IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 8, 2021, accepted January 20, 2021, date of publication February 1, 2021, date of current version February 11, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3056149

READ-IoT: Reliable Event and Anomaly Detection Framework for the Internet of Things

AYMEN YAHYAOUI[1,2], TAKOUA ABDELLATIF[1,4], SAMI YANGUI[3], AND RABAH ATTIA[1]
[1]SERCOM Laboratory, University of Carthage, Carthage 1054, Tunisia
[2]Military Academy of Fondouk Jedid, Nabeul 8012, Tunisia
[3]LAAS-CNRS, Université de Toulouse, INSA, 31400 Toulouse, France
[4]ENISo, University of Sousse, Sousse 4002, Tunisia

Corresponding author: Aymen Yahyaoui (aymen.yahyaoui@ept.rnu.tn)

This work was supported in part by the National Project (PACTE-PISTE) to Fight Terrorism in Tunisia. The project main goal is to design a reliable surveillance system for the monitoring of hostile environments for the Tunisian Army.

ABSTRACT Internet of Things (IoT) enables a myriad of applications by interconnecting software to physical objects. The objects range from wireless sensors to robots and include surveillance cameras. The applications are often critical (e.g. physical intrusion detection, fire fighting) and latency-sensitive. On the one hand, such applications rely on specific protocols (e.g. MQTT, COAP) and the network to communicate with the objects under very tight timeframe. On the other hand, anomalies (e.g. communication noise, sensors' failures, security attacks) are likely to occur in open IoT systems and can result by sending false alerts or the failure to properly detect critical events. To address that, IoT systems have to be equipped with anomaly detection processing in addition to the required event detection capability. This is a key feature that enables reliability and efficiency in IoT. However, anomaly detection systems can be themselves object of failures and attacks, and then can easily fall short to accomplish their mission. This paper introduces a Reliable Event and Anomaly Detection Framework for the Internet of Things (READ-IoT for short). The designed framework integrates events and anomalies detection into a single and common system that centralizes the management of both concepts. To enforce its reliability, the system relies on a reputation-aware provisioning of detection capabilities that takes into account the vulnerability of the deployment hosts. As for validation, READ-IoT was implemented and evaluated using two real life applications, i.e. a fire detection and an unauthorized person detection applications. Several scenarios of anomalies and events were conducted using NSL-KDD public dataset, as well as, generated data to simulate routing attacks. The obtained results and performance measurements show the efficiency of READ-IoT in terms of event detection accuracy and real-time processing.

INDEX TERMS Anomaly detection, cloud computing, event detection, fog computing, Internet of Things, intrusion detection, trust, reputation.

I. INTRODUCTION

Internet of Things (IoT) refers to the ubiquitous network of heterogeneous objects such as cameras, sensors and drones [1]. These objects are able to interact and to communicate with each other while relying on Internet protocols or other protocols addressing schemes [2] (e.g. IPv4, IPv6, IEEE 802.15.4, ZigBee, LoRaWAN) and specific messaging protocols (e.g. COAP, MQTT). The ultimate goal is to implement a common goal that would make up the so-called IoT applications (e.g. weather forecast, HVAC-Heating, Ventilation and Air-Conditioning, access control). The applications domains of IoT technologies are multiple. Precision agriculture, smart transportation and healthcare are among the several examples. IoT is adjustable to almost any technology capable of providing relevant information such as about the performance of an activity or about the related environmental conditions.

The associate editor coordinating the review of this manuscript and approving it for publication was Biju Issac.

A. CONTEXT AND MOTIVATIONS

IoT systems are often critical and latency-sensitive [3]. The information needs to travel fast from the objects to the software through the gateways and the network. Furthermore, these systems have to be efficient and reliable to manage handling critical tasks. For instance, for natural disasters

management, IoT systems must detect interesting events which are related to the purpose for which the IoT infrastructure was designed (e.g. fire, floods, gas leakage) as soon as possible and make sure not to miss any of them due to prospective anomalies. Anomalies can be classified into two categories: (i) Anomalies related to technical or inner system issues (e.g. communication failure or noise, object failure or malfunction), and (ii) anomalies related to the system integrity and security (e.g. malicious attacks, intruder objects). At run-time, IoT systems may face one or both anomalies at the same time. In this particular case, the system accuracy can be easily compromised, and this could cause damaging effects, important material loss, information theft and even injuries or people death. Therefore, it is critical to detect these anomalies, process them as soon as possible and maintain the system in good condition all the time. For instance, when technical issues occur, the system should be able to detect them and recover in an autonomous fashion (e.g. select and switch to another cluster head if the current one breaks down). Similarly, in case of security threats, the system should be able to detect them while minimizing the damage (e.g. isolate and disconnect a malicious node broadcasting erroneous data). Accordingly, it is critical to provide efficient surveillance systems that are able to detect such events and anomalies on real-time, raise alerts in a minimum delay and implement appropriate actions to improve the system reliability and then the detection accuracy (e.g. fire for fire detection applications, intrusion for monitoring systems). Interesting events and anomalies are generally considered in the literature as outliers [4].

In the relevant literature, several work already proposed to integrate Anomaly Detection Systems (ADS) in IoT (e.g. see [5]–[8]). However, ADS are generally introduced as standalone systems with their own management and data storage. This induces many efforts to connect the ADS to the IoT system compared to built-in ADS system. Furthermore, ADS can be themselves subject to failures and attacks, and then fall short to accomplish their mission. Reliability tools adopted to supervise IoT sensors and networks should be reused. It is worth mentioning that the reliability overhead must be minimal since the considered context (i.e. IoT) is often limited in terms of computing and energy resources [1].

B. CONTRIBUTIONS AND RESULTS

This paper introduces a Reliable Event and Anomaly Detection Framework for the Internet of Things (READ-IoT for short). This framework enables accurate detection of outliers thanks to the adoption of a common management system and an optimal processing workflow. The proposed approach relies on cascaded (2 steps) activation of detection components. First, a rule-based static detection is considered. Then, based on the output of the first detection, machine learning capabilities are used to complement the detection process. The cascaded detection allows to characterize the detected outliers. Specifically, known outliers are detected thanks to a rule-based detection, and then machine learning processing permits further investigation and characterization of unknown outliers.

READ-IoT framework proposes an integrated solution for ADS and EDS. There are at least four advantages to this integration:

- Building a common and unique management system: the same provisioning strategies and mechanisms used for both subsystems ADS and EDS provides administrators with unified and practical procedures. Obviously, this considerably decreases the operating cost and complexity of IoT systems. Basically, the detection components are deployed following a common optimized deployment plan aiming at reducing latency and selecting trusted deployment devices. Other common management components are QoS management and risk calculation that are shared between ADS and EDS.
- Improving the cooperation between subsystems: since they are integrated, the two subsystems can interact in real-time without building third-party bridges between them. When the ADS detects anomalies from a device or a network, the EDS has to be informed in real-time to ignore non-trusted sources of information. The built-in management system, the unified processing workflow and the common communication infrastructure eliminate the need for a 'system integrator' between the subsystems.
- Improving the data accessibility: by being able to read data from each subsystem (ADS and EDS) in a centralized way, the data is seen as one-piece. This is crucial for efficient decision making. QoS data is gathered, and IoT sensors and networks trust evaluation is build. Having related data from both systems in a centralized way allows for drawing a more complete and accurate view of the observed system compared to separate observations.
- Improving the system reliability: data integration fed from both EDS and ADS allows for building a common risk management. Untrusted data sources are discarded from the deployment plan, and the system reliability is then immediately improved.

To address real-time constraints, READ-IoT is designed in the hybrid cloud/fog system. It relies on adaptive resource-aware deployment that takes into account the resources type (i.e. cloud or fog), location with regard to the IoT application (i.e. in core network, or at the edge, close to the data sources), as well as, workload over these resources (i.e. available memory, processing capabilities and network bandwidth). For validation and evaluation purposes, the READ-IoT framework was implemented, and several real life IoT applications were provisioned over it. The performed experiments show that: (1) combining both event and anomaly detection in IoT systems improves the system reliability, (2) the cascaded activation of rule-based and machine learning processing qualifies better the detected outliers, (3) the reputation-aware deployment enforces ADS and EDS reliability and then the detection accuracy and (4) the resource-aware provisioning

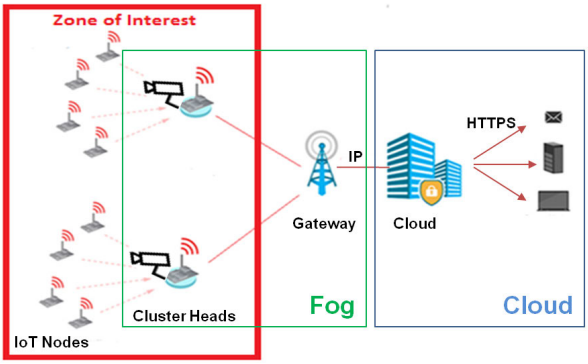


FIGURE 1. IoT system reference topology.

over hybrid cloud/fog environments enables addressing real-time constraints and reduces reliability overhead.

C. STRUCTURE OF THE MANUSCRIPT

Section II introduces background information. Section III reviews the related work in the literature. Section IV describes the READ-IoT framework architecture. Section V discusses the designed algorithms and models for ADS, EDS and placement. Section VI shows the developed prototype architecture and tools. Section VII details the evaluation experiments scenario and results. Finally, Section VIII concludes the paper and presents the work perspectives.

II. BACKGROUND INFORMATION

This Section introduces fundamental and background information that are necessary for the understanding of this work.

A. IoT FOR SURVEILLANCE SYSTEMS

IoT applications assist enterprises and organizations to realize the potential of the Internet of Things. Businesses are relying on IoT applications to optimize workflows and enhance the control of operations at each step of the supply chain [10]. In the particular case of surveillance systems, IoT turns security surveillance into smart safety and security management [11]. In the near past, Closed-Circuit Television Systems (CCTV) were the most common used technology in surveillance. However, such systems can only display and record video footage. They do not understand what they are watching and are not able to do anything about it. Nowadays, the IoT surveillance systems are able to automatically detect threats (e.g. smoke, intrusion) and make the right calls to process them. They rely on machine learning and computer vision capabilities [12]. IoT surveillance applications consume smart devices data traveling from the edge of the network, execute appropriate programs, and produce added values services (e.g. visualization, prediction) and/or suitable actions for actuators such as robots and drones.

Fig. 1 depicts a classical IoT system architectural overview [13]. The same is valid for surveillance applications. It consists of the following entities:

- IoT nodes (e.g. Sensors and Actuators): Sensor Nodes (SN) are responsible for gathering and collecting data such as temperature, humidity, gas and motion. Actuator Nodes (AN) as fire extinguishers are IoT devices that may be used to react to detected events.
- Cluster Heads (CH): responsible for receiving and forwarding data from a given set of IoT nodes (cluster) to IoT gateways. They can be involved in monitoring and analyzing tasks. Furthermore, they may be equipped with cameras to survey the whole cluster zone.
- IoT Gateways: responsible for making the bridge to cloud servers and might be involved in analysis task.
- Cloud: responsible of data analysis and storage.

B. CLOUD AND FOG COMPUTING

The American National Institute of Standards and Technology (NIST) defines cloud computing as a novel model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g. networks, servers, storage resources, applications, services, etc.) [14]. These resources should be swiftly provisioned and released with minimal management effort and according to the pay-as-you-go model [18]. Cloud computing can be defined as a specialized distributed computing paradigm. This paradigm differs from the traditional ones since: (1) it is massively scalable, (2) it can be encapsulated as an abstract entity that delivers different levels of services to customers outside the cloud, (3) it is driven by economies of scale, (4) it can be dynamically configured (via virtualization or other approaches) and (5) it can be delivered on-demand [14]–[17]. The associated service delivery models to cloud computing are Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS) and Software-as-a-Service (SaaS). Application providers use platforms offered as PaaS, to provision applications. These applications are offered to end-users (or possibly to other applications) as SaaS. Platforms add abstraction to the infrastructure offered as IaaS. The infrastructure is the actual dynamic pool of virtualized resources used by applications [9].

Cloud computing is not completely suitable to provision IoT applications [10]. The major limitation is related to the connectivity between the computing cloud resources in the core network and the devices at the edge. On the one side, communications between IoT applications in the cloud and their related object is achieved through the Internet. On the other side, IoT applications are latency-sensitive [19]. Fog computing is a computing paradigm that has been recently introduced to tackle these limitations [20]. It extends the cloud architecture and provides additional computing resources at the edge of the network, close to the object [20], [41]. The ultimate goal is to reduce the latency and processing delays for applications such IoT when being provisioned over hybrid cloud/fog environments.

C. ANOMALY AND EVENT DETECTION SYSTEMS

In this work, ADS (Anomaly Detection System) is distinguished from EDS (Event Detection System). The two terminologies can be confused in the literature to designate the detection of rare and exceptional events [42]. ADS consists in the identification of what can prevent a system from accomplishing its specified functions like nodes failures, errors, malfunctions and security attacks. EDS rather concerns the identification of events of interest and is identified in the specifications of an IoT surveillance system, like the detection of fire, flood, unauthorised person intrusion, etc.

For the purpose of anomaly detection, there are three main approaches: (1) rule-based detection, (2) anomaly-based detection and (3) reputation systems [53]. Rule-based approaches rely on pre-defined rules to classify captured packets as normal or abnormal. Although they detect well-known anomalies in high accuracy, they are unable to detect new attacks so as the attack signature database should always be updated [27]. Anomaly-based approaches are generally based on machine learning algorithms. They are used to detect anomalies and malicious activities after a training phase to construct a model that distinguishes normal from abnormal data [5], [28]. Although these approaches can detect new attacks, they have the disadvantage of generating false positives in a higher rate rather than rule-based approaches. Reputation systems generally rely on monitor nodes to supervise and evaluate their neighbors activities. Trust values are affected to each node. They are calculated according to the node observed behavior like data aggregation and routing [29], [30]. When a trust value is under a well-defined threshold, the associated node is considered malicious. Each approach has its key strengths and drawbacks. Therefore, hybrid approaches combining the previous mentioned approaches [31], [32] were adopted in order to maximize the detection rate (DR), minimize the false positive rate (FPR) and preserve energy consumption.

Several ADS [7], [8] are used in recent IoT systems to detect anomalies and malicious nodes in the network and provide countermeasures to mitigate them. Consequently, they are considered as a solution to improve the system reliability. Nevertheless, how to make sure that they are themselves reliable? This ''hen and eggs'' issue shows that, in addition to the integration of EDS and ADS, additional reliability enforcement techniques are needed. In this current work, a trust management system is built on top of ADS and EDS for reliability enforcement.

D. SUBSYSTEMS INTEGRATION FOR BETTER RELIABILITY

Generally speaking, software integration consists of data integration, process integration and analysis and decision integration [21]. Putting the subsystems together aims at improving business processes and simplifying the administrator work. Systems integration targets the optimization of three parameters:

- Time: using several systems equals the necessity of switching from one platform to another in order to perform specific business tasks.
- Data: the greatest advantage of systems integration is improved data accessibility. Since data is considered as a whole, system integration leads to better decision-making and business growth in general.
- Management costs: by using just one management application instead of several, administrators spend less time introducing data to the system, deploying and updating functional processes and thus, reducing energy consumption and engineering cost.

In the context of reliability, a lot of works concentrate on the integrated reliability systems of both software and hardware [22], [23]. In addition to the advantages listed so far, subsystem integration allows for the detection of more failures than the subsystems separately. For example, software failures generated by hardware failures and hardware failures caused by software failures are detected in the integrated subsystems and not detected if the subsystems are separately considered [24]. Other works integrate anomaly and failure detection subsystems of multi domains in a single system [25]. The integration allows the optimization of resources and the reduction of management cost. IoT is itself considered as an integrated set of subsystems for observing different targets and different domains. However, management subsystems are still considered as 'silos' with different targets and tools. In this work, we propose the integration of ADS and EDS in a common system to profit from the subsystem integration advantages in terms of resource optimization, management optimization and improved reliability of the overall system.

Many software architecture solutions allow for achieving subsystem integration [26]. In our work we adopt an architecture where data storage is centralized, a publish/subscribe broker is adopted as a communication middleware for real-time interaction and a common management system is integrated for ADS and EDS deployment, monitoring and update.

III. RELATED WORK

Our contributions are related to three main fields of research in IoT systems: Hybrid fog/cloud provisioning, Event and Anomaly detection systems and Trust and Reputation systems. Recent works in each domain are described hereafter. READ-IoT is shown to take the best practices and techniques in IoT for a reliable and efficient ADS and EDS integration while related works generally address one single subsystem at one time.

A. IoT APPLICATIONS PLATFORMS IN CLOUD/FOG ENVIRONMENTS

Many works provide an overview of the core issues, challenges and general frameworks for IoT services orchestration over Edge, Fog and Cloud [33]–[36]. Authors in [37] propose a novel optimization model minimizing the total cloud

provider cost to serve client requests by shifting some of the work to the fog Node. The model is formulated as a Mixed Integer Linear Problem (MILP) with an objective function minimizing the total computational cost (load) of requests using performance and QoS parameters. In [38], authors propose an architecture entitled SoFA, which is a Spark-oriented Fog architecture that leverages Spark functionalities to provide higher system utilization. This method leverages the remaining processing capacity of edge devices.

The authors in [39] propose hydle, a hybrid deployment framework for surveillance system. The hybrid deployment is based on using different data source types (WSN and MWSN) and hosting nodes (fog and cloud nodes). The approach has been evaluated in a surveillance application using incremental layered Deep Learning-based image processing. Results show processing delay optimization using a hybrid fog and cloud provisioning with different network and processing metrics. In [40], authors propose an architecture for a Platform as-a-Service (PaaS) to automate applications provisioning in a hybrid fog/cloud environment. This architecture was used in [41] to provide a demo of the proposed PaaS by deploying an IoT healthcare application components across fog and cloud nodes. The demo also depicts the support of the whole application life cycle (i.e. developing, deploying and managing).

The proposed solutions are proven efficient, and the underlying principle consists in resource-aware deployment and orchestration for reducing resource consumption and processing delays. In our work, the deployment is also driven by targets' trust level in addition to resource availability. Another difference is the adoption of the same hybrid deployment for both EDS and ADS.

B. ANOMALY AND EVENT DETECTION SYSTEMS IN IoT

Authors in [43] propose a pattern-sensitive partitioning model for IoT sensors data streams capable of paralyzing data processing in order to achieve a high detection accuracy for event pattern in a minimum delay. Authors in [44] extend Sipresk, a big data analytic platform, to detect, classify and report events in Ontario highways in a minimum delay. In [45], authors propose an anomaly detection system for asynchronous events coming from a fleet of devices. The system defines an analysis workflow for each specific use case, and it is deployed as a Cloud Web service exposing all functionalities via REST API. Authors in [46] propose a novel anomaly detection method, called Fog-Empowered anomaly detection, using an efficient hyper-ellipsoidal clustering algorithm. They also use a fog computing architecture to minimize latency in anomaly detection. In [47], authors propose a Deep-Learning algorithm to detect malicious traffic in IoT networks. The IDS was proven efficient against various attacks in IoT networks and was deployed as a standalone device in the network. A lightweight distributed IDS in a three-layered IoT architecture including the cloud, fog and edge layers was proposed in [48]. Authors in [49] propose SVELTE, a distributed intrusion detection system for IoT that detects communication attacks in 6LoWPAN networks using RPL as a routing protocol. Recent works [50]–[52] focus mainly in detecting anomalies for large scale systems using Big Data technologies. In our previous work, READ [53], a reliable event and anomaly detection system for WSN was proposed. Although the proposed system was proven to be reliable ensuring a high detection rate and low energy consumption, it focuses only on WSN and does not consider IoT challenges as security attacks where the gateway is connected to cloud servers, and IP protocol is used. In our recent work [54], an anomaly detection system that considers both WSN and IoT anomalies detection using machine learning components depending on nodes capabilities is proposed. However, similar to the other cited works, ADS is used for reliability but the reliability of ADS itself is not addressed. In this work, the reliability of both the ADS and EDS is enforced with reputation-based deployment.

C. TRUST AND REPUTATION SYSTEMS IN IoT

Trust and reputation systems are mainly used for monitoring task in WSN and IoT systems. They rely on monitor nodes or watchdogs that are used to supervise and evaluate their neighbors behaviors. Trust values are calculated for the nodes based on their observed activities, and then decision for data aggregation or routing is taken based on this evaluation [55], [56]. Many works address the problem of finding the compromise between resource consumption and reputation calculation like in [57] where monitors are periodically changed to optimize their energy. Other works aim at automating the reputation management to reduce administration cost like in [58] where authors present an architecture pattern for trusted orchestration management (TOM) in edge and cloud using a blockchain-based security solution. In READ-IoT, the reputation system enforces both EDS and ADS reliability.

We present in Table 1 the main anomaly and event detection solutions listed in this section. We summarize the main key points and strengths of these works and classify them following three main criteria: The system goal (Event or anomaly detection) and deployment reliability techniques.

The related work study shows clearly that IoT systems are designed either as EDS or ADS. To our knowledge, few works propose an integrated solution as we propose in READ-IoT. Furthermore, ADS systems are deployed for reliability, but they are supposed to be reliable themselves. This table highlights, therefore, the contribution of our work compared to related ones: an integrated ADS and EDS with an enforced reliability.

IV. READ-IoT DESIGN

This Section describes the READ-IoT framework design. It first introduces the system architecture with its different layers and components. Then, it details the several designed procedures that supports every single phase of the IoT applications life-cycle. The design is based on the

TABLE 1. Summary of recent related work about anomaly and event detection in IoT.

Work reference	Key points/Strengths	Detection		Reliable Deployment			
		Event	Anomaly	Static	Dynamic	Risk driven	Resource driven
Sharkh et al. [37]	Optimized hybrid fog/cloud provisioning	✓			✓		✓
Ben Abdallah et al. [39]	Efficient incremental layered DL-based image processing	✓			✓		✓
Yangui et al. [40]	Support of the whole IoT application life cycle	✓		✓			✓
Cramer et al. [45]	Cloud Web service via REST API		✓	✓			
Lyu et al. [46]	Efficient hyper-ellipsoidal clustering algorithm at fog resources		✓	✓			✓
Thamilarasu et al. [47]	Efficient DL algorithm for IoT anomaly detection		✓	✓			
Rettig et al. [51]	Online Anomaly Detecting for streaming application		✓		✓		✓
Abdellatif al. [57]	Adaptable and energy efficient monitoring for WSN (limited to WSN)		✓		✓	✓	✓
Pahl et al. [58]	Block-chain based solution for a trusted orchestration in fog and cloud		✓		✓	✓	

service computing life-cycle model introduced in [59]. This reference stipulates that services and applications, including IoT applications provisioning process consists of three phases: (i) Development, (ii) deployment and (iii) management. The READ-IoT architecture specification is inline with this model. The development phase consists of developing, testing and building the application executables. Application executables include all the files needed to execute the application once deployed (e.g. source code, configuration files). The READ-IoT framework provides the developer with an appropriate Integrated Development Environment (IDE), as well as, all the necessary libraries and resources implementing the required machine learning algorithms for anomaly detection, reputation management, etc. These libraries are offered as adapted development kits that could be used to assist developers at development time. The deployment phase consists of: (1) Allocating and making ready the READ-IoT resources (e.g. object, hosting containers, storage services) needed to host and execute the end-user application and (2) uploading its executables over these resources. The management phase consists in: (1) Activating the deployed IoT application in order to make it available, (2) executing it when receiving requests and (3) performing appropriate management operations when needed at run-time (e.g. migrate, scale up/down).

A. HIGH-LEVEL ARCHITECTURE

Fig. 2 shows the target IoT system (READ-IoT) that includes IoT nodes, two deployment areas (fog and cloud) for data processing, storage and analysis, management modules and application developers and subscribers. IoT nodes which are organized in a cluster based topology communicate with their relative cluster heads (CH) using different protocols (ZigBee, LoRaWAN, 6LoWPAN, etc.). CHs aggregate and send data to gateways (GWs) which have local storage and analytic capabilities. Also, CHs have cameras installed to survey their cluster zones. In fog, processing nodes are CHs, GWs, local machines or virtual machine nodes. On the cloud, processing nodes are virtual machines instances hosted in a cloud platform with high performance and analysis capabilities. The processing nodes permit to host and execute EDS or ADS components following a deployment plan. The implementation of these components depends on the supervised system. In our case, two kind of implementation are proposed: rule based for detecting well defined events or anomalies and machine-learning based for enforcing rule-based detection and for new non-expected event or anomaly discovery.

On top of IoT supervision system, a management layer contains the necessary modules that permit to monitor the IoT nodes and network, calculate and update the best deployment plan of processing components. Application subscribers receive notifications and alerts for events of interest and anomalies.

Supervised nodes and the managed system communicate thanks to a publish/subscribe communication model following a set of topics of an MQTT broker. Topics concern collected data, detected events or anomalies, QoS

data or management data like deployment plans. The choice of a publish/subscribe communication breaks any dependency between the system nodes, and then allows to easily add or remove sensors, gateways or cloud processing units. The controller is the orchestrator and the contact point in the management layer while master components are the contact points on the processing nodes. Data and calculated information are exchanged between master components and the controller through the MQTT broker.

ADS or EDS components are handled with the same deployment strategy and workflow logic which reduces their administration cost and simplifies their deployment and update.

B. PROCEDURES

The following describes procedures designed as part of READ-IoT specification to implement the phases : development, deployment and management.

1) DEVELOPMENT PROCEDURE

This layer hosts an Integrated Development Environment (IDE) that provides developers with the necessary development tools (e.g. development kits, libraries, APIs). This IDE permits developing, composing and testing application components taking into account the properties and capabilities of the target placement domain (either fog or cloud). The developed component, planned to be deployed as part of an application, is tested, validated then pushed to be handled by the deployer module.

For machine learning components, developers are provided with a pipeline implementing different known supervised algorithms for best features selection and appropriate algorithms choice based on accuracy metric. The training phase and model generation should be updated based on data manipulated in the application. The collected data should be extracted and formatted in comma-separated values. Once a model is generated, it should be updated in run-time based on new arriving data. For rule based components, developers are provided with templates implementing rules for detecting specific attacks on WSN and IoT. The rules are updated regularly based on newly discovered attacks.

2) DEPLOYMENT PROCEDURE

Once calculated or updated, the deployment plan is communicated to the master nodes. A placement flow of EDS and ADS components is described (on fog and cloud nodes). The deployer deploys components as docker containers in the dedicated processing nodes. Then, the containers are run as web services. When an application execution flow ends, results are sent by the last executed component via MQTT publish service. In case of an anomaly, the reputation values of nodes are updated. In case of a detected event, the controller sends the information to the MQTT broker and activates the actuator nodes for event handling by sending the information to its relative gateway. Also, subscribers could receive alerts from the broker through the event detection topic.

3) MANAGEMENT PROCEDURE

The management process is handled by five main modules: deployer, controller, QoS manger, risk manager and storage manager.

- Deployer: A repository (e.g. docker [60] private hub) keeps track of EDS and ADS component description (related application, order of execution in the application, performance requirement and desired placement) in a registry (e.g. JSON or XML file). The deployer calculates and updates a placement plan and deploys components as docker containers in the dedicated processing nodes. It uses QoS information updated by QoS manager, the reputation of nodes updated by the risk manager and the list of available resources (fog/cloud processing nodes) information that is regularly updated by the controller. At regular interval, it calculates the best placement plan for EDS and ADS component placement based on available resources. Then, it communicates this deployment plan to the controller whenever it is ready.
- Controller: It interacts with QoS manager and Risk manager to receive information about nodes QoS and reputations and update the resource registry. It interacts with the deployer to provide resource information (availability, QoS, reputations) and to obtain updated deployment plan. It communicates with the master nodes to get EDS and ADS data.
- Qos manager: This module checks and updates regularly nodes quality of service data (availability, communication link bandwidth and busyness). The availability of a node is verified thanks to ICMP ping messages. The communication link is checked by both Time-to-live ICMP ping parameter and head/get/put HTTP command execution time using a light HTTP server. The node is busy when a component is assigned to it for deployment.
- Risk Manager: It is in charge of calculating and updating reputation values of supervised nodes.
- Storage manager: It subscribes to the topic raw data to receive all published IoT data and store it in a scalable database.

V. READ-IoT ALGORITHMS AND MODELS

This Section presents main READ-IoT algorithms (placement calculation and cascaded detection) and models. The algorithm of placement calculator deploying k components (C_1,..,C_k) over n fog nodes (FN_1,...,FN_n) and m cloud nodes (CN_1,...,CN_m) is shown in Algorithm 1. Symbols are described in Table 2. Mainly, execution of heavy components as machine learning based components is carried out in cloud [61] since more resources are needed to ensure a faster processing time. Furthermore, the choice depends on the communication link bandwidth to ensure a minimum

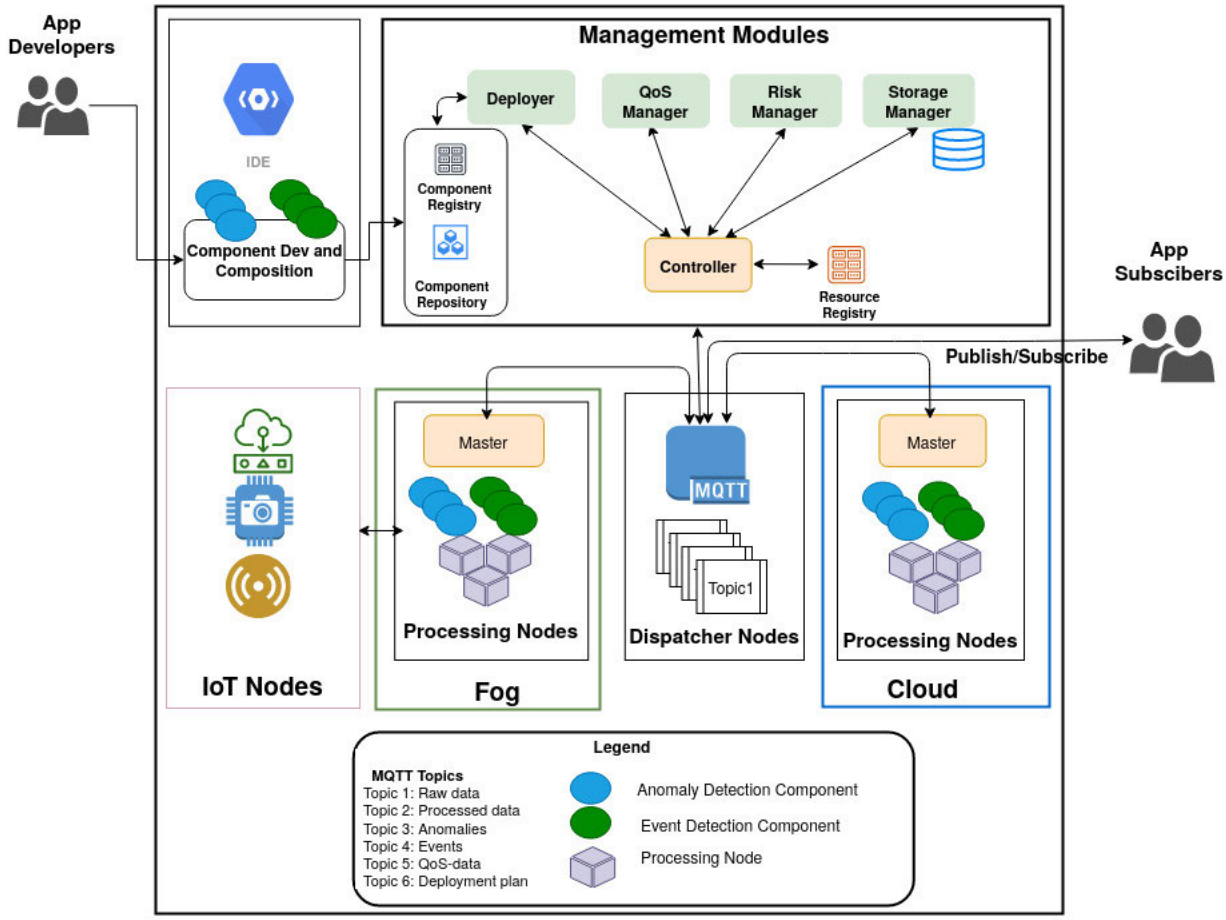


FIGURE 2. READ-IoT Architecture.

delay in sending data and receiving results from the cloud. Only nodes with reputation score over a certain threshold are elected for deployment in order to avoid compromised or suspicious nodes. Reputation threshold and the execution frequency of the algorithm are set by administrators, and depend on application security requirements and the environment risk. Finally, for faster execution, when the node is busy and a component is assigned to it for deployment, our strategy consists in favoring the free remaining nodes to avoid overhead.

A. CASCADED DETECTION APPROACH

Fig. 3 shows READ-IoT unified processing of both EDS and ADS. They both inherit from a parent class depending on the application risk and execute the same workflow. They receive data from IoT sources (SN, CH, GW) and process them either on the fog or on the cloud following the deployment plan. Fig. 4 depicts READ-IoT flowchart for event and anomaly detection. The event detection process is based on sensing data, whereas anomaly detection process is based on communications data (actions 1 and 5). The rule based detector permits to check whether there is an event/anomaly or not following specific rules (actions 2 and 6). For example, for the use case fire detection, if the temperature/humidity are higher/lower than certain thresholds, a fire event is triggered by the rule based detector. Therefore, the output of the rule based detector is binary ('YES' or 'NO'). The machine learning detector is executed after that according to the result of the rule based detector and depending on the use case (actions 3 and 7).

- For the fire detection use case, if the binary result of the rule based detector is 'NO', a double check is done by ML detector to detect unknown fluctuations that are not detected by the rule based detector. For instance, temperatures in summer and winter are different. Also temperatures at night and during the day are different. So, rules may not be suitable for such changes' detection, while a model update will detect such changes. If the output of the rule based detector is 'YES', there is no need to check using the ML component. The output

of the machine learning component for the fire detection use case is a binary class ('YES' or 'NO').

- For the unauthorized person detection use case, if the rule based detector outputs 'YES' which means a motion was detected, the machine learning detector is executed following the rule based detector to identify objects in images taken by camera, so its output will be a class (human, animal, vehicle, etc.). If it is the class human, the ML detector sends the final results 'YES', otherwise, it sends 'NO'.
- For the anomaly detection use case, the rule based detector is executed to detect if there is an anomaly. For instance, if a malicious node does a selective forwarding attack (does not forward correctly received packets), the rule based detector outputs 'YES' if it detects it. In contradiction, if it outputs 'NO', further investigation for anomaly using the machine learning detector is done. This component outputs 'YES', whenever the classifier detects an attack and 'NO' otherwise.

Based on the received result, the decision maker sends appropriate notifications about the event/anomaly (actions 4 and 8). Management modules rely on QoS data, resources information and their reputation values to update the deployment plan that permits the execution of detection components over selected nodes (actions 9,10 and 11). Also, received data are stored in database (action 12). End users receive notifications on events/anomalies they subscribe to (action 13).

Algorithm 1: Placement Calculation Algorithm

Result: Placement Plan e.g. $C1$ on FN_1, C_2 on FN_3, C_3 on $CN_3, \ldots, C_k$ on CN_m

1 **Inputs:** resources=[$FN_1, \ldots, FN_n, CN_1, .., CN_m$], components=[$C_1, \ldots, C_k$], availableNodes=[]
2 **if** *environment is vulnerable* **then**
3 set $timeperiod = tvulenerable$
4 **else**
5 set $timeperiod = tnonvulenerable$
6 **end**
7 **while** $time = timeperiod$ **do**
8 **for** N_i *in resources* **do**
9 check-availability(N_i)
10 **if** N_i *is available* **then**
11 availableNodes.add(N_i)
12 **end**
13 **end**
14 Call Dijkstra_Risk_Aware(availableNodes, components)
15 **end**

Fig. 5 depicts the detection process for each specific use case. As shown in Fig. 5 (a), EDS relies on three main components: (1) Rule Based Event Detector (RB-ED): this component has pre-defined rules for detecting events as fires based on configurable thresholds. As an example, when a temperature is higher than a threshold th-t (e.g. 57 °C) and the humidity is lower than a threshold th-h (e.g. 30%), a fire detection event is triggered, (2) Machine Learning Event Detector Component (ML-ED): detects events using machine learning techniques. It is triggered by RB-ED only when it does not detect any event, (3) Event Decision Maker (EDM): When the event is confirmed by one of the two previous detectors, EDM checks the source node reputation and reacts to detected events by sending notifications and alerts. For unauthorized person detection, Fig. 5 (b) describes the detection process which starts with the RB-ED component (Motion detector). When a motion is detected, the ML-ED permits to analyze a picture taken and verifies if a person is present or not. The EDM sends the notification when the person detection is confirmed. The use of a hybrid detection technique combining rule based and machine learning results [31], [62] has the advantage of minimizing false alerts and increasing the detection rate. Like EDS and as shown in Fig. 5 (c), ADS relies on three main components: (1) Rule Based Anomaly Detector (RB-AD): this component has pre-defined rules for detecting communication anomalies (CA) like hello flooding attacks, selective forwarding attacks (SFA) and blackhole attacks (BHA) as used by [31], (2) Machine Learning Anomaly Detector Component (ML-AD): this component detects anomalies and attacks using machine learning techniques e.g. One Class Support Vector Machines (OCSVM) for WSN anomalies, Deep Learning for IoT anomalies as described in [54]. The choice of OCSVM and Deep Learning was motivated by lessons learned from comparative studies and works related to this field. The comparative study in [5] demonstrates OCSVM efficiency in WSN anomaly detection in comparison with different machine learning techniques for anomaly detection. When it comes to the use of deep learning, the fact that this technique is known in the community as very efficient in detecting IP network intrusions was considered [63]. (3) Anomaly Decision Maker (ADM): this component permits to react to detected anomaly by sending notifications and alerts to system administrators for malicious nodes eliminating. Also, its output is used to update nodes reputations. READ-IoT cascaded detection protocol is summarized in Algorithm 2.

B. READ-IoT REPUTATION MODEL

READ-IoT uses a reputation system that keeps track of all nodes (data sources and deployment nodes) reputation values using the Beta function [64]. Let s be the number of successful past actions for a Node N_i, and let f be the number of its unsuccessful past actions. Then, the reputation of the node RN_i can be estimated as shown in the following equation:

$$RN_i = (s+1)/(s+f+2) \quad (1)$$

For rule-based detection, the value of s is incremented whenever a rule is checked as successful. Inversely f is incremented if the rule is not respected. Basically for Selective Forwarding Attack (SFA), s is incremented when a node forwards correctly a received packet. When the node drops the packet,

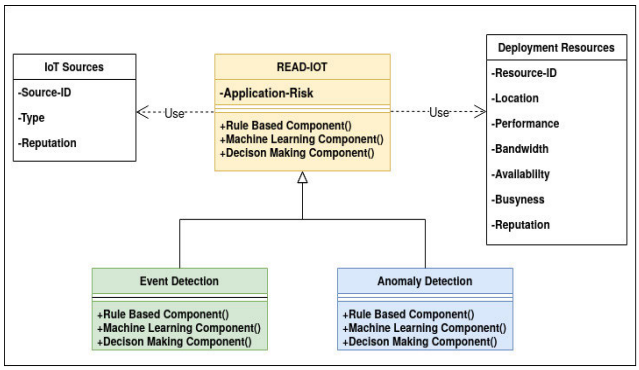


FIGURE 3. READ-IoT as a unified system.

f is then incremented. For machine learning detection, s is incremented whenever a packet is classified as normal. If the packet is classified as malicious f is incremented.

Algorithm 2: READ-IoT Cascaded Detection Algorithm

```
if environment is vulnerable then
    set tperiodAnomaly, THsAnomaly = tvulAnomaly, THsvulAnomaly
    set tperiodEvent, THsEvent = tvulEvent, THsvulEvent
else
    set tperiodAnomaly, THsAnomaly = tnonvulAnomaly, THsnonvulAnomaly
    set tperiodEvent, THsEvent = tnonvulEvent, THsnonvulEvent
end
while True do
    check (lastplacementplan)
    if time= tperiodAnomaly then
        call ADS (THsAnomaly)
        /* Reputation values updated */
    else
        if time= tperiodEvent then
            call EDS (THsEvent)
            /* Notifications sent and actuators activated */
        end
    end
end
```

TABLE 2. Symbols and notations.

Symbol	Description
C_i	Component number i
N_i	Cloud of fog Node number i
CN_i	Cloud Node number i
FN_i	Fog Node number i
$environment$	The status of the surveyed environment either $vulnerable$ or $nonvulnerable$
$timeperiod$	The time period of calculating a new placement plan which may be $tvulnerable$ if the environment is vulnerable or $tnonvulnerable$ if it is not
$tperiodAnomaly$	The time period of executing Anomaly detection components which may be $tvulAnomaly$ if the environment is vulnerable or $tnonvulAnomaly$ if it is not
$THsAnomaly$	Thresholds used by the anomaly detection components which may be $THsvulAnomaly$ if the environment is vulnerable or $THsnonvulAnomaly$ if it is not
$tperiodEvent$	The time period of executing Event detection components which may be $tvulEvent$ if the environment is vulnerable or $tnonvulEvent$ if it is not of
$THsEvent$	Thresholds used by the event detection components which may be $THsvulEvent$ if the environment is vulnerable or $THsnonvulEvent$ if it is not
$lastplacementplan$	The last placement plan

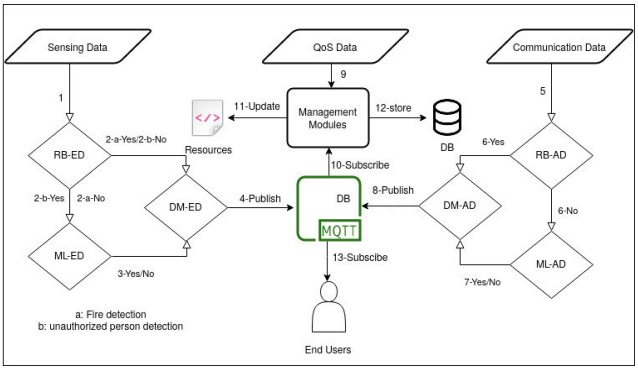


FIGURE 4. READ-IoT flow chart.

C. READ-IoT DEPLOYMENT MODEL

The problem of latency minimizing is modeled using Dijkstra's algorithm [39], [65], [66] to find the optimal deployment plan that optimizes the end to end delay of detection and response. In our model, in addition to the delay constraint, the deployment risk constraint is considered to guarantee a secure deployment based on nodes reputations (*Dijkstra_Risk_Aware*). Fig. 6 describes the optimal solution as the shortest path delay in executing the application component flow in the hybrid fog/cloud environment and the notification of the end user (EU). Mainly two different delays are considered: latency in communication with a deployment node Ni (LcN_i) and latency in processing a component C_j over the deployment node N_i ($L_pC_jN_i$). The end to end delay of deployment of an application using three components $C1$, $C2$ and $C3$ in the fog/cloud environment over three nodes $N1$, $N2$ and $N3$ that may be located in fog or cloud and with reputation values RN_i that should be higher than a predefined threshold TH is shown in the following optimization

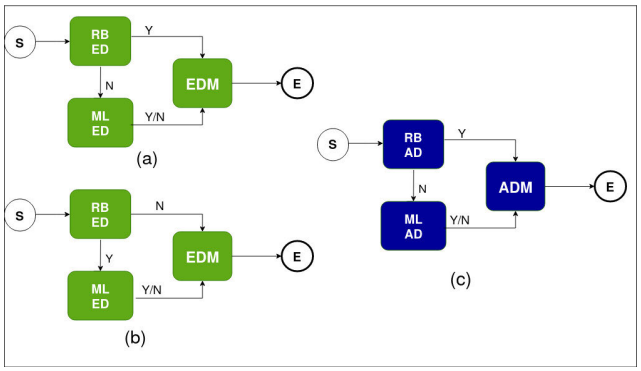


FIGURE 5. READ-IoT component flow (a: Fire detection process, b: unauthorized person detection process, c: Anomaly detection process).

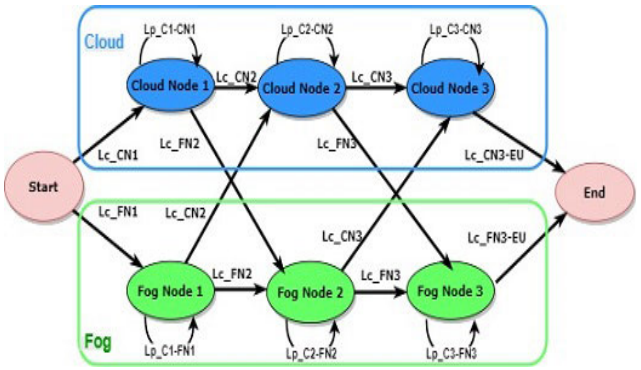


FIGURE 6. Shortest path finding deployment.

problem:

$$\min(End\ to\ End\ delay)$$
$$\text{s.t. } RN_i \geq TH,\ i = 1, 2, 3. \tag{2}$$
$$End\ to\ End\ delay = \sum_{i,j=1}^{3} (LcN_i + LpC_jN_i) + LcN_3EU \tag{3}$$

VI. READ-IoT AND USE CASES IMPLEMENTATION

This Section presents some details about READ-IoT implementation with a description of the software tools and evaluation metrics.

A. DEVELOPMENT IDE

EDS and ADS were developed using WING PYTHON IDE. They were deployed as REST Web services docker containers in the hybrid fog/cloud environment as Event Detectors (ED) or Anomaly Detectors (AD). Tensorflow and Scikit-learn PYTHON libraries were used in developing the machine learning components. Furthermore, READ-IoT modules as deployer, controller, QoS Risk and storage managers were developed using the same IDE. A dedicated machine in the local network was used mainly for management purpose (calculating and updating the placement plan).

TABLE 3. AWS used EC2 instances characteristics.

Name	Model	vCPU	Mem (GiB)	Network Performance
CN1	t2.large	2	8	Low to Moderate
CN2	t2.xlarge	4	16	Moderate
CN3	t2.2xlarge	8	32	Moderate

B. PROOF OF CONCEPT

In order to validate the proposed architecture and provide a proof of concept, a real IoT system was implemented and deployed it in the Polytechnic School of Tunisia. Fig. 7 shows a high-level view of the proposed prototype. Two WSN clusters composed each of one Cluster Head and three different sensing nodes (Arduino Uno nodes) were used: two motion detection nodes based on a PIR sensor (hc-sr501 sensor) and one fire detection node based on temperature-humidity sensor (DHT22 sensor) and gas sensor (MQ7 sensor). The set of used devices locations is presented in Fig. 8. Motion detection nodes were deployed in the school garden and fire detection nodes were deployed inside offices. The sensing nodes communicate with their respective CH (Raspberry Pi 3 node) using ZigBee protocol based on RF modules (ZB S2C Pro modules) which support indoor communication with a range up to 75-100 meters and an outdoor communication over 300 meters. The communication security is ensured by a 128-bit AES encryption algorithm and a 4-byte message integrity code (MIC). CHs aggregate sensing node data and forward results to a gateway node (Raspberry Pi 3 node) located in SERCOM Lab office in Tunisia.[1] CHs are equipped also with cameras (Raspberry PI 5MP cameras) to survey their cluster zones. At the level of the gateway, a MySQL Database is used to store collected data. The gateway node communicates with a local server where three VMs with similar capabilities (Mem: 4 GB, vCPU:1) are deployed and ready to be used as fog nodes (FN1, FN2 and FN3) in addition to the gateway node. The gateway is connected via Internet to a cloud service platform (Amazon Web Services) [2] [3] where three EC2 instances with different performance capabilities described in Table 3 (CN1, CN2 and CN3) are deployed and ready to be used as cloud nodes. Sources codes are available at GitHub.[4]

C. APPLICATIONS USING READ-IoT FRAMEWORK

To validate the READ-IoT framework, three different applications were implemented and deployed: two event detection applications and one anomaly detection application that are described hereafter.

[1] http://www.ept.rnu.tn/laboratoires/sercom-laboratoire-systemes-electroniques-et-reseaux-de-communications/
[2] https://aws.amazon.com
[3] http://aws.amazon.com/es/ec2
[4] https://github.com/aybsyah/READ-IoT

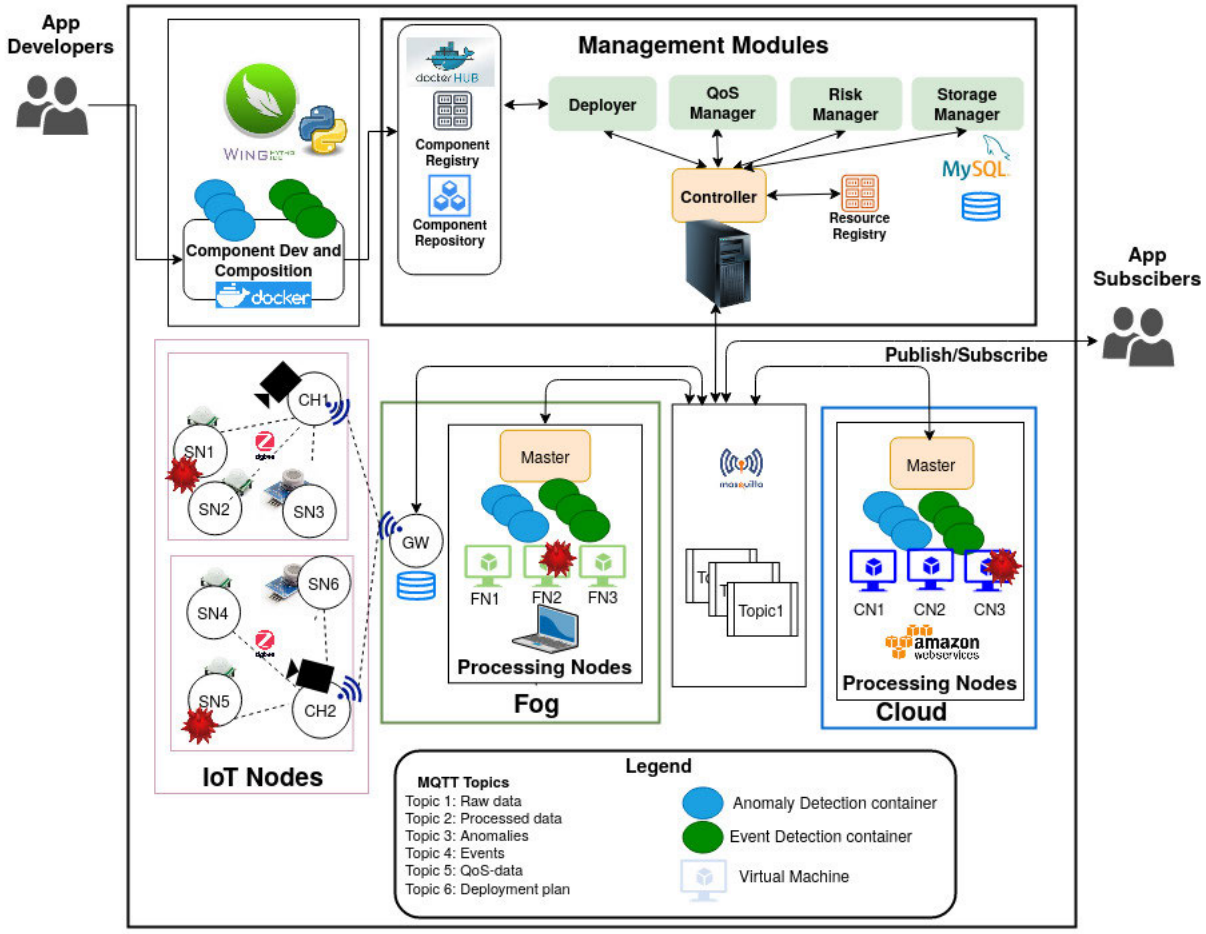


FIGURE 7. READ-IoT framework implementation.

- The first event detection application is a fire detection application. Three main components (RB-ED, ML-ED and EDM) are used. RB-ED triggers the ML-ED (OCSVM based classifier) based on temperature/humidity and gas sensors data. If the fire is confirmed by at least one component, EDM is executed to send notifications to MQTT application subscribers via the MQTT Broker.
- The second event detection application is an unauthorized person detection and response application. The RB-ED uses motion sensor data to detect intruders. When a motion is detected by RB-ED, the camera at CH node takes a picture of the surveyed zone and the ML-ED (Deep Learning object detection classifier) is triggered to analyze the picture, classify detected objects in the picture and confirm if the motion is caused by a real intruder (person) or a false alert generated by the motion sensor. If an intruder is detected, EDM sends notifications to MQTT application subscribers.
- The anomaly detection application is based on three components (RB-AD, ML-AD and ADM). RB-AD uses predefined rules to detect network anomalies. ML-AD used OCSVM for WSN anomaly detection and Deep Learning for IoT intrusion detection. When the anomaly is confirmed by at least one of two detectors, the ADM sends notifications to update nodes reputation values and blacklist malicious nodes if their reputation values is below a predefined threshold.

In the following, each use case is presented and evaluated separately. However, they may be run at the same time. As an instance, Fig. 8 depicts a screenshot of cartography web application designed to provide a global and real view of the surveyed zone. Different attacks against nodes as well as a fire event are detected and shown in this cartography in real-time. The nodes under attacks are shown in red circles with low reputation values, and event messages are shown over source nodes (e.g. 'Fire Detected').

VII. EXPERIMENTAL EVALUATION

This Section presents the system setup, evaluation metrics, placement calculation, EDS and ADS evaluation.

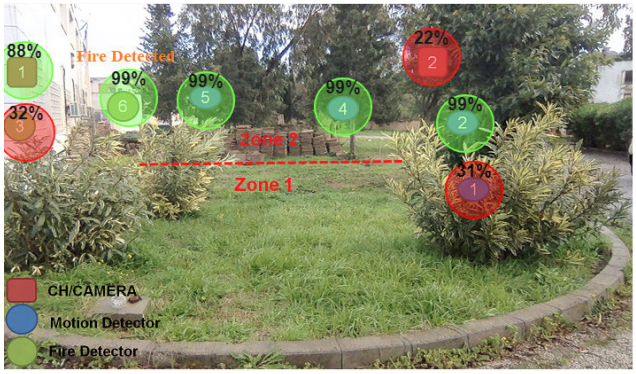


FIGURE 8. Real IoT network deployed in polytechnic school.

A. SYSTEM SETUP

READ-IoT system needs two phases: a bootstrap phase and run-time adaptation phase. In the bootstrap phase, the initial policies settings, configurations (thresholds, applications be provisioned), and resources are set by the administrator. Then, the system enters a run-time adaptation phase where any changes should be considered and treated to re-calculate the best deployment plan in real-time ensuring a high detection accuracy and low end-to-end delay.

B. EVALUATION METRICS

To assess the efficiency of our proposed system, certain parameters are changed in the run-time adaptation phase, and READ-IoT is monitored how it handles these changes dynamically and efficiently. The deployer should be able to update the placement plan following a change in the Internet Speed (communication bandwidth), nodes availability, data size and nodes reputations. For provisioning, READ-IoT should use the last updated placement plan ensuring both low end to end detection and response delay and high detection accuracy. The detection accuracy is calculated in the following equation:

$$Accuracy = \frac{TP + TN}{FP + FN + TP + TN} \tag{4}$$

- True Positives (TP): number of attack instances correctly classified as attacks.
- True Negatives (TN): number of normal instances correctly classified as normal.
- False Positives (FP): number of normal instances classified as attacks.
- False Negatives (FN): number of attack instances classified as normal.

To evaluate the impact of the communication link bandwidth, two different Internet speed variations were used (low: less than 512 Kbps and high: greater than 2 Mbps). Table 4 provides a summary of the used evaluation metrics (metric, related domain, description and equation). For instance, for the hybrid fog cloud provisioning, the metric 'End to End delay' was used. This metric is calculated based on two other metrics (communication and processing latencies). Equation 3 was used to calculate this metric.

TABLE 4. Evaluation metrics.

Domains: 1-Hybrid fog cloud provisioning, 2-Event and anomaly detection systems, 3-Trust and reputation systems

Metrics/domain	Description	Formulas
Communication latency/(1)	Latency in communication with a deployment node Ni (seconds)	LcN_i
Processing latency/(1)	Latency in processing a component C_j over the deployment node N_i (seconds)	LpC_jN_i
End to End delay/(1)	Delay in seconds taken after executing all applications components and sending notification to end user (seconds)	Equation 3
Accuracy/(2)	The ratio of number of correct predictions to the total number of input samples (percentage)	Equation 4
Reputation/(3)	The reputation value of nodes based on successful and unsuccessful past actions (percentage)	Equation 1

C. REAL-TIME PLACEMENT PLAN CALCULATION AND UPDATE

Fig. 9 depicts the real-time placement calculation and update following QoS data reception. To test the efficiency of the process, a low Internet speed, one fog node (FN1) and three cloud nodes (CN1, CN2 and CN3) as resources, three components (a rule based component C1, a machine learning based component C2 and a rule based decision making component C3) to be hosted in the fog/cloud environment were used. Fog nodes FN2 and FN3 were not started in purpose to minimize memory usage at the local machine. The deployer picks the only available fog resource FN1 to host C1. It chooses CN3, the best cloud VM instance in performance capability to host C2 and CN2 (second in performance capability in cloud nodes) to host C3. Few seconds later, CN3 is shutdown, the QoS manager that checks continuously the availability of all resources detects that quickly within a delay of few seconds and sends the information to the controller to update the available resource list. Then, CN3 is deleted from the placement plan and replaced by the next cloud node (CN2) to host component C2 that is a machine learning component. CN1 which was not used for placement previously joins the deployment nodes to host the third component C3 instead of CN2 that was assigned C2. When CN3 is up again, QoS manager notifies the controller and a new placement plan is calculated in which CN1 (least in performance capability in cloud nodes) is deleted and replaced by CN3 (best in performance capability) to host the machine learning component C2. CN2 which was assigned C2 previously takes charge of C3 instead.

D. EDS EVALUATION

READ-IoT was executed and evaluated with the fire detection and the unauthorized person detection applications that have three components: C1:RB-ED, C2:ML-ED and C3:EDM. These components are executed orderly over a set of deployment nodes in fog and cloud.

For the fire detection application, a fire is set up as shown in Fig. 10. Table 5 depicts the end to end delay following

FIGURE 9. Real-time placement plan update following QoS data reception.

FIGURE 10. Fire detection prototype.

FIGURE 11. Unauthorized person detection prototype.

TABLE 5. Fire detection application: end-to-end delay using different placement scenarios.

Deployment	RB-ED	ML-ED	EDM	High Speed Internet	Low Speed Internet
Static (fully fog)	FN1	FN2	FN3	10.44 s	10.44 s
Static (fully cloud)	CN1	CN2	CN3	6.86 s	25.03 s
Static (hybrid)	CN1	FN2	FN3	09.75 s	21.71 s
Static (hybrid)	FN2	CN2	CN3	08.01 s	24.31 s
Dynamic (READ-IoT)	CN2	CN3	CN1	6.34 s	-
Dynamic (READ-IoT)	FN1	FN2	FN3	-	10.44 s

different deployment scenarios: fully fog deployment, fully cloud deployment and a hybrid fog/cloud deployment. With a low Internet speed, the best choice was to deploy all components in the fully fog environment to avoid the communication delay. In contrast, with a high Internet speed, the ultimate choice was to run all components in cloud. Furthermore, the deployment of the three components in a hybrid environment gives an average result and a compromise between the two previous scenarios since the communication task is avoided for some components deployed in fog. Besides, Fig. 12 demonstrates the effect of machine performance capability on the end to end delay minimization by varying its capability and the file size used for training and prediction by the machine learning components. These components are considered heavier than rule based ones and time consuming in execution especially in the training phase. Results showed that the file size affects also the latency. With small data files sent, cloud computing was the ultimate solution. But, when the file size sent increases, fog computing gives better result despite the use of the best AWS machine in performance capability (CN3). The previous deployment experiments shows the importance of one parameter related to communication bandwidth which is the Internet speed for end to end delay minimization. Therefore, READ-IoT which checks and updates regularly the communication link bandwidth and considers this parameter in calculating the best placement plan gives the best result (least end to end delay).

For the surveillance application which is an unauthorized person detection and response application. Detection test of persons crossing the zone where motion detectors are installed is shown in Fig. 11. The challenge was to deploy the Deep Learning component (ML-ED) which is a heavy and time consuming component in execution especially in training phase. Table 6 compares different deployment scenarios with a high Internet speed. The minimum delay was obtained by deploying all components in cloud and running the Deep Learning component over the best machine in performance CN3 (fully cloud scenario3). With a hybrid deployment scenario, the longest end to end delay was obtained when running the Deep Learning component over the fog node FN1 (hybrid scenario1). READ-IoT considers deploying heavy components over the best machine in performance. Therefore, it gives the best result (least end to end delay).

E. ADS EVALUATION

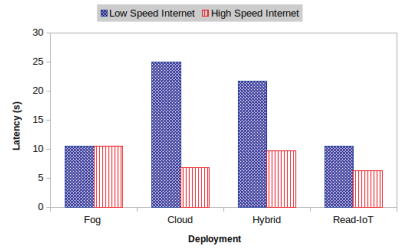
READ-IoT is executed and evaluated with an anomaly detection application that has three components : C1:RB-AD, C2:ML-AD (DL for IoT anomaly detection and OCSVM for WSN anomaly detection) and C3:ADM with two different datasets.

1) DATASETS AND MACHINE LEARNING ALGORITHMS

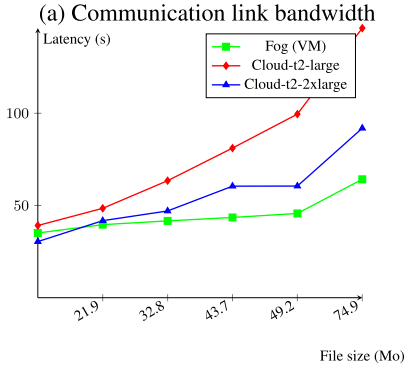
The anomaly detector detects two families of IoT anomalies. For the WSN part (Sensing Nodes, CHs and gateways), most

TABLE 6. End-to-end delay measurement using different placement scenarios.

Deployment Scenario	RB-ED	ML-ED	EDM	End to End delay
Static (fully cloud scenario1)	CN1	CN2	CN3	22.86s
Static (fully cloud scenario2)	CN2	CN1	CN3	21.29s
Static (fully cloud scenario3)	CN2	CN3	CN1	20.87s
Static (hybrid scenario1)	CN1	FN1	CN2	67.87s
Static (hybrid scenario2)	FN1	CN3	CN2	45.13s
Static (hybrid scenario3)	CN1	CN3	FN1	34.24s
Dynamic (READ-IoT)	CN2	CN3	CN1	20.87s



(a) Communication link bandwidth



(b) File size and machine performance

FIGURE 12. End to End delay varying the link bandwidth and the data file size.

known anomalies are attacks as selective forwarding attacks, black hole attacks, and hello flooding. Data generated from Castalia 3.2 simulator for WSN environment simulation were injected to the network. The simulation was related to the deployment of 15 nodes in a field size of 100 x 100 m2 [53]. The simulation time was 1000s. Normal data was collected when the network is running without attacks. Then, a selective forwarding attack was simulated, and attack data were collected. The selected OCSVM features (packet rate, consumed energy) are standardized and normalized by subtracting the mean and dividing by the standard deviation for each feature. Cross validation technique was used to select the best

TABLE 7. Intrusion activities.

Intrusion type	Description
dos	attacks on availability of services
probe	monitoring or probing in order to obtain host information
u2r	unauthorized access to privileged userâs account
r2l	unauthorized remote access

values of (gamma, nu) parameters for OCSVM technique with an allowable false alarm rate set in the validation and test phase.

When it comes to the network entities such as gateways, servers, virtual machines, the IP protocol is used for communication. Therefore, all kinds of IP attacks are considered as anomalies that target the network. For this reason, NSL-KDD [67]–[70] data were used in exchanged packets to simulate different types of IP attacks. Other datasets may be considered, but the objective of the performed evaluation is to demonstrate the feasibility and the reliability of the proposed approach in detecting these attacks and may be used to detect other types of IP attacks if we provide other datasets and consider training and generating new models. The dataset contains data packets relative to four types of intrusions mainly Probing, Remote to User (R2L), Denial of Service (DoS) and User-to-Root (U2R) as described in Table 7 in addition to normal traffic packets are injected. The same neural network architecture in [54] was used with 3 hidden layers containing 10, 50 and 10 neural nodes in order to identify records as normal or malicious. ReLU served as the activation function for the hidden layers whereas softmax was employed at the output layer.

2) OBSERVATIONS

The existence of two malicious nodes FN2 and CN3 was simulated by assigning most malicious packets to these nodes and normal packets to remaining nodes FN1, FN3, CN1 and CN2. Table 8 shows the reputation values obtained after running the anomaly detection application over the different deployment nodes and varying the sample data size. The two malicious nodes are detected by their low reputation values with a high accuracy (over 98%) using the ML-AD (based on Deep Learning technique) even with small sample data size. Therefore, READ-IoT automatically eliminates these nodes from deployment resources to avoid the risk of deploying components on risky nodes. For WSN anomaly detection, a selective forwarding attack is simulated by having two malicious nodes dropping 80% of received packets (SN1 in cluster 1 and SN5 in cluster 2). Fig. 13 depicts the source node reputation values. The two malicious nodes are detected as suspicious nodes by their low reputation values calculated using the RB-AD. The ML-AD based on OCSVM confirms the attack with a high accuracy (over

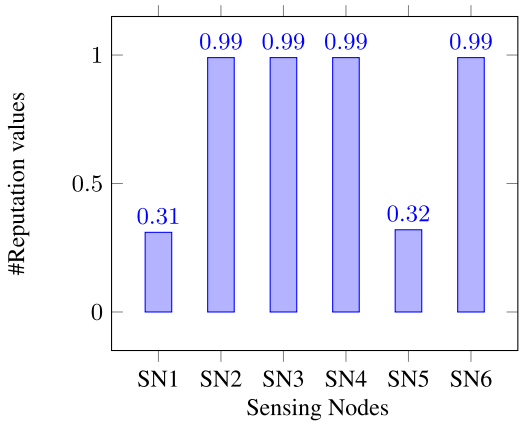


FIGURE 13. Sensing nodes reputation values.

```
Available Resources: ['FN1', 'CN1', 'CN2', 'CN3']
Placement Plan: C1 on FN1 , C2 on CN3 , C3 on CN2

Message from ADS:
Reputation values:  FN1: 0.9937  ,FN2: 0.0039  ,FN3: 0.6667
                    CN1: 0.8333  ,CN2: 0.9697  ,CN3: 0.0018
Available Resources: ['FN1', 'CN1', 'CN2', 'CN3']
Placement Plan: C1 on FN1 , C2 on CN2 , C3 on CN1

Available Resources: ['FN1', 'CN1', 'CN2', 'CN3']
Placement Plan: C1 on FN1 , C2 on CN2 , C3 on CN1
```

FIGURE 14. Real-time placement plan update following reputation values reception.

0.91). Therefore, these nodes are considered as malicious nodes and consequently blacklisted. Fig. 14 demonstrates the placement plan update following reputation values reception form ADS. When a message is received by the placement calculator with reputation values, CN3 which was used previously for placement has a very low reputation value (0.0018). Automatically, the deployer excludes it from the deployment plan and replaces it by CN1 though it remains an available resource for usage. Inversely, FN2 and FN3 are not available as resources since they are not started. Fig. 15 shows the execution time for training and prediction using OCSVM for WSN anomaly detection and Deep Learning for IoT anomaly detection. OCSVM showed similar plots for training and prediction even-though training time is obviously higher than prediction time. For Deep Learning the training phase was very time consuming in contrast with a stable execution time for prediction. Therefore, READ-IoT considers deploying time consuming components in cloud as training machine learning algorithms when the communication link bandwidth is high.

F. RESULTS' SUMMARY AND LEARNED LESSONS

This work has allowed us to show READ-IoT efficiency but also to identify some limitations. The lessons learned can be summarized in the following points:

- Classically, EDS and ADS are separate systems, and even if they co-exist in a single system, they are generally managed and processed with different tools and technologies. READ-IoT implementation and evaluation show that integrating EDS and ADS can be performed with a common management system and a common workflow processing. This is very practical to reduce the management cost. Also, the cascaded activation of rule-based and machine-learning processing is interesting for a better identification of outliers in both EDS and ADS.
- ADS is a way to provide a reliability feature to EDS. But, it needs itself to be reliable to accomplish correctly its mission. Reputation-aware placement is showed to be an efficient technique to enforce the reliability of both EDS and ADS.

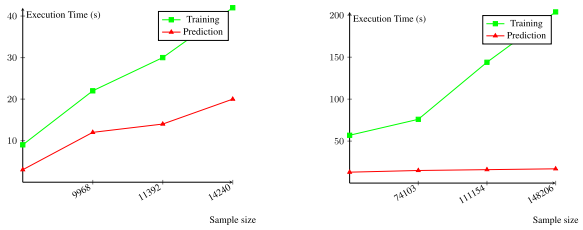


FIGURE 15. Comparison of ML-AD execution time varying the sample size.

TABLE 8. Deployment nodes reputation values.

Sample size	FN1	FN2	FN3	CN1	CN2	CN3	Accuracy
100	0.958	0.037	0.500	0.666	0.800	0.019	0.989
200	0.963	0.020	0.666	0.500	0.916	0.008	0.988
400	0.986	0.010	0.666	0.666	0.900	0.004	0.986
600	0.989	0.005	0.666	0.750	0.954	0.003	0.984
800	0.993	0.004	0.666	0.800	0.928	0.002	0.990
1000	0.993	0.003	0.666	0.875	0.962	0.001	0.988

TABLE 9. List of Abbreviations.

6LoWPAN	IPv6 Low power Wireless Personal Area Networks
ADS	Anomaly Detection System
CH	Cluster Head
COAP	Constrained Application Protocol
DL	Deep Learning
EDS	Event Detection System
GW	Gateway
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
LoRaWAN	Long Range Wide Area Network
MQTT	Message Queuing Telemetry Transport
OCSVM	One Class Support Vector Machines
RPL	Routing Protocol for Low-Power and Lossy Networks
SN	Sensor Node
WSN	Wireless Sensor Network

- Integrating both resource-aware and reputation-aware deployment allows to find an interesting compromise between reliability and real-time processing constraints. The fog/cloud paradigm is the basis of such flexible and hybrid deployment but a smart deployment plan calculation is required to reach this goal.
- Reputation-based classification is the main technique used in this work for reliability enforcement. Enhancing the detection algorithms is also an efficient technique towards reliability. Indeed, recent techniques about reinforcement-learning machine learning algorithms can adapt to system evolution and change [71], [72]. It can be an interesting extension to our work to adapt such algorithms and to test them on the fog and on the cloud.
- The current work makes an implicit assumption that all IoT parts can be under a common management control. Indeed, all data flow go through the controller that makes decisions about reputation, QoS and deployment plan. This assumption does not hold when considering a distributed IoT under different management domains. A collaboration between the domain managers is required to achieve both reliability and real-time processing.

VIII. CONCLUSION AND PERSPECTIVES

This paper introduces a Reliable Event and Anomaly Detection Framework for the Internet of Things (READ-IoT for short). The designed framework supports outliers management in IoT. It handles events and anomalies thanks to a common and integrated rule-based and machine learning-based detection. The ultimate goal is to reduce operating cost, management complexity and to enhance reliability.

READ-IoT is designed and provisioned over a hybrid cloud/fog ecosystem to address the specific IoT applications requirements such as the overhead-effectiveness and the latency sensitivity. The provisioning process covers the whole IoT applications life-cycle (i.e. develop, deploy, and manage). The resources provisioning relies on a reputation-aware deployment that takes into account the vulnerability of the deployment at the target cloud/fog hosts.

To validate these findings and to show the feasibility of the proposed approach, READ-IoT was implemented and evaluated using a real-life IoT applications such as fire detection and unauthorized human detection solutions. Several scenarios of anomalies and events were conducted to experiment the system efficiency of its reliability components. The performed experiments validate the efficiency of READ-IoT in terms of event detection accuracy and real-time processing. It also shows that the overhead due to the integration of cloud/fog domains is reasonable.

As for the next steps, we plan to re-architect the IoT system management to hierarchical fashion. Indeed, in the current work, all the considered resources are supervised using the same rule-based and machine learning process flows. Furthermore, the sensors are all connected to the same publish/subscribe broker. A hierarchical management architecture can be adopted to reuse the management layer of the framework. The management layer centralizes all collected data and metadata, builds the trust management layer and defines the deployment plan. The challenge consists in connecting efficiently the management subsystems to the framework management layer so that the communication latency overhead is minimized and real-time processing is preserved.

ACKNOWLEDGMENT

The authors would like to acknowledge Prof. K. Drira, the Head of SARA research team and at LAAS-CNRS, for his support.

REFERENCES

[1] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Comput. Netw.*, vol. 54, no. 1, pp. 2787–2805, 2010.
[2] G. Kumar and P. Tomar, "A survey of IPv6 addressing schemes for Internet of Things," *Int. J. Hyperconnectivity Internet Things*, vol. 2, no. 2, pp. 43–57, Jul. 2018.
[3] F. A. Kraemer, A. E. Braten, N. Tamkittikhun, and D. Palma, "Fog computing in healthcare—A review and discussion," *IEEE Access*, vol. 5, pp. 9206–9222, 2017.
[4] A. Fawzy, H. M. O. Mokhtar, and O. Hegazy, "Outliers detection and classification in wireless sensor networks," *Egyptian Informat. J.*, vol. 14, no. 2, pp. 157–164, Jul. 2013.
[5] V. Garcia-Font, C. Garrigues, and H. Rifà-Pous, "A comparative study of anomaly detection techniques for smart city wireless sensor networks," *Sensors*, vol. 16, no. 6, p. 868, Jun. 2016.
[6] A. Lazarevic, L. Ertoz, V. Kumar, A. Ozgur, and J. Srivastava, "A comparative study of anomaly detection schemes in network intrusion detection," in *Proc. SIAM Int. Conf. Data Mining*, May 2003, pp. 25–36.
[7] M. Turkoz, S. Kim, Y. Son, M. K. Jeong, and E. A. Elsayed, "Generalized support vector data description for anomaly detection," *Pattern Recognit.*, vol. 100, Apr. 2020, Art. no. 107119.
[8] A. N. Ahmed, F. B. Othman, H. A. Afan, R. K. Ibrahim, C. M. Fai, M. S. Hossain, M. Ehteram, and A. Elshafie, "Machine learning methods for better water quality prediction," *J. Hydrol.*, vol. 578, Nov. 2019, Art. no. 124084.
[9] S. Yangui, R. H. Glitho, and C. Wette, "Approaches to end-user applications portability in the cloud: A survey," *IEEE Commun. Mag.*, vol. 54, no. 7, pp. 138–145, Jul. 2016.
[10] S. Yangui, "A panorama of cloud platforms for IoT applications across industries," *Sensors*, vol. 20, no. 9, p. 2701, May 2020.
[11] T. Sultana and K. A. Wahid, "IoT-guard: Event-driven fog-based video surveillance system for real-time security management," *IEEE Access*, vol. 7, pp. 134881–134894, 2019.
[12] Q. Xu, W. Zheng, X. Liu, and P. Jing, "Deep learning technique based surveillance video analysis for the store," *Appl. Artif. Intell.*, vol. 34, no. 14, pp. 1055–1073, Dec. 2020.
[13] S. Arjunan and S. Pothula, "A survey on unequal clustering protocols in wireless sensor networks," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 31, no. 3, pp. 304–317, Jul. 2019.
[14] P. Mell and T. Grance, "The NIST definition of cloud computing," NIST, Gaithersburg, MD, USA, Tech. Rep. Special Publication 800-145, 2011.
[15] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *Commun. ACM*, vol. 53, no. 4, pp. 50–58, 2010.
[16] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: State-of-the-art and research challenges," *J. Internet Services Appl.*, vol. 1, no. 1, pp. 7–18, May 2010.
[17] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, "A break in the clouds: Towards a cloud definition," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 1, pp. 50–55, 2008.
[18] R. L. Grossman, "The case for cloud computing," *IT Prof.*, vol. 11, no. 2, pp. 23–27, 2009.
[19] L. Jiao, R. Friedman, X. Fu, S. Secci, Z. Smoreda, and H. Tschofenig, "Cloud-based computation offloading for mobile devices: State of the art," in *Proc. Future Netw. Mobile Summit*, Dec. 2013, pp. 1–11.

[20] C. Mouradian, D. Naboulsi, S. Yangui, R. H. Glitho, M. J. Morrow, and P. A. Polakos, "A comprehensive survey on fog computing: State-of-the-art and research challenges," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 1, pp. 416–464, 1st Quart., 2018.
[21] B. Gold-Bernstein and W. Ruh, *Enterprise Integration: The Essential Guide to Integration Solutions*. Reading, MA, USA: Addison-Wesley Longman Publishing, 2004.
[22] J. Park, H.-J. Kim, J.-H. Shin, and J. Baik, "An embedded software reliability model with consideration of hardware related software failures," in *Proc. IEEE 6th Int. Conf. Softw. Secur. Rel.*, Jun. 2012, pp. 207–214.
[23] S. Sinha, N. K. Goyal, and R. Mall, "Survey of combined hardware–software reliability prediction approaches from architectural and system failure viewpoint," *Int. J. Syst. Assurance Eng. Manage.*, vol. 10, no. 4, pp. 453–474, Aug. 2019.
[24] E. Feng, J. Zheng, and C. Liu, "An integrated reliability model of hardware-software system," in *Proc. 10th Int. Conf. Rel., Maintainability Saf. (ICRMS)*, Aug. 2014, pp. 577–580.
[25] C. Mutha, D. Jensen, I. Tumer, and C. Smidts, "An integrated multidomain functional failure and propagation analysis approach for safe system design," *Artif. Intell. Eng. Des., Anal. Manuf.*, vol. 27, no. 4, pp. 317–347, Nov. 2013.
[26] S. S. Gokhale and K. S. Trivedi, "Reliability prediction and sensitivity analysis based on software architecture," in *Proc. 13th Int. Symp. Softw. Rel. Eng.*, Nov. 2002, pp. 64–75.
[27] N. A. Alrajeh, S. Khan, and B. Shams, "Intrusion detection systems in wireless sensor networks: A review," *Int. J. Distrib. Sensor Netw.*, vol. 9, no. 5, May 2013, Art. no. 167575.
[28] Y. Zhang, N. Meratnia, and P. J. M. Havinga, "Distributed online outlier detection in wireless sensor networks using ellipsoidal support vector machine," *Ad Hoc Netw.*, vol. 11, no. 3, pp. 1062–1074, May 2013.
[29] A. A. Ghamdi, M. Aseeri, and M. R. Ahmed, "A novel trust and reputation model based WSN technology to secure border surveillance," *Int. J. Future Comput. Commun.*, vol. 2, no. 3, pp. 263–265, Jun. 2013.
[30] S. Srivastava and K. Johari, "A survey on reputation and trust management in wireless sensor networks," *Int. J. Sci. Res. Eng. Technol.*, vol. 1, no. 3, pp. 139–149, 2012.
[31] Y. Maleh, A. Ezzati, Y. Qasmaoui, and M. Mbida, "A global hybrid intrusion detection system for wireless sensor networks," *Procedia Comput. Sci.*, vol. 52, no. 1, pp. 1047–1052, 2015.
[32] M. M. Ozcelik, E. Irmak, and S. Ozdemir, "A hybrid trust based intrusion detection system for wireless sensor networks," in *Proc. Int. Symp. Netw., Comput. Commun. (ISNCC)*, May 2017, pp. 1–6.
[33] A. Mavromatis, C. Colman-Meixner, A. P. Silva, X. Vasilakos, R. Nejabati, and D. Simeonidou, "A software-defined IoT device management framework for edge and cloud computing," *IEEE Internet Things J.*, vol. 7, no. 3, pp. 1718–1735, Mar. 2020.
[34] J.-M. Fernandez, I. Vidal, and F. Valera, "Enabling the orchestration of IoT slices through edge and cloud microservice platforms," *Sensors*, vol. 19, no. 13, p. 2980, Jul. 2019.
[35] H. Cao, M. Wachowicz, C. Renso, and E. Carlini, "Analytics everywhere: Generating insights from the Internet of Things," *IEEE Access*, vol. 7, pp. 71749–71769, 2019.
[36] S. Taherizadeh, V. Stankovski, and M. Grobelnik, "A capillary computing architecture for dynamic Internet of Things: Orchestration of microservices from edge devices to fog and cloud providers," *Sensors*, vol. 18, no. 9, p. 2938, Sep. 2018.
[37] M. A. Sharkh and M. Kalil, "A quest for optimizing the data processing decision for cloud-fog hybrid environments," in *Proc. Int. Conf. Commun. Workshops (ICC Workshops)*, May 2018, pp. 1–6.
[38] N. Maleki, M. Loni, M. Daneshtalab, M. Conti, and H. Fotouhi, "SoFA: A spark-oriented fog architecture," in *Proc. IECON-45th Annu. Conf. IEEE Ind. Electron. Soc.*, Oct. 2019, pp. 2792–2799.
[39] H. B. Abdallah, T. Abdellatif, and F. Chekir, "Hydle: A deployment framework for efficient IoT surveillance systems," in *Proc. IEEE 27th Int. Conf. Enabling Technol. Infrastruct. Collaborative Enterprises (WETICE)*, Jun. 2018, pp. 60–65.
[40] S. Yangui, P. Ravindran, O. Bibani, R. H. Glitho, N. B. Hadj-Alouane, M. J. Morrow, and P. A. Polakos, "A platform as-a-service for hybrid cloud/fog environments," in *Proc. IEEE Int. Symp. Local Metrop. Area Netw. (LANMAN)*, Jun. 2016, pp. 1–7.
[41] O. Bibani, S. Yangui, R. H. Glitho, W. Gaaloul, N. B. Hadj-Alouane, M. J. Morrow, and P. A. Polakos, "A demo of a PaaS for IoT applications provisioning in hybrid cloud/fog environment," in *Proc. IEEE Int. Symp. Local Metrop. Area Netw. (LANMAN)*, Jun. 2016, pp. 1–2.
[42] H. Fanaee-T, M. D. B. Oliveira, J. Gama, S. Malinowski, and R. Morla, "Event and anomaly detection using tucker3 decomposition," 2014, *arXiv:1406.3266*. [Online]. Available: http://arxiv.org/abs/1406.3266
[43] R. Mayer, B. Koldehofe, and K. Rothermel, "Predictable low-latency event detection with parallel complex event processing," *IEEE Internet Things J.*, vol. 2, no. 4, pp. 274–286, Aug. 2015.
[44] H. Khazaei, R. Veleda, M. Litoiu, and A. Tizghadam, "Realtime big data analytics for event detection in highways," in *Proc. IEEE 3rd World Forum Internet Things (WF-IoT)*, Dec. 2016, pp. 472–477.
[45] I. Cramer, P. Govindarajan, M. Martin, A. Savinov, A. Shekhawat, A. Staerk, and A. Thirugnana, "Detecting anomalies in device event data in the IoT," in *Proc. 3rd Int. Conf. Internet Things, Big Data Secur.*, 2018, pp. 52–62.
[46] L. Lyu, J. Jin, S. Rajasegarar, X. He, and M. Palaniswami, "Fog-empowered anomaly detection in IoT using hyperellipsoidal clustering," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1174–1184, Oct. 2017.
[47] G. Thamilarasu and S. Chawla, "Towards deep-learning-driven intrusion detection for the Internet of Things," *Sensors*, vol. 19, no. 9, p. 1977, Apr. 2019.
[48] F. Hosseinpour, P. V. Amoli, J. Plosila, and T. Hamalainen, "An intrusion detection system for fog computing and IoT based logistic systems using a smart data approach," *Int. J. Digit. Content Technol. Appl.*, vol. 10, no. 5, pp. 34–46, 2016.
[49] S. Raza, L. Wallgren, and T. Voigt, "SVELTE: Real-time intrusion detection in the Internet of Things," *Ad Hoc Netw.*, vol. 11, no. 8, pp. 2661–2674, Nov. 2013.
[50] R. A. A. Habeeb, F. Nasaruddin, A. Gani, I. A. T. Hashem, E. Ahmed, and M. Imran, "Real-time big data processing for anomaly detection: A survey," *Int. J. Inf. Manage.*, vol. 45, pp. 289–307, Apr. 2019.
[51] L. Rettig, M. Khayati, P. Cudré-Mauroux, and M. Piorkówski, "Online anomaly detection over big data streams," in *Applied Data Science*, vol. 45. Cham, Switzerland: Springer, 2019, pp. 289–312.
[52] M. Zhang, C. Chen, T. Wo, T. Xie, M. Z. A. Bhuiyan, and X. Lin, "SafeDrive: Online driving anomaly detection from large-scale vehicle data," *IEEE Trans. Ind. Informat.*, vol. 13, no. 4, pp. 2087–2096, Aug. 2017.
[53] A. Yahyaoui, T. Abdellatif, and R. Attia, "READ: Reliable event and anomaly detection system in wireless sensor networks," in *Proc. IEEE 27th Int. Conf. Enabling Technol., Infrastruct. Collaborative Enterprises (WETICE)*, Jun. 2018, pp. 193–198.
[54] A. Yahyaoui, T. Abdellatif, and R. Attia, "Hierarchical anomaly based intrusion detection and localization in IoT," in *Proc. 15th Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, Jun. 2019, pp. 108–113.
[55] J. Lopez, R. Roman, I. Agudo, and C. Fernandez-Gago, "Trust management systems for wireless sensor networks: Best practices," *Comput. Commun.*, vol. 33, no. 9, pp. 1086–1093, Jun. 2010.
[56] X. Li, F. Zhou, and J. Du, "LDTS: A lightweight and dependable trust system for clustered wireless sensor networks," *IEEE Trans. Inf. Forensics Security*, vol. 8, no. 6, pp. 924–935, Jun. 2013.
[57] T. Abdellatif, K. Rouis, and M. Mosbah, "Adaptable monitoring for intrusion detection in wireless sensor networks," in *Proc. IEEE 26th Int. Conf. Enabling Technol., Infrastruct. Collaborative Enterprises (WETICE)*, Jun. 2017, pp. 54–59.
[58] C. Pahl, N. E. Ioini, S. Helmer, and B. Lee, "An architecture pattern for trusted orchestration in IoT edge clouds," in *Proc. 3rd Int. Conf. Fog Mobile Edge Comput. (FMEC)*, Apr. 2018, pp. 63–70.
[59] M. Jacobs and P. Leydekkers, "Specification of synchronization in multimedia conferencing services using the TINA lifecycle model," *Distrib. Syst. Eng.*, vol. 3, no. 3, pp. 185–196, Sep. 1996.
[60] B. I. Ismail, E. M. Goortani, M. B. A. Karim, W. Ming Tat, S. Setapa, J. Y. Luke, and O. Hong Hoe, "Evaluation of docker as edge computing platform," in *Proc. IEEE Conf. Open Syst. (ICOS)*, Aug. 2015, pp. 130–135.
[61] S. Abolfazli, Z. Sanaei, E. Ahmed, A. Gani, and R. Buyya, "Cloud-based augmentation for mobile devices: Motivation, taxonomies, and open challenges," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 337–368, 1st Quart., 2014.
[62] A. Abduvaliyev, S. Lee, and Y.-K. Lee, "Energy efficient hybrid intrusion detection system for wireless sensor networks," in *Proc. Int. Conf. Electron. Inf. Eng.*, Aug. 2010, pp. 383–396.
[63] S.-S. Wang, K.-Q. Yan, S.-C. Wang, and C.-W. Liu, "An integrated intrusion detection system for cluster-based wireless sensor networks," *Expert Syst. Appl.*, vol. 38, no. 12, pp. 15234–15243, Nov. 2011.

[64] A. Jøsang and R. Ismail, "The beta reputation system," in *Proc. 15th Bled Electron. Commerce Conf.*, 2002, pp. 2502–2511.
[65] M. Razzaq, G.-R. Kwon, and S. Shin, "Energy efficient Dijkstra-based weighted sum minimization routing protocol for WSN," in *Proc. 3rd Int. Conf. Fog Mobile Edge Comput. (FMEC)*, Apr. 2018, pp. 246–251.
[66] A. Rafi, A. Ur Rehman, G. Ali, and J. Akram, "Efficient energy utilization in fog computing based wireless sensor networks," in *Proc. 2nd Int. Conf. Comput., Math. Eng. Technol. (iCoMET)*, Jan. 2019, pp. 1–5.
[67] *NSL-KDD Data*. Accessed: Jun. 19, 2020. [Online]. Available: https://www.unb.ca/cic/datasets/nsl.html
[68] J. Liu, B. Kantarci, and C. Adams, "Machine learning-driven intrusion detection for Contiki-NG-based IoT networks exposed to NSL-KDD dataset," in *Proc. 2nd ACM Workshop Wireless Secur. Mach. Learn.*, Jul. 2020, pp. 497–511.
[69] S. Sharma, Y. Gigras, R. Chhikara, and A. Dhull, "Analysis of NSL KDD dataset using classification algorithms for intrusion detection system," *Recent Patents Eng.*, vol. 13, no. 2, pp. 142–147, May 2019.
[70] S. S. Panwar and Y. Raiwani, "Performance analysis of NSL-KDD dataset using classification algorithms with different feature selection algorithms and supervised filter discretization," in *Intelligent Communication, Control and Devices*. Singapore: Springer, 2020, pp. 497–511.
[71] L. Huang, X. Feng, C. Zhang, L. Qian, and Y. Wu, "Deep reinforcement learning-based joint task offloading and bandwidth allocation for multi-user mobile edge computing," *Digit. Commun. Netw.*, vol. 5, no. 1, pp. 10–17, Feb. 2019.
[72] X. Liu, Z. Qin, and Y. Gao, "Resource allocation for edge computing in IoT networks via reinforcement learning," in *Proc. ICC-IEEE Int. Conf. Commun. (ICC)*, May 2019, pp. 1–6.

AYMEN YAHYAOUI received the degree in computer science engineering from the Aviation School of Borj El Amri, in 2005, the M.S. degree in information system and network security from the High Institute of Computer Science, in 2007, the M.S. degree in modeling from the High Institute of Management, in 2009, and the M.S. degree in computer science from the Naval Postgraduate School, USA, in 2014. He is currently pursuing the Ph.D. degree with the Polytechnic School of Tunisia. He is also an Assistant Teacher with the Tunisian Military Academy of Fondouk Jedid. He works mainly on anomaly and intrusion detection in the IoT. He is also a member of the EPT/SERCOM Research Team.

TAKOUA ABDELLATIF received the degree IT engineering from the ENSIMAG (Grenoble) Engineering School, in 1998, and the Ph.D. degree in collaboration with the INRIA on Automating Management of JavaEE clusters, Bull/Grenoble, in 2006. She worked as a Research and Development Engineer with Hewlett Packard in Grenoble on designing and improving Telecom and Internet convergence, during five years. She then worked as a Research Engineer in Bull/Grenoble. She was during 13 years an Associate Professor in IT with Sousse University and with the Polytechnics School of Tunisia (EPT), Carthage University. She is currently a Senior Researcher with the EPT/SERCOM Research Laboratory, coordinating the lab research activities around scalable and secure distributed systems.

SAMI YANGUI received the M.Sc. degree in computer science from the University of Tunis El-Manar, Tunisia, in 2010, and the Ph.D. degree in computer science from the Telecom SudParis, Institut Mines-Telecom, France, in 2014. He is currently an Associate Professor with the Institut National des Sciences Appliquées (INSA), Toulouse, France. He is a member of the CNRS LAAS Research Team. He is also working on different aspects related to these topics, such as cloud/fog computing, network functions virtualization, and content delivery networks. He is also involved in different European and International projects, as well as, standardization efforts. He published several scientific articles in high-ranked conferences and journals in his field of research. His research interests include distributed systems and architectures, service-oriented computing, and the Internet of Things. He also served on many program and organization committees of International conferences and workshops.

RABAH ATTIA received the Ph.D. degree in telecommunications from the University of Valenciennes, France, in 1986. Since 2013, he has been the Director of the EPT/SERCOM Laboratory, Tunisia. He is currently a Professor of optical communication with the Polytechnic School of Tunisia. His research interests include new-generation of optical fiber, photonic crystal component, electro-optic modulators, and cooperative/relay networks.