDown	Tue Nov 5 08:56:08 2013	Tue Nov 5 09:01:40 2013
AVPN-CEIC-039	AVPN-CEIC-039	InterfaceDown	Tue Nov 5 03:21:11 2013	Tue Nov 5 03:29:04 2013
AVPN-CEIC-039	AVPN-CEIC-039	InterfaceDown	Tue Nov 5 03:27:08 2013	Tue Nov 5 04:01:10 2013
NIX1-FORTIGATE	CICE-SEV-039	InterfaceDown	Tue Nov 5 22:53:06 2013	Wed Nov 6 08:46:14 2013
NIX1-FORTIGATE	SAS-SEV-061	InterfaceDown	Wed Nov 6 00:44:06 2013	Wed Nov 6 02:55:09 2013

Fig. 4. A snapshot of the set of alerts considered in the example.

operations that we applied here are able to group this set of alerts into five clusters (3 for NIX1-FORTIGATE, and 2 for AVPN-CEIC-039) based on the timing and some topological information (basically Node ID). The first 13 alerts are overlapped in time and have the same common Node ID (NIX1-FORTIGATE). Thus, they are grouped into one representative alert aggregating all of them (Cluster # 1). The next 5 are also grouped into a single "event" that represents this group (Cluster # 2); finally, the last 2 are treated in the same way (Cluster # 3). Regarding the alerts generated by node AVPN-CEIC-039, the correlation method generates 2 clusters (# 4 and # 5). Thus, the output after alert correlation would be composed of five groups of aggregated alerts. Fig. 5 shows the corresponding ticket for this set of alerts after preprocessing and extracting the useful features. As we will discuss in Section 6, it is worth noting that in the considered ticketing system, every ticket is characterized by 273 different fields. Thus, only the relevant fields are shown in Fig. 5. Observing the ticket, it is clear that this information intersects with that from alerts in several fields, e.g., Node ID, Description, Interface ID, and timing information.

Therefore, as a primer solution, we could use these fields to correlate both sources of information as shown in Fig. 6. In this figure, we draw over time the ticket and the considered 24 alerts. In the upper timeline we draw the ticket lifetime considering both creation and resolution times. In the second timeline we draw the 24 alerts and show their overlapping periods and the five created clusters after applying the basic event correlation model (see Section 3.2). Without considering ticket information, the basic event correlation model would

Staff_ID	CAU 24X7	***Ticket_ID***	HD000000082221	***Node_ID***	AVPN-CEIC-039
Creation_Time	Fri Nov 1 05:47:48 2013	***Resolution_Time***	Wed Nov 6 12:23:12 2013		
Incident Description	Alarm OVO important --X-X-2:20:31 1/11/13 nix1-fortigate NIX CICE-SEV-039-1M Interface Down				
Affected_Nodes	AVPN-CEIC-039, NIX1-FORTIGATE, 62.15.232.121, 10.118.72.1, 194.224.229.1				
Worklog	Fri Nov 1 06:55:42 2013 surveillance3.24x7 PDT equipment status. Fri Nov 1 08:40:10 2013 surveillance 2.24x7 Pending contact on Monday at Sat Nov 2 13:39:12 2013 schedule? Vigilance 4.24x7 Pending review of the other end. We check connectivity with the other end: Ping statistics for 194.224.229.1: Packets sent = 4, received = 4, lost = 0 (0% loss) Approximate times of return in milliseconds: Minimum = 22ms, Maximum = 40ms, Mean = 26ms Ping statistics for 62.15.232.121: Packets sent = 4, received = 4, lost = 0 (0% loss) Approximate times of return in milliseconds: Minimum = 26ms, Maximum = 28ms, Mean = 26ms				
Solution Description	Ping statistics for 194.224.229.1: Packets: sent = 4, received = 4, lost = 0 Ping statistics for 62.15.232.121: Packets: sent = 4, received = 4, lost = 0 Ticket HD000000082221 has been closed automatically on 07 / 11/13 10:43:57. The solution used is recovered.				

Fig. 5. A snapshot of the ticket considered in the example.

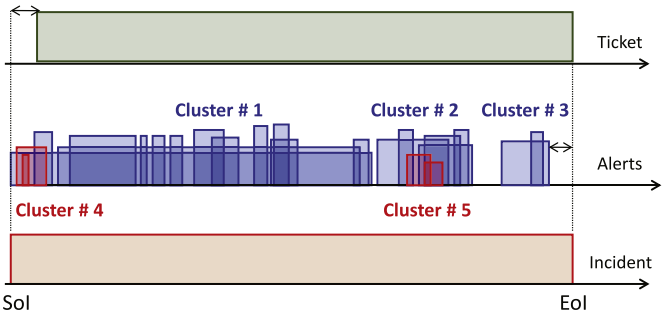

Fig. 6. A graphical time representation of the alerts, the ticket, and the generated incident.

create five uncorrelated clusters of alerts despite being related to the same incident. This is due to the fact that in real scenarios like the example it is usual to have unpredictable time gaps between alerts that make it hard to guess their relationship. Besides, it is hard to correlate alerts coming from different network nodes. In our case, by incorporating the ticket information we observe that the five alert clusters overlap with the ticket, and consequently we can consider that they belong to the same incident. The final number of events to be considered in this example is then reduced to 1 instead of 5, which supports the idea that using the ticket makes it possible to correlate them together (see incident timeline in Fig. 6).

This illustrative example gives clear indications of the beneficial outcomes provided to both technical and decision-making staff. For example, from the point of view of efficiency, having a single incident with more information might reduce the resolution time. From an audit perspective, having more realistic information about the real incidents solved by management staff will help in the decision-making process.

5. Ticket-alert correlation model

To apply the basic event correlation model suggested in Section 3 to correlate both tickets and alerts, it is important to first understand the specificities of both tickets and alerts (see Section 3.1) and then properly design a correlation algorithm able to handle the associated challenges. In the following, we first discuss the specific issues to be taken into account and then suggest our proposal for the correlation algorithm.

- *Tickets provide better semantic information than alerts*
  Although tickets can be automatically generated by the NMS (automatic tickets), they are usually generated manually by the members of the staff, either as a response to alerts or from customers' complaints. Every ticket represents a complete record of an incident to be used by the management team during the incident management lifecycle.
  Normally, tickets contain more semantic information about the incidents than alerts. First, every ticket contains many free text fields, which are used by ticket creators and resolvers to clearly describe the incident, its possible causes and the solutions applied to solve it. In alerts, these fields are normally automatically generated by network facilities, and thus the semantic information is very restricted to a list of possible values. Second, tickets are generated by humans only when alert events are considered so important that a record of an incident is needed. For example, the appearance of alerts regarding non-production services, alerts generated by low-priority nodes in a network, or warning alerts of low-priority should not cause the creation of tickets, as these events should not be considered as incidents.
  Thus, tickets are expected to provide better information than alerts with regard to identifying the actual number of incidents that occur in a network. If we assume that the number of incidents derived from a ticket correlation process is I_T, the number of incidents noted by an alert correlation process is I_A, and the number of real incidents is I_{actual}, we expect to have the following relation:

$$I_{actual} < I_T < < I_A \tag{7}$$

  For this reason, we show in our correlation algorithm that we pay more attention to tickets when deciding the number of incidents.
- *Alerts provide better temporal information than tickets*
  In contrast with our higher confidence in the semantic information contained in tickets, we claim that the temporal information found in them is less trustworthy than that provided by alerts. This is because in the ITS system, a large number of tickets are created or closed manually by the management staff; thus, their responsiveness is not as fast as in the alert management system where alerts are generated automatically in a few milliseconds when an event is perceived. Thus, to determine the beginning and end times of an incident, we consider the timestamps provided by alerts to be the best approximations.
  Going back to Fig. 2, in which we show the complete system, note that instead of considering both tickets and alerts as general events and applying our basic event correlation algorithm to the complete set, we separate both into two processes. This will allow us to determine and identify the number of different incidents (and their semantic information) from the ticket correlation process and adjust their temporal information from the feedback provided by the alert correlation process. In summary, the output of ticket correlation is refined with the alert correlation output to adjust the time information of the incidents noted by tickets.
- *Dealing with border effects and the existence of consecutive incidents*
  As previously mentioned, it is expected to have some temporary misalignments between the start and end of an incident and the opening and closing times of the associated ticket(s) due to, presumably, human response times. This effect has been included in the basic correlation model through the parameters FOD and BOD. At first sight (Fig. 7), the problem with these two delays is that it is possible to exclude or include events related or not really related, respectively, to the ongoing incident in the representative event and, therefore, in the incident as perceived after the correlation process. As depicted in the example in Fig. 7, depending on whether the first event in the events line is an alert or a ticket, it is even possible for the appearance of two different incidents at the beginning, while two different incidents can be merged at the end. In contrast, it is also possible for the staff members to prematurely close a ticket if they have the perception that the problem is solved, thus making BOD negative. In this case, it is highly probable that another ticket truly related to the same incident will appear after some delay. In fact, this is exactly the former situation in the case that the first event in the event line is a ticket.

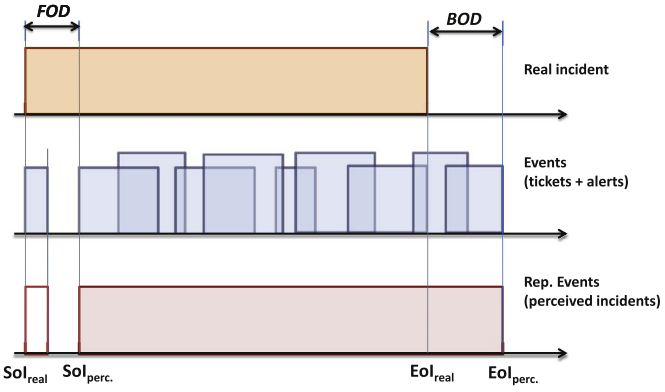

Fig. 7. Potential effects of FOD and BOD on the correlation results.

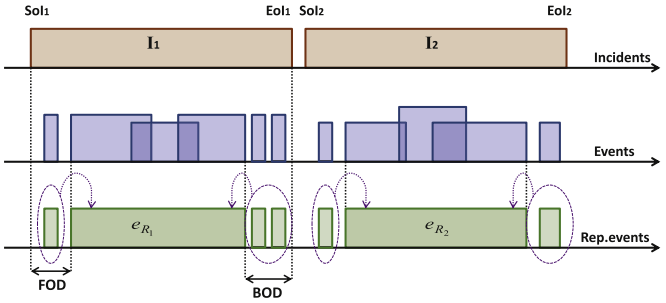

Fig. 8. Example showing the problem of directly applying the basic event correlation model to two consecutive incidents.

To handle this situation, our main argument is that there is a high probability that potentially correlated events that occur in the proximities of other representative events really belong to the same incident. This way, the simultaneity condition –Eq. (2)– used to merge events is relaxed by considering FOD and BOD as thresholds to consider alerts and tickets in the proximity as included in the same incident.

From the point of view of the correlation method, the major impact is expected to arise from the "orphan" alerts, that is, from that alerts at the beginning or end of an incident that are not assigned to it due to the border effects. Therefore, some experimental tuning is needed to estimate the values for both FOD and BOD. Obviously, this is addressed in the experimental setup.

Nevertheless, the scenario can become a bit more complex when consecutive incidents appear. The problem is how to discriminate between any two consecutive incidents having some properties in common, i.e., how to correctly separate events that could correspond to both incidents or even decide that both incidents are the same and should be merged. To clarify this point, we show an example in Fig. 8. Here, we assume that we have two truly consecutive incidents, I_1 and I_2, each one starting and ending at the instants shown in the *incidents* timeline. We also have a sequence of events, each one starting and ending as shown in the *events* timeline. Furthermore, we assume that each of these events is related to either I_1 or I_2. If we apply the basic event correlation model suggested in Section 3, we obtain two representative events, e_{R1} and e_{R2}, as shown in the third timeline (*Rep. events*), which, at the same time, will be considered the incidents from our point of view.

If we look carefully at this example, we observe that some events are discarded from the correlation process and are not correlated simply because they do not overlap with any other event. We can assume that the non-overlapped events belong to other different incidents, in which case we would have up to seven different incidents, far more than the actual two incidents.

Thus, directly applying the basic event correlation model in this example would lead to inaccuracies, especially when alert events are considered because, as mentioned above, alerts may appear earlier than tickets and might not overlap with them, and they may not be considered in the correlation process.

In addition, note that there is another problem when consecutive incidents are considered as in our example. We must select the specific incident, if any, to which the events in between belong. In our example, there are three events between e_{R_1} and e_{R_2}. The choice of assignment between event-incident modifies the duration of both incidents, thus affecting the accuracy of the system.

5.1. Ticket-alert correlation algorithm

To handle all of the above issues, we modify the basic correlation model to consider non-overlapped subsets of tickets and alerts as explained before. As shown in Fig. 1, FOD and BOD will be used as extra delay thresholds so that an event is correlated to a representative event, e_R, that is active in the time interval $[t_{CT},\ t_{RT}]$ if that event satisfies Eq. (1), that is, if it satisfies the similarity criteria and is active in the interval

$$[t_{CT} - FOD,\ t_{RT} + BOD] \tag{8}$$

Note that, with the expansion of the intervals with FOD and BOD, it might happen that the extended intervals of two consecutive incidents sharing some affected element overlap. In this case, two operations are considered: (i) any potentially related event falling in these intervals will be assigned to the nearest-in-time representative event, and (ii) the incidents will be merged only if, after adding the in-between events, they are overlapping. Thus, extended intervals are not considered valid for merging incidents simply based on the new limits.

It is obvious that the selection of the values of FOD and BOD directly affects the performance of the correlation algorithm. In Section 6, we show how to experimentally determine optimal values for these parameters and how they affect the overall results.

In summary, we propose an algorithm (Listing 1) that starts from an empty list of incidents and consists of two iterations. First, it takes every representative ticket from the correlated tickets set, T_C, each of which is considered to represent different incidents. It is worth noting that as a result of the ticket correlation, none of those tickets are overlapped in time. For each incident (or representative ticket), the different model parameters are extracted (creation and resolution times, affected elements and descriptions). Then, it searches for correlated representative alerts. Thus, every representative ticket has a list of correlated alerts assigned to it, and the temporary limits (SoI and EoI) of the incidents are revisited according to the new information. Second, after extracting a tuple of correlated tickets and alerts, in the second iteration, the algorithm takes every correlated ticket and its associated list of correlated alerts and searches for other tickets having at least one alert in common. The target of this step is to join the representative tickets that resulted in being overlapped after adding the alerts in the first step. All matched tickets are aggregated into a single one including all the information from the entire group. Finally, SoI and EoI are determined by, respectively, taking the first of the creation times of any of the alerts in the correlated set or the ticket creation time (line 42) and the last of the resolution times of any of the alerts or the ticket resolution time (line 43).

The final output is a set, S_I, of k incidents, such that $S_I = \{I_1, I_2, I_3, ..., I_k\}$, being a single incident $I = \{T_C, A_C, T_E, T_D\}$, where T_C and A_C are the subsets of correlated tickets and alerts for this incident, respectively, T_E is the list of affected elements, and T_D is the description of the incident. This set S_I is the estimation of the actual incidents represented by all the events (tickets and alerts).

6. Experimental results

In this section we present the experimental assessment of the proposal using datasets of tickets and alerts captured in a real production network. According to the main target of the paper, our purpose is to check the usefulness of the joint use of tickets and alerts in the event correlation procedures. For this, we must tune the parameters related to the potential misalignments between incidents, tickets, and alerts timing information, i.e. BOD and FOD, in order to obtain the set of representative events for the available dataset.

On the other hand, two questions have to be addressed to assess the results: (1) whether the original events combined in each representative event are really associated to that representative, and (2) whether all the events associated to an incident are included in the representative event. The answer to these questions is not straightforward due to the lack of a ground truth to account for each class. Therefore, we developed a strategy in three steps, as described in Section 6.2, to address these questions.

```
procedure: TicketsAlertsCorrelation

  Initialize: incidentList()=null, FOD, BOD
  // First step: correlate alerts to tickets
  loop      // For each representative ticket
    ticket = getNewRepresentativeTicket( )
    L = listOfAlerts(ticket)
    T_CT = getTicketCreationTime(ticket)
    T_RT = getTicketResolutionTime(ticket)
    T_E = getListAffectedElements(ticket)
    T_D = getListDescriptions(ticket)
    loop   // Searching for correlated alerts
      alert = getNewIncidentAlert( )
      A_CT = getAlertCreationTime (alert)
      A_RT = getAlertResolutionTime(alert)
      A_E = getAlertAffectedElement(alert)
      A_D = getAlertDescription(alert)
      if ((A_CT ≥ T_CT − FOD) AND (A_RT ≤ T_RT + BOD))
          AND ((A_E ∈ T_E) OR A(_D ∈ T_D)):
          L.add (alert)
    end loop
    SoI = min(getFirstCreationTimeAlerts(L_i),T_CT)
    EoI = max(getLastResolutionTimeAlerts(L_i),T_RT )
  end loop
  // Second step: merge newly related tickets
  for i=1 to m: // m number of correlated Tickets
    ticket_i = getCorrelatedTicket( )
    L_i= getlistOfCorrelatedAlerts(ticket_i)
    T_CT = getTicketCreationTime(ticket_i)
    T_RT = getTicketResolutionTime(ticket_i)
    for j=i+1 to m: // get other correlated ticket
      ticket_j = getOtherCorrelatedTicket( )
      L_j= getlistOfCorrelatedAlerts(ticket_j)
      if L_j ∩ L_i ≠ φ:
        L_i.add(L_j)
        T_E_i.add(T_E_j)
        T_D_i.add(T_D_j)
        del ticket_j, L_j
        CT = getTicketCreationTime(ticket)
        RT = getTicketResolutionTime(ticket)
        SoI = min(getFirstCreationTimeAlerts(L_i), T_CT, CT)
        EoI = max(getLastResolutionTimeAlerts(L_i), T_RT, RT)
    incidentList.add (SoI, EoI, T_E_i, T_D_i)
end procedure
```

Listing 1. Ticket-alert correlation algorithm.

6.1. Real scenario: Dataset and preprocessing

The scenario considered is the network, event data and procedures handled by a management company with which we are collaborating. This company is in charge of the supervision, from a management point of view, of the operation of the corporate network providing services to the regional government. Thus, the supervised network serves millions of habitants from many public sectors, such as education, health and civil. The topology of the network resembles the organizational structure of the government and thus uses a hierarchical approach.

As previously mentioned, the handling of alerts and tickets inside the company is performed by two different departments: a technical department, which is in charge of the effective supervision of the network and, subsequently, the alerts; and a service desk, which attends the requests and complaints from customers. In what follows, we refer to the technical department as MS (Management Staff) and to the Service Desk as SD. Both departments have the capability to create tickets, although, as previously mentioned, MS do it mainly as a response to alerts generated by some network failures or after receiving some feedback from other sources, whereas SD will create tickets from end-user complaints. This is an important fact from the point of view of the current work, as it is expected that many of the tickets created by MS have some pointers to the alerts triggering them, while the tickets created by SD can provide richer information regarding the ongoing incidents but no pointers to technical details or alerts is expected.

The data gathered in this scenario are the set of alerts and tickets in the system during a period of six months spanning from October 1, 2013, to the end of March, 2014. It is important to note that no information regarding the topology, apart from a list of nodes and links classified by MS as critical for the operation of the network, is considered.

The information included in each alert is presented in a fixed structured format (Fig. 9), which is mapped to the elements in the model (Section 3). It is relevant to mention that the affected element name (Object ID in the alert dataset) presents a hierarchical structure that fits with the topological counterpart. Alternately, tickets are far more complex because they also include additional information related to incident tracking and solution and various free text fields. Nevertheless, it is easy to map some of these fields to the elements in the model.

A major inconvenience from this raw dataset that conditions the manner in which the experiments are to be carried out is the lack of a

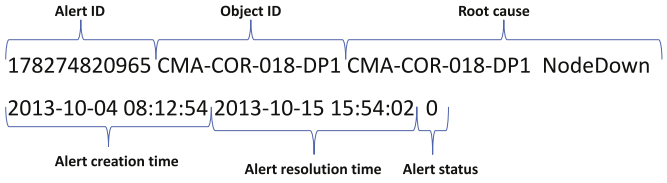

Fig. 9. Example of an alert format taken from the alert dataset.

"ground truth," i.e., a set of labeled incidents with their corresponding alerts and tickets is not available. Furthermore, and as expected, we manually found that not all of the tickets/alerts are related to real incidents and that some of the tickets present incoherent or null values in relevant fields. These types of alerts/tickets were removed during the preprocessing step, for which we set up some rules, mainly based on keyword detection, after some consultations with the staff at the company.

The lack of labeled data can introduce some confusion in the interpretation of the results, as not all the real alerts are to generate tickets because the MS can consider them nonrelevant at a given time. In fact, alerts in NMS are usually classified according to their severity and/or criticality. Thus, not all of the alerts present the same effects on the stability of the managed system, and MS is prone to ignore or postpone the creation of a ticket for non-critical alerts, especially if they are busy trying to solve an incident with higher priority. As a consequence, even if the number of tickets was accurate, not all of the alerts would be correlated to a ticket, i.e., to an incident, which could be interpreted as a failure in the proposed method.

To address this situation, we have checked the performance of the correlation method with a special subset of events, that is, we only consider *relevant incidents*. By relevant incidents, we refer to those that affect the operation of the network in a critical way and that, consequently, must present associated tickets. According to the company's technical procedures, there are two situations in which an alert should mandatorily trigger a ticket from MS: *critical alerts*, which are those truly affecting critical elements, and *massive alerts*, which are alerts automatically generated by the NMS as a response to a large number of alerts from topologically related elements in the network. The first situation is identified by using a list of network nodes (the critical ones) and the type of critical alerts (i.e., *NodeDown, InterfaceDown*) such that a critical alert in a critical node should generate a ticket by MS. Therefore, the main criterion for measuring the performance of the proposed correlation procedure can be stated as

All relevant alerts should be assigned to a ticket.

Thus, during the preprocessing phase the *relevant alerts* are extracted, and the tickets related to critical nodes are identified. To correlate the tickets (Section 4.1), we adopt the same methodologies proposed in our previous work [49] to preprocess and correlate tickets that have the same root causes. As a result, a set of *representative tickets,* T_C, arguably composed of a single ticket per network incident, is obtained. For the alert database, the processed alerts are entered into the correlation phase (Section 4.2), in which the proposed alert correlation model is applied to obtain the set of *representative alerts,* A_C. Similar to the ticket case, a two-step procedure is used for alert correlation. In the first step, the correlation is applied to critical alerts. Then, massive alerts are also considered. Both datasets are the input for application of the ticket-alert correlation model presented in Section 5.

Some relevant figures for both the tickets and the alerts used during the phases previous to the joint ticket-alert correlation are provided in Table 2. The first row 'Total number of records' represents the size of raw datasets taken from the IT management company. 'Mean number of affected elements/record' provides the number of *officeIDs* found in each record. To extract this number, we applied pattern matching methods to obtain the list of affected elements for every event (ticket or alert). 'Number of records after preprocessing' represents the remaining number of records after the preprocessing step, mainly filtering out void/spamming events. 'Number of relevant records' gives the number of records that contain at least one critical *officeID*. As previously explained, these are the records used to assess the system. 'Number of representatives (before joint correlation)' is the number of representative events in each correlated set after the first step of the correlation, that is, before applying the ticket-alert correlation module, i.e., the number of alerts/tickets considered independent. Finally, the last row of the table provides the average number of records for each representative one.

Table 2
Number of records in the dataset and results for the independent correlation of alerts and tickets.

Database	Alerts	Tickets
Total number of records	1,703,662	9612
Mean number of affected elements/record	1.15	1.42
Number of records after preprocessing	913,042	8105
Number of relevant records	7436	520 (348 MS / 172 SD)
Number of representatives (before joint correlation)	3022	256 (194 MS/ 62 SD)
Mean number of records / repr. set	2.46	2.03

6.2. Evaluation of the system

The evaluation of the performance of the proposed method is not straightforward because, as previously stated, the dataset lacks a ground truth in which the existing incidents are related to their corresponding alerts and tickets. Thus, despite some figures of merit being obtained, a strategy to assess the results should be designed. A trivial approach would be to manually label the dataset, or a part of it, to properly check the obtained correlations. However, even if an expert could manually handle that huge volume of data, the process would be prone to errors because it is frequently difficult to determine, even for an experienced manager, what the real incidents are from the limited information in the alerts and tickets and whether they are related. Therefore, other approaches should be explored.

Alternately, as explained in Section 5, it is necessary to consider some "border effects" in the correlation procedure due to potential misalignments between the real timing of an incident and its manifestation in tickets and alerts, probably due to the humans involved not being reactive enough. To handle this, the model uses two tunable parameters: FOD, that is, the maximum accepted time from the appearance of the first alert of an incident and its corresponding ticket; and BOD, that is, the maximum accepted time from the end of the incident and the closing of the ticket. Obviously, these two parameters should be adjusted during the experimentation, which introduces an additional degree of complexity for evaluating the proposal.

Therefore, we developed a strategy in three steps with different targets: first, we assessed the precision of the correlation, in terms of the number of correctly correlated elements; second, we analyzed the potential impact of varying FOD and BOD, which is somehow related to the recall; and, finally, we manually explored the reasons for portions of the alerts and tickets not being correlated. The first step is related to question (1), while the second and third are related to question (2).

6.2.1. Precision estimation

Once the alerts and the tickets have been independently correlated (See Table 2), we proceeded with the joint correlation of the obtained representatives for alerts and tickets. As previously mentioned, we assume that every critical incident should be univocally related to a final representative ticket. Therefore, in the next discussions, we refer to an incident as a representative ticket after the joint correlation.

In this regard, given the set of representative events and the events (alerts and/or tickets) associated to them, the precision of the correlation, P, can defined as the rate between the number of events correctly associated to any of the representative events (TP) against the total number of events associated to any of the representative events ($TP + FP$), being FP the number of events incorrectly associated to any of the representative events, that is, $P = TP/(TP + FP)$.

In a first step, we varied FOD from 0.025 to 256 h, obtaining a slightly different number of incidents for each value of FOD and different numbers of alerts and tickets associated to each of the incidents.

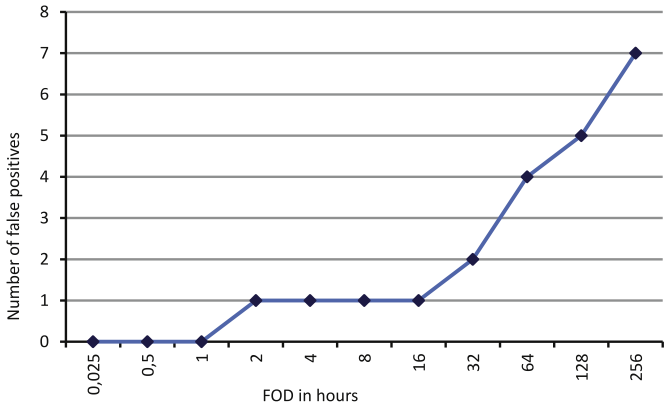

Fig. 10. Number of incorrectly correlated incidents (false positives) for different values of FOD.

At this point, the problem is to assess whether all of the alerts and tickets that have been correlated into a representative ticket are truly related to that incident. Due to the lack of a ground truth, validating these results is not straightforward. To overcome this issue, we estimated the precision of the correlation by manually inspecting many samples and applying the knowledge and rules of thumb provided by the company management team, which helped us during this procedure. Furthermore, for the cases that were not sufficiently clear for us, we obtained additional feedback from the company.

Consequently, for validation purposes, a set of 100 randomly chosen incident samples, together with their correlated tickets and alerts, is considered and studied manually. The result of the analysis is depicted in Fig. 10, which shows the number of false positives, in terms of incidents for which we found inappropriately assigned events, as a function of the value for FOD. It is relevant to mention that for values of FOD between 2 and 16 h, we found that 99 of the 100 samples were undoubtedly classified as correctly correlated. The remaining sample is not a clear case, as there is not enough information in the ticket and the alerts to decide whether they are really related or not. Thus, assuming the worst case, there is a single error in 100 samples, providing an estimated value of 99% of a posteriori accuracy, that is, 99% of the found correlations are correct at those operation points. This result suggests to using a value of FOD lower than 2 h. As will be noted in the next subsection, the greater the value of FOD, the greater the percentage of correlated elements. Therefore, we select 1 h for FOD.

To confirm the validity of these results, we complemented the inspection with another check. For this, we evaluated various indicators for the timings of the incidents, as shown in Table 3. In this regard, as explained in Section 5, one of the most conflictive cases for the correlation was related to consecutive incidents affecting the same network elements. To be more confident about the results and have insight into this particular case, we can consider the delay between the first appeared alert for each incident and the closing time of the previous ticket, if any, including any of the affected elements in the incident, that is, the interval to the previous potentially related ticket. As shown in Table 3, the mean value for this magnitude is 33.3 h., which is significantly greater than the selected value for FOD. This can be interpreted as a clear indication that the first appeared alert is related to the current incident and not to a previous one for the selected FOD value of 1 h.

Table 3
Mean values for some variables for the correlated subset (FOD = 1 h.).

Ticket lifetime	58.8 h
Incident lifetime	81.4 h
Number of rep. alerts/ sample	5.5
Number of alerts/rep. alert	4.8
Interval to previous potentially related ticket	33.3 h

6.2.2. Joint correlation results

Another relevant parameter to evaluate the performance of the system is the recall, defined as $recall = TP/(TP + FN)$, where TP is the number of events that are correctly correlated to any representative event and FN is the number of events that are incorrectly kept uncorrelated to any representative event. Under the assumption that every relevant incident should generate at least one ticket and that all the relevant alerts should be associated to an incident, FN is equal to the number of alerts not being correlated to a representative event.

However, similar to the case of the estimation of the precision, both FOD and BOD may play an important role in the percentage of correlated elements. Simply stated, if FOD (or BOD) is sufficiently large, all of the alerts and tickets related to a common element may end up correlated, thus providing an artificially high value for the recall.

To evaluate this effect, a series of experiments were carried out by varying the values of FOD and BOD. It is worth mentioning that these experiments were made in parallel with those described in the previous section, all of them mainly targeted at tuning the system. The percentage of alerts correlated to incidents as a function of the values of FOD and BOD is presented in Fig. 11. As shown and as expected, as FOD or BOD increases, the percentage of correlated alerts does as well, which would imply that the greater FOD/BOD is, the better the correlation is. However, this might be an erroneous conclusion, as large values for FOD/BOD would merge together independent incidents involving some common affected element. Alternately, having long delays is not reasonable in ticket creation (FOD) for critical incidents. Regardless, the figure shows that there is no relevant influence from the value of FOD in the results for FOD below 64 h. This was somehow expected because the delay in the closing of the last ticket should be associated with a lack of related alerts, not the presence of them. The behavior in relation to BOD is similar.

These results, together with the analysis of the tickets/alerts that resulted in inappropriate merges (correlation errors), as described in the previous subsection, made us select FOD = BOD = 1 h.

The correlation results taking into account only the tickets created by MS are listed in Table 4. As shown, approximately 80% of the representative tickets and 46% of the representative alerts potentially related to relevant incidents are correlated. Furthermore, if we consider the initial non-correlated tickets and alerts, up to 70.3% of the alerts and 84.5% of the tickets are correlated. For the tickets, this means that 84.5% of them are truly related to relevant incidents (there exist associated critical alerts), which provides no further information on the quality of the correlation itself, as the remaining 15.5% could be related to the critical nodes they refer to but not to a critical episode. In fact, we ended up with 40 tickets not correlated to any alert, which deserves a

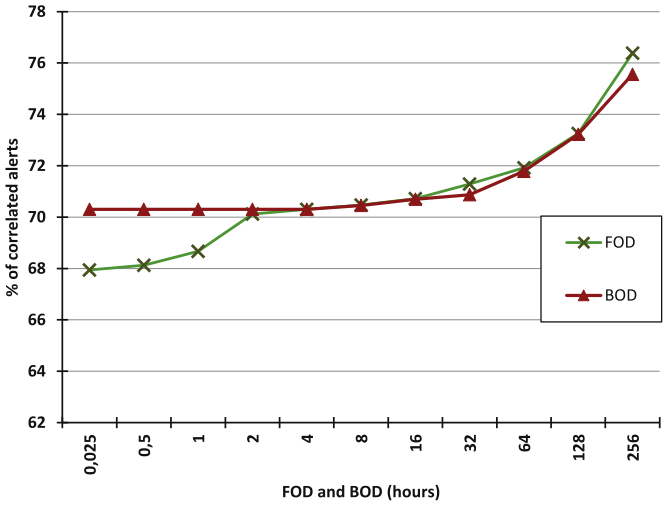

Fig. 11. Percentage of correlated alerts at different values for FOD and BOD.

Table 4
Correlation results considering only tickets created by the MS group.

	Alerts	Tickets
Number of raw input elements	7436	348
Number of representatives (before joint correlation)	3022	194
Number of representatives correlated by joint correlation	1391 (46.0%)	154 (79.4%)
Number of raw elements correlated (after joint correlation)	5228 (70.3%)	294 (84.5%)

posterior analysis. Alternately, having only 46.0% of the relevant representative alerts be correlated to a ticket is, at first sight, not a very good result, although it represents an advance that no other similar system has achieved. Therefore, this result requires deeper analysis to find the potential causes for such a figure, which is addressed next.

6.2.3. Analysis of the non-correlated alerts

Although the effectiveness of the proposed technique in terms of improperly correlated events has been shown to be high –question (1)–, approximately half of the representative relevant alerts are not assigned to a ticket –question (2)–, which requires further analysis. According to the protocols in use at the company, all of the considered alerts should have generated tickets from MS. This incoherency can be initially attributed to the fact that the proposed method is not accurate enough. However, an improper application of that policy or some problems with the staff could also explain it. Therefore, we analyzed those non-correlated alerts in search of some explanation for them not triggering tickets. For this we considered two potentially influential factors: the alert durations and inter-arrival delays of alerts having the same affected element.

The analysis of the duration of the non-correlated alerts provided the results shown in Fig. 12. As a first conclusion, we observed that a significant percentage of them present a short duration, which also implies that they are shown as active in the management console for a short time. This finding motivated us to further analyze those alerts. With the help of the MS, we reached to the conclusion that a short duration alert can be considered irrelevant and that the dedication of the staff to other tasks can also hide them, thus not generating tickets from MS. In fact, up to 64.3% of the uncorrelated alerts can be attributed to this effect if we consider a threshold of 10 mins for their durations, which seems reasonable for the MS.

To continue with the analysis, for alerts that have durations greater than 10 mins, we analyzed the inter-arrival delay between consecutive alerts having the same affected element to see if they are created close in time or if there is a time gap between them. Fig. 13 shows the histogram of the inter-arrival delays. We found that more than 80% of them were repeated within a period greater than 2 days, that is, most of them appear and, despite its long or short duration, no additional alert related to the same affected element appears in at least two days. This means that the alert is scaling down in the list of active alerts on the management console. Therefore, we conclude that the lack of an associated ticket can be possibly attributed to the existence of a "window of opportunity" for the creation of the tickets. Thus, if an alert does not trigger a ticket within a given period and it is not repeated, it is likely it will not trigger a ticket at all. After consultation, MS confirmed this observation.

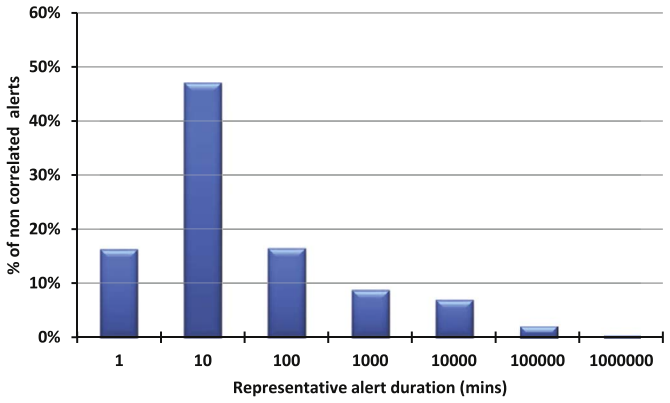

Fig. 12. Histogram for the duration of the non-correlated alerts.

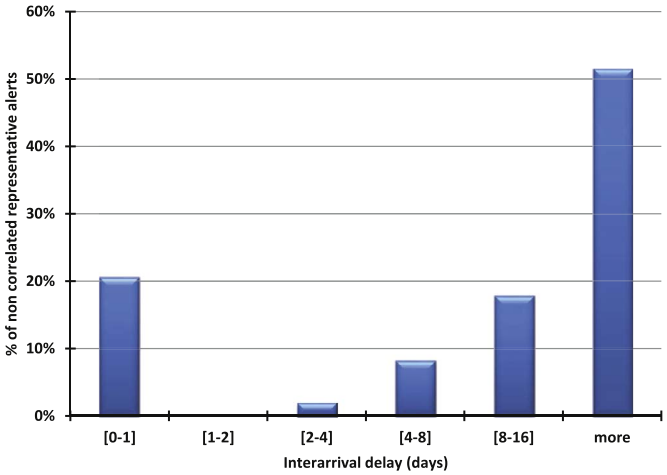

Fig. 13. Interarrival delay between consecutive non correlated alerts having the same affected element.

Additionally, and besides the above conclusion for the last subset, we found that approximately 68.8% of the non-correlated alerts included names for affected elements not conforming to the naming conventions in use. After consulting with the staff, we were informed that these types of elements, despite being classified as critical ones, have special functions that are not being used by other nodes. Thus, the MS do not usually open tickets for those types of affected elements. That is, we were initially provided with an inaccurate list of critical nodes.

As a resume, Fig. 14 summarizes the results from the assessment of both correlated and non-correlated alerts. It is worth mentioning that if we accept that alerts lasting less than 10 mins are prone to being ignored, only 10.6% of the initial alerts remain uncorrelated due to unknown reason. Considering those with names conforming to the critical nodes list, the percentage of non-correlated critical alerts is considerably reduced to 7.3% with the proposed method.

6.2.4. Analysis of the non-correlated tickets

As shown in Table 4, there exist 40 representative tickets that are not correlated to any representative alert after applying the joint correlation. This is an unexpected result, as it is supposed that all these tickets are opened by technical staff as a response to abnormal events in critical nodes, which implies the appearance of critical alerts. Similar to the alerts, these tickets have been found to have some insights into the potential causes for them not being correlated. In this analysis, the

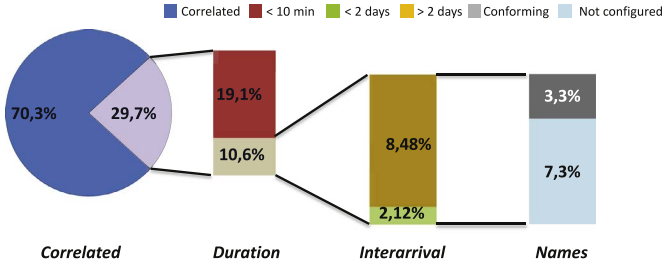

Fig. 14. Distribution of the number of raw alerts among the different assessment criteria.

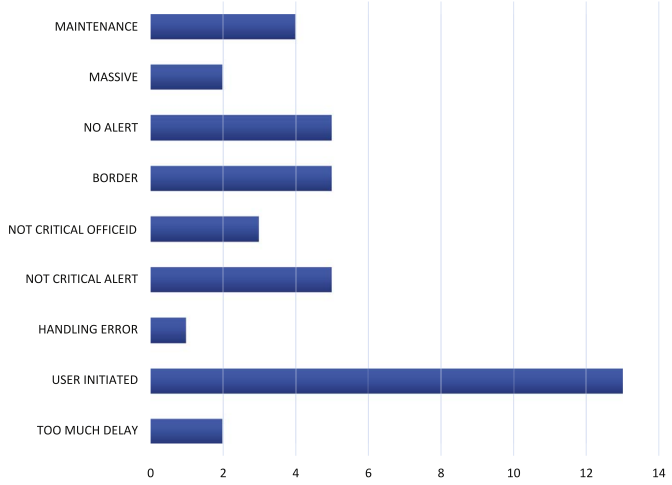

Fig. 15. Results from the analysis of the uncorrelated tickets.

original dataset of alerts, as provided by the corporation, is also considered to check for any potential problem during the preprocessing and later phases of the procedures. Nevertheless, as in the previous cases, manual inspection is not trivial because the available information from tickets and alerts can be inconclusive or even incomplete.

The manual analysis of these 40 tickets provided up to 9 different potential causes (Fig. 15):

1. Maintenance. These tickets are generated during maintenance operations in some elements of the network, both programmed and unprogrammed. No associated alerts are found in the alert dataset, probably because they are filtered out during the maintenance procedures.
2. Massive. These tickets are related to massive failures in ADSL connections at critical *officeIDs* although no associated MASSIVE alert is present in the alert dataset. The free text fields in the ticket refer to NodeIDs that are not critical, and thus they are not included in the alert dataset.
3. No alert. No associated alert is found in the alert dataset for the ticket duration or its vicinity, despite the NodeIDs being critical.
4. Border. Tickets generated at the beginning of the observation period (first 2 days) with no associated alert in the dataset. Some of them even refer to alerts and previous tickets outside of the observation period.
5. Non-critical *officeID*. The value for the *officeID* refers to one of the non-conforming names, so it should not be considered a critical *officeID*.
6. Non-critical alert. There exists at least one alert related to this ticket, but it is not a critical reason. Thus, the alerts were removed during preprocessing.
7. Handling error. The NodeID for this ticket is wrong. The incident is not truly related to that NodeID.
8. User-initiated. Although these tickets come from MS, some of them are generated as a response to phone calls from the technicians at different locations. No alerts are observed, and most of them are related to internal problems at the location, e.g., a local email server not responding.
9. Too much delay. There exist alerts potentially related to these tickets, but they are placed outside the considered FOD or BOD.

As a result, we can conclude that 28 of these tickets (user-initiated, non-critical alert, non-critical *officeID*, maintenance, handling error and massive) should have been filtered out because they are not opened as a response to the observation of critical alerts by the MS. Alternately, border tickets can obviously be attributed to an experimental limitation related to the observation period. Furthermore, the lack of correlation for the no alert class cannot be attributed to the correlation method, as no related alerts are found. This could be due to a problem with the acquisition of the alert dataset. Finally, the 2 tickets in the too much delay class are clearly related to the limitations of the proposed method to handle events when the creation of the tickets is not responsive enough.

In summary, from the set of 348 initial tickets, 2 are not properly correlated by the proposed method due to its limitations, 38 are not correlated due to the lack of related information in the alert dataset due to different reasons, and 308 are correctly correlated.

7. Applications

Finally, we discuss and evaluate two possible applications of the proposed method. Two major contributions can be identified: improvement of alert correlation, and provision of some insights into evaluating the efficiency of the management team handling network incidents.

7.1. Alert reduction

The results regarding the capabilities of the ticket-alert correlation provided in Section 6.2 are a clear indicator of the potentialities of the proposed method. As shown in Table 4, there is a large reduction in the number of alert representatives, that is, in the number of different alerts after alert clustering when including the information from the tickets. In fact, 1391 of the initial 3022 representative alerts after alert correlation are associated with 194 incidents; that is, those 1391 representative alerts are merged into 194 representative ones through the joint correlation, with an average of 7.17 representative alerts per incident. Therefore, the final alert set to consider contains only 1825 alerts (those representative alerts not correlated by joint correlation plus the 194 newly correlated representatives), that is, half of the original correlated set and 1/4 of the original number of alerts. Furthermore, the analysis of the alerts that could not be correlated evidences a low confidence in their relevance.

These results confirm our intuition regarding related alerts not overlapped in time (Fig. 3) and the inclusion of additional relationships created by tickets. Therefore, we conclude from this interesting finding that incorporating ticket information into the alert correlation process will definitely help in reducing a higher percentage of related alerts.

Nevertheless, it is important to note that we have not yet used the full potentiality of the system, as the tickets from SD have not been used during the assessment of the method. The information in these tickets can be far more significant than that in the MS tickets because they incorporate the end users' perceptions of the incident.

Alternately, if we consider the tickets related to both MS and SD, we might check whether SD systems play an important role in the incident-solving process by applying the correlation algorithm to tickets created by both the MS and SD groups.

The results, shown in Table 5, evidence that the proposed correlation model is able to correlate tickets from SD at a similar percentage as that for tickets from MS and that the inclusion of these tickets improves the results. In particular, there is a 6.8% increase in the percentage of raw correlated alerts, and more importantly, the number of correlated

Table 5
Correlation results for relevant tickets created by both the MS and SD groups.

	Alerts	Tickets
Number of raw input elements	7436	520
Number of representatives (before joint correlation)	3022	256
Number of raw elements correlated (after joint correlation)	5734 (77.1%)	436 (83.8%)
Number of representatives correlated	1683 (55.7%)	189 (73.8%)

incidents rises from 154 to 189. This means that SD is not only creating redundant tickets, as would be a priori expected, but also generating tickets for incidents not acknowledged by the technical staff.

Therefore, we can conclude that: i) SD systems are meaningful to assist in the incident-solving problem and are not just a call center for handling customer calls and complaints; and ii) the proposed method is able to incorporate relevant information, which is not available from any other source, into the correlation process, thus improving the quality of the results and reducing the number of elements in the output.

7.2. Measuring staff efficiency

A second good candidate application for our system is in providing some insight into how to assist with and evaluate the efficiency of the management team in the incident resolution process. The proposed system might help decision-makers in answering several questions related to the quality of the management, such as the following: How fast/accurate is the staff? Do all the working shifts and management groups behave the same way?

As an example, consider the case in which the operator is interested in a measure of the reaction time of the management team, a group of persons or even an individual member of the staff in dealing with incidents; e.g., we need to measure how much time the management team needs to open a ticket for an ongoing incident. We can measure the delay between the first appeared alert and the first ticket creation time of an incident. In this case, we obtained an average value of 1.27 h. However, if we need to measure how much time the management team needs to close a ticket for an already resolved incident, we can measure the delay between the first resolved alert and the last resolved ticket related to an incident. In our case, the mean value for this magnitude is 132.3 h, which is certainly a large value. Similarly, other measures from the model can be used and interpreted.

Another example is related to the assessment of the performance of the working shifts. Because the number of persons in charge of the management uses is not the same for all shifts and the workload is usually different, the analysis of the correlated alerts and tickets for the different working shifts can reveal relevant information. In particular, the analyzed company considers three working shifts in a day: the morning shift (MorS), from 7:00 AM to 15:00 PM; the afternoon shift (AS), from 15:00 PM to 23:00 PM; and the night shift (NS), from 23:00 PM to 7:00 AM. Furthermore, the characteristics of the working shifts change during weekends or holidays. The analysis of non-correlated relevant alerts as a function of the working shift is summarized in Table 6. As shown, there exist differences in the working shift regarding the distributions of incidents (alerts) and the percentage of non-correlated alerts. During MorS and AS shifts, the percentage of non-correlated alerts is not in consonance with the percentage of existing alerts. In fact, it seems that AS is less responsive to alerts than MorS and NS, which can imply a shortage in personnel. The opposite occurs for RS.

8. Conclusions and future work

Our main contribution in this article is to show that leveraging the information provided by incident tickets is relevant to increase the efficiency of the usual incident management process in a corporate network. To achieve this target, we have proposed a methodology to incorporate incident-related semantic information, coming in the form of tickets created by users and management staff, into an alert database that contains incident-related information from a network perspective. Adding human knowledge and relevance into the process enhances the quality of the discovery of incidents. Also, our findings showed that incorporating such types of information in the alert correlation process increase the alert reduction rate, and consequently speed up the diagnosing process. Furthermore, this rate is increased even more when considering tickets created by Service Desk systems. At the same time, the proposed methodology is based on simple elements and reasoning, making its application in a real NMS, by both management staff and decision makers, almost straightforward. Finally, we conclude that any new good solution for the alert correlation problem should consider such kind of expert information in its design.

Table 6
Distribution of the number of alerts over working shifts.

working shifts	% of relevant alerts	% of non-correlated alerts
MorS	50.6	33.6
AS	16.9	29.9
NS	32.5	36.5

8.1. Limitations

Although the lightweight approach proposed here can relate alerts and tickets together and hypothesize about possible relationships between them, it is limited in several ways. First, the main assumption behind this approach is that it correlates the timely overlapped tickets and/or alerts and does not cover the non-overlapped sets beyond the time thresholds, BOD and FOD. Besides, the approach treats any two simultaneous different incidents that affect a common resource as a single one and thus a single representative event will be generated. Second, the algorithm used for the initial alert correlation is really simple and also based in temporary relationships and node identity similarities. This is by choice, as the focus was on demonstrating that tickets can help to improve the correlation. The proposed method can be easily adapted to consider any of the available alert correlation methods for this module, and even for the ticket correlation module, with the only limitation being that timestamps are required at the output. Third, the similarity and filtering criteria used for alerts and tickets depend on various rules and a list of human-provided keywords that may depend on the considered network.

8.2. Future work

Once the relevance of including the tickets in the correlation procedures is shown, the next steps should be targeted at improving the efficiency of the entire system. For this, as noted in the previous paragraph, two major issues can be addressed: improving the elementary correlation modules by using state-of-the-art methods and including new sources of information, e.g., topological, that complement the similarity function to set relationships between the events.

Acknowledgments

This work has been partially supported by Spanish MICINN through project TIN2014-60346-R.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at 10.1016/j.inffus.2018.01.011.

References

[1] E. Marilly, O. Martinot, H. Papini, D. Goderis, Service level agreements: a main challenge for next generation networks, Proceedings of the 2nd European Conference on Universal Multiservice Networks (ECUMN), (2002), pp. 297–304.
[2] The office of government commerce (OGC). IT infrastructure library (ITIL), 2016 Available at http://www.itil-officialsite.com/. (accessed: 11.12.2016).
[3] The Office of Government Commerce (OGC). Service Operation, IT Infrastructure Library version 3 (ITIL v3), Technical report, The Stationary Office, 2007. (accessed 11.12.2016).
[4] G. Jakobson, M.D. Weissmann, Alarm correlation, IEEE Netw. 7 (6) (1993) 52–59.
[5] F. Valeur, G. Vigna, C. Kruegel, R.A. Kemmerer, A comprehensive approach to

intrusion detection alert correlation, IEEE Trans. Dependable Secure Comput. 1 (3) (2004) 146–169.
[6] J. Hu, H. Chen, T. Liu, H. Tseng, D. Lin, C. Yang, C.E. Yeh, Implementation of alarm correlation system for hybrid networks based upon the perfSONAR Framework, Proceedings of the International Conference on Advanced Information Networking and Applications Workshops (WAINA'10), (2010), pp. 893–898.
[7] D.S. Kim, H. Shinbo, H. Yokota, An alarm correlation algorithm for network management based on root cause analysis, Proceedings of the 13th International Conference on Advanced Communication Technology (ICACT'11), (2011), pp. 1233–1238.
[8] L. Lewis, G. Dreo, Extending trouble ticket systems to fault diagnostics, IEEE Netw. 7 (6) (1993) 44–51.
[9] R. Costa, N. Cachulo, P. Cortez, An intelligent alarm management system for large-scale telecommunication companies, Proceedings of the 14th Portuguese Conference on Artificial Intelligence (EPIA'09), (2009), pp. 386–399.
[10] V. Holub, T. Parsons, P. O'Sullivan, J. Murphy, Run-time correlation engine for system monitoring and testing, Proceedings of the 6th IEEE International Conference on Autonomic Computing (ICAC-INDST '09), (2009), pp. 43–44.
[11] S. Klinger, S. Yemini, Y. Yemini, D. Ohsie, S. Stolfo, A coding approach to event correlation, Proceedings of the 4th International Symposium on Integrated Network Management, (1996), pp. 266–277.
[12] A. Valdes, K. Skinner, Probabilistic alert correlation, Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection (RAID), (2001), pp. 54–68.
[13] S. Dadkhah, M.R. KhaliliShoja, H. Taheri, Alert correlation through a multi components architecture. Int. J. Electr. Comput. Eng. 3 (4) (2013) 46–466.
[14] M. Bateni, A. Baraani, Time window management for alert correlation using context information and classification, Int. J. Comput. Netw. Inf. Secur. (IJCNIS) 5 (11) (2013) 9–16.
[15] J. Yu, Y.V. Ramana Reddy, S. Selliah, S. Kankanahalli, S. Reddy, V. Bharadwaj, TRINETR: an intrusion detection alert management systems, Proceedings of the 13th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), (2004), pp. 235–240.
[16] J.H. Bellec, M.T. Kechadi, Towards a formal model for the network alarm correlation problem, Proceedings of the 6th International Conference on Simulation, Modeling and Optimization (SMO'06), (2006), pp. 458–463.
[17] S. Salah, G. Maciá-Fernández, J.E. Díaz-Verdejo, A model-based survey of alert correlation techniques, Comput. Netw. 57 (5) (2013) 1289–1317.
[18] S.A. Mirheidari, S. Arshad, R. Jalili, Alert correlation algorithms: a survey and taxonomy, Lect. Notes Comput. Sci. 8300 (2013) 183–197.
[19] T. Chyssler, S. Nadjm-Tehrani, S. Burschka, K. Burbeck, Alarm reduction and correlation in defense of IP networks, Proceedings of the13th IEEE Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), (2004), pp. 229–234.
[20] P. Calyam, M. Dhanapalan, M. Sridharan, A. Krishnamurthy, R. Ramnath, Topology-aware correlated network anomaly event detection and diagnosis, J. Netw. Syst. Manag. 22 (2) (2014) 208–234.
[21] H. Yan, L. Breslau, Z. Ge, D. Massey, D. Pei, J. Yates, G-RCA: a generic root cause analysis platform for service quality management in large IP networks, IEEE/ACM Trans. Netw. 20 (6) (2012) 1734–1747.
[22] G. Jakobson, W.M. D., Real-time telecommunication network management: extending event correlation with temporal constraints, Proceedings of the 4th International Symposium on Integrated Network Management (IEEE/IFIP), (1995), pp. 290–301.
[23] S.H. Ahmadinejad, S. Jalili, Alert correlation using correlation probability estimation and time windows, Proceedings of the International Conference on Computer Technology and Development (ICCTD'09), 2 (2009), pp. 170–175.
[24] K. Julisch, Clustering intrusion detection alarms to support root cause analysis. ACM Trans. Inf. Syst. Secur. 6 (4) (2003) 443–471.
[25] M. Xiao, Y. Yang, Z. Du, Ontology based alarm correlation technology in TD-SCDMA network, J. Comput. Inf. Syst. 9 (3) (2013) 933–940.
[26] Y. Chen, J. Lee, Autonomous mining for alarm correlation patterns based on time-shift similarity clustering in manufacturing system, Proceedings of the International Conference on Prognostics and Health Management (PHM'11), (2011), pp. 1–8.
[27] K. Lee, J. Kim, K. Kwon, Y. Han, S. Kim, DDOs attack detection method using cluster analysis, J. Expert Syst. Appl. 34 (3) (2008) 1659–1665.
[28] H.O. Mynit, P. Meesad, Incremental learning algorithm based on support vector machine with mahalanobis distance (ISVMM) for intrusion prevention, Proceedings of the 2nd International Conference on Intelligent Computation Technology and Automation, 2 (2009), pp. 25–28.
[29] M. Siraj, M.A. Marrof, S.Z.M. Hashim, Intelligent alert clustering model for network intrusion analysis. Int. J. Adv. Soft Comput. Appl. 1 (2009) 33–48.
[30] T. Li, X. Li, Novel alarm correlation analysis system based on association rules mining in telecommunication networks, Inf. Sci. (Ny) 180 (16) (2010) 2960–2978.
[31] W. Jian, L.X. Ming, A dynamic mining algorithm of association rules for alarm correlation in communication networks, Proceedings of the 3rd IEEE/Create-Net International Conference on Communication System Software and Middleware (COMSWARE'08), (2008), pp. 799–802.
[32] J. Wu, X. Li, Communication network alarm correlation based on multi-dimensional fuzzy association rules mining, Proceedings of the 2nd International Conference on Electric Information and Control Engineering (ICEICE'12), 1 (2012), pp. 439–443.
[33] R. Agrawal, R. Srikant, Fast algorithms for mining association rules, Proceedings of the 20th International Conference on Very Large Databases, (1994), pp. 487–499.
[34] R. Sadoddin, A. Ghorbani, Real-time alert correlation using stream data mining techniques, Proceedings of the 20th International Conference on Innovative Applications of Artificial Intelligence, 3 (2008), pp. 1731–1737.
[35] K. Yamanishi, Y. Maruyama, Dynamic syslog mining for network failure monitoring, Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, (2005), pp. 499–508.
[36] H. Sizu, Z. Xianfei, Alarms association rules based on sequential pattern mining algorithm, Proceedings of the 5th International Conference on Fuzzy Systems and Knowledge Discovery, (2008), pp. 556–560.
[37] G. Dreo, V. Robert, Using master tickets as a storage for problem solving expertise, Proceedings of the 4th IFIP/IEEE International Symposium on Integrated Network Management IV, (1995), pp. 328–340.
[38] D. Johnson, NOC Internal integrated trouble ticket system, Funct. Specif. Wishlist (1992). RFC 1297.
[39] A. Medem, R. Teixeira, N. Feamster, M. Meulle, Determining the causes of intradomain routing changes, Technical report, UMIACS University (2009).
[40] N. Feamster, H. Balakrishnan, Detecting BGP configuration faults with static analysis, Proceedings of the 2nd International Conference on Symposium on Networked Systems Design and Implementation (NSDI), 2 (2005), pp. 43–56.
[41] D. Turner, K. Levchenko, J.C. Mogul, S. Savage, A.C. Snoeren, On failure in managed enterprise networks, HP Labs (2012). HPL-2012-101.
[42] L. Tang, T. Li, L. Shwartz, F. Pinel, G. Grabarnik, An integrated framework for optimizing automatic monitoring systems in large IT infrastructures, Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, (2013), pp. 1249–1257.
[43] A.F. AlEroud, G. Karabatis, Queryable semantics to detect cyber-attacks: a flow-based detection approach, IEEE Trans. Syst. Man Cybern.: Syst. 99 (2016) 1–17.
[44] A. Medem, R. Teixeira, N. Feamster, M. Meulle, Joint analysis of network incidents and intradomain routing changes, Proceedings of the International Conference on Network and Service Management (CNSM), (2010), pp. 198–205.
[45] R. Potharaju, N. Jain, C. Nita-Rotaru, Juggling the jigsaw: towards automated problem inference from network trouble tickets, Proceedings of the 10th USENIX Symposium on Networked Systems Design and Implemenentation, (2013), pp. 127–141.
[46] Logstash., 2016, Available at https://www.elastic.co/products/logstash. (accessed: 11.12.2016).
[47] Splunk., 2016,. Available at https://www.splunk.com/. (accessed: 11.12.2016).
[48] L. Sumo, 2016, Available at https://www.sumologic.com/. (accessed: 11.12.2016).
[49] G. Maciá-Fernández, J.E. Díaz-Verdejo, P. García-Teodoro, Mathematical model for low-rate dos attacks against application servers, IEEE Trans. Inf. Forensics Secur. 4.3 (2009) 519–529.
[50] G. Maciá-Fernández, J.E. Díaz-Verdejo, G.-T. Pedro, Evaluation of a low-rate dos attack against application servers, Comput. Secur. 27.7 (2008) 335–354.
[51] F.N. Patel, N.R. Soni, Text mining: a brief survey, Int. J. Adv. Comput. Res. 2 (4) (2012) 243–248.
[52] S. Salah, G. Maciá-Fernández, J.E. Díaz-Verdejo, L. Sánchez Casado, A model for incident tickets correlation in network management, J. Netw. Syst. Manag. 1 (24) (2016) 57–91.