Journal of Parallel and Distributed Computing 128 (2019) 1–14

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

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

# An efficient scheme for applying software updates in pervasive computing applications

Kostas Kolomvatsos

*School of Computing Science, University of Glasgow, Lilybank Gardens, G12 8RZ, Glasgow, UK*

## HIGHLIGHTS

- We propose an time optimized, performance-aware scheme for software updates management.
- The mechanism decides when an update process should be activated.
- We offer a method for deriving the optimal time for initiating the update process.
- With the proposed mechanism, IoT nodes enjoy the best possible performance.
- We offer extensive simulations and an analysis of the results.

## ARTICLE INFO

*Article history:*
Received 10 December 2016
Received in revised form 20 January 2019
Accepted 23 January 2019
Available online 1 February 2019

*Keywords:*
Pervasive computing
Internet of Things
Updates management
Optimal stopping theory

## ABSTRACT

The *Internet of Things* (IoT) offers a vast infrastructure of numerous interconnected devices capable of communicating and exchanging data. Pervasive computing applications can be formulated on top of the IoT involving nodes that can interact with their environment and perform various processing tasks. Any task is part of intelligent services executed in nodes or the back end infrastructure for supporting end users' applications. In this setting, one can identify the need for applying updates in the software/firmware of the autonomous nodes. Updates are extensions or patches significant for the efficient functioning of nodes. Legacy methodologies deal with centralized approaches where complex protocols are adopted to support the distribution of the updates in the entire network. In this paper, we depart from the relevant literature and propose a distributed model where each node is responsible to, independently, initiate and conclude the update process. Nodes monitor a set of metrics related to their load and the performance of the network and through a *time-optimized scheme* identify the appropriate time to conclude the update process. We report on an *infinite horizon optimal stopping model* on top of the collected performance data. The aim is to make nodes capable of identifying when their performance and the performance of the network are of high quality to efficiently conclude the update process. We provide specific formulations and the analysis of the problem while extensive simulations and a comparison assessment reveal the advantages of the proposed solution.



## 1. Introduction

### 1.1. Motivation

The rapid evolution of the Internet of Things (IoT) in combination with pervasive computing sets new challenges for the development of novel services and applications. The adoption of wireless technologies (e.g., *Wireless Sensors Networks*—WSNs) and Internet accompanied by the respective hardware gives the opportunity to numerous nodes to be interconnected. The transition from closed networks to interconnected autonomous nodes capable of interacting with their environment and performing simple processing tasks increases the quality of services that end users enjoy. The basic concept of adopting many autonomous devices is the pervasive presence around end users of various technologies such as sensors, actuators, smartphones and so on [5]. IoT nodes have, usually, different characteristics and capabilities that pose a set of requirements for any novel application. For instance, communications should be realized on top of specific rules adopted for the exchange of the collected data while intelligent interfaces may assist in the efficient management of the observed heterogeneity.

IoT nodes can collect and process ambient data in very dynamic environments. They come with the software/firmware of the corresponding manufacturers. Two main issues may positively affect their performance: (i) the application of software updates extending their functionalities; (ii) the application of firmware updates covering any potential gaps. As IoT nodes could manage sensitive user data, it is imperative to apply the discussed updates

*E-mail address:* kostas.kolomvatsos@glasgow.ac.uk.

https://doi.org/10.1016/j.jpdc.2019.01.010
0743-7315/

during their functioning. The challenge is to apply these updates in the entire network. Timing is everything, thus, the application of the updates should not be delayed. Once nodes are in operation, they will start receiving updates that should be applied as soon as possible. However, every node performs some tasks significant for supporting applications, thus, applying updates should not disturb them from their initial goal. Updates must be delivered in a way that conserves the limited bandwidth and intermittent connectivity and eliminates the possibility of compromising functional safety.

Various models have been proposed for the application of updates in WSNs [27,28,30,37,38,40,42,49,52–54,60,61,66]. To the best of our knowledge, the vast majority of them deal with centralized systems. A central server is responsible to define the steps for distributing the updates based on a delivery protocol. Apart from the simplicity of the solution (updates are located in a single point and transferred to all nodes at the same time), one can identify a number of disadvantages in the centralized approach: (i) the central server is the main responsible for delivering the updates, thus, it should apply complex protocols to distribute them without affecting the performance of the network; (ii) the central server cannot be aware of nodes being connected in the network, thus, it cannot be aware if all nodes have received the updates; (iii) when the central server distributes the updates, nodes should interrupt their processing tasks to apply them; (iv) the network is flooded by update distribution messages limiting the available bandwidth and reducing the performance of the network.

In this paper, we propose a distributed model for applying updates to avoid the aforementioned disadvantages. Through our approach, the central server should not be aware of the status of each node and the network, thus, it acts as a 'repository'. Nodes undertake the responsibility of downloading and applying the updates when they observe that it is the appropriate time to do it. Nodes' line of actions is realized by our *time-optimized, performance-aware mechanism* defined by means of the *Optimal Stopping Theory* (OST). The proposed scheme aims to intelligently support nodes and provide a decision making model that finds the appropriate time for initiating and concluding the update process. We focus on an *infinite horizon scheme* with each node having no deadline for concluding the process. In such scenarios, updates are considered as 'non critical' (e.g., software extensions), however, they should be applied as soon as possible as they positively affect nodes' performance. We consider a reward function and a discount factor for each time step where nodes delay the process, thus, we try to 'force' them towards its immediate conclusion. The reward function involves multiple metrics monitored by nodes that depict their load and the performance of the network.

### 1.2. Our idea, contribution & paper organization

We propose an update scheme building on the autonomous nature of nodes, i.e., an autonomous entity capable of performing lightweight processing on top of collected data. Nodes are capable of collecting information related to the performance of the network. Specific metrics could be taken into account like the *bandwidth*, the *error rate* and so on. Our scheme is distributed i.e., each node autonomously decides ***when*** to download and apply software updates (from this point forward we refer to 'updates' to depict software/firmware updates, extensions and patches).

Legacy systems, as we discuss in Section 2, deal with centralized schemes. The most common model in such systems is broadcasting, thus, one of the key challenges is the messaging overhead [39] and [65]. Nodes rebroadcast any new incoming data packet increasing the number of the unnecessary transmissions. For instance, if a software update of $x$ packets is to be sent over a network of $y$ nodes, potentially $x \cdot y$ times $y$ broadcast packets could be sent out [39]. This may have a negative effect on the performance of the network, i.e., a high number of broadcasts could lead to more cumulative power that should be consumed to support communications. In addition, the increased messaging overhead could introduce more collisions, thus, it may affect the reliability of the channel. Due to the increased size or the multiple assignments of the updates, a set of techniques has been proposed to alleviate the burden of multiple broadcasted messages. Incremental updates, compression, diffusion and various dissemination protocols are adopted to find the appropriate means for delivering the updates. However, all these techniques have specific drawbacks. Incremental updates require an efficient management mechanism to maintain the updates history while compression and, accordingly, decompression require a complex management. The additional overhead for the management of incremental and compression/decompression schemes becomes significant limiting the central system's and nodes' performance. Incremental techniques should also incorporate mechanisms for handling the heterogeneity of nodes while diffusion mechanisms require an increased number of messages especially in dense networks. In any case, the design of a reliable dissemination protocol that could manage the heterogeneity of nodes is a real challenge. Heterogeneity makes difficult the application of a protocol that is capable of covering all the types of nodes.

Various operating systems like Contiki [13], LiteOS [8] and RETOS [9] already support dynamic linking and updates loading, however, the adopted *ELF* format (Contiki adopts the format) leads to binaries potentially large for transfer [62]. In these cases, the minimization of the disseminated messages [54] can reduce the load of the network. In general, legacy dissemination protocols assume an ad-hoc network and use a form of controlled flooding [6]. Epidemic approaches combined with scalable multicasting through which updates are periodically transferred to nodes are another approach (e.g., [28,42,45]). Apart from the increased number of messages, one can observe an increased complexity in the resources required for maintaining data related to the management of the updates (e.g., [28,45,53,61] ). These data are related to the patterns of the updates, their advertisement, the combination of their parts and so on. There is a trade off between the overhead in the management of the updates compared to the time for the final conclusion.

The aforementioned problems of the centralized techniques combined with: **(i)** the absence of knowledge about if nodes are connected to the network during the dissemination, **(ii)** the absence of knowledge about if nodes have the availability to receive and conclude the updates at the time of the broadcasting, and **(iii)** the complexity of the presented dissemination protocols, consists of the motivation behind the current work. The proposed approach can be applied in any network involving autonomous devices, i.e., IoT and WSNs. It should be noted that we do not aim to provide a methodology on **how** updates will be installed but on **when** updates should be retrieved and installed towards the maximization of the performance of the network and nodes. The computational and storage complexity of our model is minimal, thus, it can be easily executed by devices with limited capabilities. The novelty compared to the state of the art is that our scheme does not 'impose' the application of updates and leaves nodes to independently decide their course of actions. Hence, we can avoid all the negative consequences of centralized systems for the dissemination, the recreation and the application of the updates locally.

Every node monitors its current load (e.g., the tasks waiting for execution) and the status of the network. We should discern the tasks to: (i) *regular tasks* (e.g., temperature/humidity monitoring, complex events reasoning); (ii) *tasks related to the application of the updates*. Nodes, when applying updates, cannot work efficiently till

the process is finished, especially, when the size is high. This time is critical as nodes should continuously operate and deliver their results. Our time-optimized scheme derives **when** the updates could be applied based on the current status (e.g., current load, current status of the network) and the estimation of the future status (i.e., the expectation of the adopted parameters). The central system sends to nodes only a lightweight message containing the indication about the presence of an update. We consider two schemes: (i) the server defines a deadline for the application of the update; (ii) the server does not impose a deadline, however, the sooner the update is applied, the better for nodes' performance. The following list reports on the benefits of the proposed solution compared to the remaining research efforts in the domain:

- the central server should not be aware of the status of each node;
- the central system should not spend resources to handle the heterogeneity of nodes;
- the proposed scheme alleviates the complexity of the central system's behaviour as it is not necessary to define and adopt complex protocols and dissemination schemes;
- each node independently runs the proposed scheme and selects the appropriate time for applying the update;
- the proposed model takes into consideration the dynamic nature of nodes, i.e., nodes initiate the process by themselves, thus, the application of the updates is secured;
- the proposed model is fully adapted on the performance of the network securing the uninterrupted application of the updates. Nodes will decide to initiate the process only when they 'see' that the network's and their performance are of high quality.

The contribution of our work is summarized as follows:

- a novel *time-optimized, performance-aware mechanism* based on OST, which decides when an *update process*, $\mathcal{P}$, should be activated. As mentioned, the proposed model does not deal with **how** $\mathcal{P}$ should be installed but **when** $\mathcal{P}$ should be concluded towards the maximization of the performance of the network and the node;
- a method for delivering the optimal time $t^*$ for initiating $\mathcal{P}$. $\mathcal{P}$ consists of two stages: (i) get the software from the central system; (ii) install the component locally. At $t^*$, the node enjoys the best possible performance to support $\mathcal{P}$. As the best performance, we define the highest possible network performance and the limited load of nodes;
- a method that alleviates the overhead required by legacy systems to disseminate and maintain the updates. The overhead is related to messaging as well as the required resources for combining parts of the updates and their patterns;
- a mechanism that is characterized by a lower complexity than legacy systems giving to nodes the room for acting autonomously and defining their own scheduling for $\mathcal{P}$;
- a model that requires only light weight advertisement messages compared to legacy systems that require additional messaging for controlled flooding or multicasting parts of $\mathcal{P}$;
- extensive simulations and an analysis of the results where we reveal the strengths of the proposed model and compare it with baseline solutions and a centralized approach.

The paper is organized as follows. Section 2 reports on the related work. In Section 3, we describe and formulate the problem under consideration. Section 4 presents the analysis of the problem and provides specific formulations for its solution. Simulations reveal the performance of the proposed model in Section 5. Finally, in Section 6, we conclude our paper by presenting our future research agenda.

## 2. Related work

The pervasive nature of the IoT is enhanced by the presence of sensors either in the 'standalone' mode or embedded into users' smart devices. An IoT node consists of two parts: (i) the *hardware* that makes the device capable of recording/collecting/sending data from/to the environment, and, (ii) the *software* that makes the node capable of processing the collected data and producing some outcomes. The produced knowledge could be adopted to deliver decisions related to the presence of events and the provision of the appropriate responses. In this section, we shortly review the IoT domain by presenting the taxonomies of the devices, the communication schemes as well as the envisioned applications. Finally, we report on schemes for delivering software updates in IoT nodes.

Specific taxonomies have been already defined for describing the components required by the IoT vision [7,63,64]. There are three components that enable the pervasive computing aspect [21]: (i) the required hardware i.e., sensors, actuators and communication hardware; (ii) the required middleware i.e., functionalities related to storage, processing and support for intelligent analytics; (iii) the required presentation i.e., tools that support visualization and interpretation. Hardware defines a number of requirements related to power, connectivity and security that should be handled to provide a node ready to support high quality applications. For instance, a device cannot execute energy consuming tasks when the underlying hardware poses strict constraints in the energy consumption. In addition, when a node has limited power or computational capabilities, it cannot be used in applications that require the continuous presence of the node in its environment. For communication purposes, the technology that is widely adopted is RFID. RFIDs help in the automatic identification of anything where they are attached and act as electronic bar codes [31,69]. In addition, the WSN technology leads to the adoption of low power, miniaturized nodes capable of supporting remote sensing applications. Sensors enable the collection, processing, analysis and dissemination of the observed information gathered in a variety of environments [3]. Sensors can easily support both, distributed or centralized applications and, especially, the provision of the basis for deriving intelligent analytics. Nodes communication capabilities offer functionalities for connecting them to e.g., the Cloud to invoke specific services or to fire more complex processing.

Smart IoT nodes are based on the appropriate middleware to be capable of performing the required functionalities. The middleware is a mechanism to combine cyber infrastructure with *Service Oriented Architecture* (SOA) and WSNs to have access on heterogeneous sensor resources [18]. The management of heterogeneity, accompanied by a platform-independent middleware, is of high importance in future applications. The *Open Sensor Web Architecture* (OSWA) [58] offers a set of operations and standard data formats to 'cover' the retrieved data and provide a uniform view on the sensors results. The middleware connects different, complex, new or existing software that are not designed to be connected. The architectural aspects of middleware, the requirements and the available methodologies have been already discussed in various research efforts. For surveys on the IoT middleware, the interested reader could refer in [26,33,50,67].

Concerning the application domains, those can be classified according to the type of networks, coverage, scalability management, heterogeneity, users involvement and impact [19]. Four categories of the IoT application domains are identified i.e., [21]: (i) *Personal and home applications*. Intelligent applications are delivered on top of the collected data fully adapted to users' characteristics and the dynamics of the environment; (ii) *Enterprise applications*. The information is collected by enterprise networks and

intelligent applications are delivered in domains like environmental monitoring [1,20,23,24,35,36], smart IoT environment [19,44], etc.; (iii) *Utilities*. They involve intelligent applications on top of information retrieved by networks adopted to produce solutions for service optimization. Typical examples are the Smart Grid and smart energy management applications like those presented in [14,16,47,55]; (iv) *Mobile applications*. They are built on top of the information conveyed by mobile nodes. A smart transportation system [4,10,32,71], is the typical representative of such applications.

Software updates for IoT nodes are necessary after deployment to have the nodes capable of efficiently performing the assigned tasks. A survey on the adopted methodologies is presented in [6]. Firmware updates are necessary in non-modular sensor operating systems [41]. Apart from firmware, generic re-programming (i.e., software components that are not directly related to the firmware) is another problem that aims to extend or correct devices' functionalities. Re-programming is related to updates in the software components that aim to provide extensions i.e., new functionalities or solve possible errors. As mentioned, the envisioned algorithms target to minimize the time required for applying any update through the minimization of the amount of data that will be transferred to nodes. Incremental updates and data compression could be adopted to reduce the size of messages [62]. However, splitting the data in multiple parts requires an increased number of messages for concluding the whole process. When only the differences with the previous version of the software are sent, a mechanism that combines the new version with the old one without jeopardizing the functioning of nodes is necessary. The incremental management of the updates does not eliminate the necessary process for maintaining updates history and the aforementioned combination.

Data dissemination protocols are the key part of a dissemination strategy for transferring software updates to any member of a WSN. Some protocol examples are discussed in [27,30,37,40,52,60,66]. The use of a controlled flooding differs from the data collection protocols in storage, coverage and data flow. However, controlled flooding does not eliminate the need for an increased number of messages that will be distributed in the network, especially, in dense networks. In WSNs, resources are limited and nodes cannot cache overheard packets which might not be useful [27]. The normal pattern for dissemination protocols is a three-step process [6]: (i) the advertisement of available software; (ii) the selection of a source; and (iii) the reliable downloading to the target. A subscription approach could be adopted, however, this results in a significant overhead in the network and, more specifically, in the server where the updates are present.

Some widely cited research efforts in the domain are as follows. Trickle [42] is an algorithm that disseminates and, accordingly, maintains software updates in WSNs. Trickle adopts an epidemic approach with scalable multicasting through which updates are periodically transferred. Epidemic approaches, in general, may involve the transmission of several copies to random nodes, thus, there is an increased cost for the management of the received messages. In addition, there is no guarantee that nodes will be always connected to the network to receive the envisioned messages. Special attention is paid by the data discovery and the *Dissemination Protocol* (DIP) on the elements that could be exchanged between nodes [46]. The protocol tries to randomly scan the network and detect new items while maintaining the latency at low levels. DHV [12] is an efficient code consistency maintenance protocol that ensures that nodes will, eventually, have the same code. The *Multicast-based Code redistribution Protocol* (MCP) [45] is another protocol that performs code maintenance. MCP 'forces' every node to maintain a table that contains the information of known applications. The table supports the delivery of multicast-based code dissemination requests. In any case, the use of additional data structures increase the storage complexity of the corresponding models. *Multi-hop, Over-the-Air code distribution Protocol* (MOAP) [61] uses a store-and-forward approach providing a pattern of updates. The updated code is broadcasted in a neighbour-per-neighbour basis forcing nodes to disseminate the incoming code to reduce the latency. Deluge [28] is a protocol that builds on top of algorithms related to density-aware, epidemic maintenance protocols and includes several optimizations. It adopts Trickle for the advertisement of code and splits the code into a set of fixed-size pages. Through this approach, the time required for the propagation of large components is reduced. In any case, the adoption of multiple optimizations increases the complexity of the proposed solution especially for the recreation of the updates from multiple parts. Stream [53] adopts Deluge and optimizes the code parts sent in the network. Stream deals with pre-installing in each node the re-programming application. Hence, Stream transmits the minimal support (about one page) needed for the activation of the re-programming image. Resource-awareness, time-efficiency, and the integration of security solutions are involved in the model presented in [48]. A multi-hop propagation scheme is proposed enhanced by security codes and means from fuzzy control theory. In any case, the definition of fuzzy logic rules that cover all the aspects of real scenarios is very difficult. MELETE [73] is another code dissemination protocol designed to support multiple concurrent applications in a WSN. It assumes that the network is a set of groups of nodes that execute different tasks. The framework adopts a group-keyed model to selectively distribute the code to only the interested nodes, and reactively distribute the code only when it is required.

A monitoring process (like in our model) over various parameters before performing a set of actions is adopted in various domains. In [57], the authors describe a distributed scheduler that opportunistically schedules data transmissions, with a view of minimizing the energy consumption of a wireless device. By exploiting the stochastic characteristics of the channel, the model postpones the communication up to an acceptable time deadline until it finds the best expected channel conditions. In [51], the authors study Device-to-Device (D2D) communications and demonstrate the energy, capacity and Quality of Service (QoS) benefits of link-aware opportunistic D2D communications. In [11], the authors study the energy efficiency of channel-aware random access with multiple parallel channels under a collision channel. An asymptotic relationship between the energy efficiency and the total bandwidth is described, which shows that the relationship depends on the energy consumption properties of sensors. In [17], the authors focus on the Distributed Opportunistic Scheduling (DOS) model that exploits multiuser diversity in wireless networks without the requirement of a central scheduler. With DOS, users take their own scheduling decisions based on local observations related to the channel. In [68], the authors propose an online scheduling algorithm designed to decide the optimal action in each time slot (i.e., to transmit or hold the packet on the top of the transmitter queue) based on the predicted channel condition and the packet queue status. Finally, [43] reports on a model that builds on top of the prediction of the channel SNR to schedule the desired transmissions.

## 3. Problem statement

In this section, we discuss the problem under consideration and present basic information about our model. In Table 1, we provide the basic notation adopted throughout the paper.

**Table 1**
Nomenclature.

| Notation | Description |
|---|---|
| $\mathcal{P}$ | The update process |
| $\mathcal{S}$ | The update server |
| $\mathcal{N}$ | The set of nodes ($n_i$, $i = 1, 2, \ldots, \lvert\mathcal{N}\rvert$) |
| $UE$ | The update epoch |
| $\mathcal{U}$ | The deadline for an update epoch |
| $\mathcal{M}$ | The set of the performance metrics ($m_k$, $k = 1, 2, \ldots, \lvert\mathcal{M}\rvert$) |
| $I_k$ | The function that defines if a metric is proportional or not |
| $r_t$ | The reward value at $t$ |
| $t^*$ | The optimal stopping time |
| $y_t$ | The maximum reward at $t$ |
| $z_t$ | The discounted maximum reward at $t$ |
| $\beta$ | The discount factor |
| $S^*$ | The stopping rule |
| $Z^*$ | The optimal reward |
| $\mathbb{T}$ | The discrete time domain |

### 3.1. Preliminaries

We envision a setting where a set of IoT nodes $\mathcal{N} = \{n_1, n_2, \ldots, n_{|\mathcal{N}|}\}$ performs specific processing tasks. These tasks are independent of each other even if some of the nodes could perform the same task (e.g., temperature or humidity monitoring). An *IoT node*, $n_i$, is a physical device embedded with electronics and software having capabilities of collecting, processing and exchanging of data. An *update process* $\mathcal{P}$ is an independent task that alters the software of a node. An update could be [62]: (i) an update of the operating system; (ii) an update of an application; (iii) an addition of a new application; (iv) a modification of parameters in an existing application. A server $\mathcal{S}$, a software component, is responsible to store, manage and disseminate the updates. We try to alleviate the load of $\mathcal{S}$ when serving a high number of nodes. When many nodes try to contact $\mathcal{S}$, download and install the available updates, bottlenecks could be present. In such cases, $\mathcal{S}$ should adopt an intelligent algorithm or be accompanied by powerful hardware to efficiently serve the increased load. In Fig. 1, we depict an example architecture of the envisioned model. In general, the execution of $\mathcal{P}$ is characterized by bursts allocated near the time of $\mathcal{P}$'s advertisements. For alleviating the problem and avoid possible bottlenecks, we propose a distributed scheme that each node adopts to be able to get and apply $\mathcal{P}$.

$\mathcal{P}$ (from each node perspective) is concluded after the reception of the corresponding $\mathcal{S}$'s advertisement message and consists of the following steps: (i) $n_i$ is connected to $\mathcal{S}$; (ii) $n_i$ downloads $\mathcal{P}$; (iii) $n_i$ applies $\mathcal{P}$, locally. We assume that $\mathcal{S}$, when $\mathcal{P}$ is available, sends a lightweight message (e.g., a single packet) for the presence of $\mathcal{P}$. Based on the criticality of $\mathcal{P}$, the message could be accompanied by a deadline. We define the *update epoch UE* as the time required to apply $\mathcal{P}$. *UE* is an interval $[1, \mathcal{U}]$ in which the entire set of nodes should have concluded $\mathcal{P}$. When $\mathcal{U} = \infty$, we consider the *infinite horizon* version of our scheme, otherwise, we deal with the *finite horizon* version of the problem. When no deadline is set, $n_i$ has unlimited time to conclude $\mathcal{P}$. In any case, $n_i$ should conclude $\mathcal{P}$ as soon as possible, however, under specific constraints related with $n_i$'s performance that secures the successful conclusion of $\mathcal{P}$. We focus on the *infinite horizon* version of the aforementioned problem. The finite horizon version is studied in [34] where we provide specific formulations and the solution of the problem through backward induction. The finite version 'suffers' from the need of meeting the pre-defined deadline, thus, nodes are forced to immediately conclude $\mathcal{P}$. This can affect the performance as multiple nodes may decide to initiate $\mathcal{P}$ at the same time. The identified research challenges are as follows:

- **Research Challenge 1.** Each node should monitor its performance and the status of the network to retrieve the corresponding data. These data will become the basis for the decision making related to the conclusion of $\mathcal{P}$. The challenge is to select the appropriate performance metrics and build the discussed mechanism.
- **Research Challenge 2.** Based on the collected performance data, for every *UE*, we should find the optimal stopping time $t^* \in [1, \mathcal{U}]$ where nodes will stop the monitoring process and initiate/conclude $\mathcal{P}$.

### 3.2. Update management optimization

A *performance metric* $m_k$ measures the activities and the performance of an entity. Let $\mathcal{M}$ be the set of the adopted metrics, i.e., $\mathcal{M} = \{m_k, k = 1, 2, \ldots, |\mathcal{M}|\}$ on top of which the decision for concluding $\mathcal{P}$ is made. For instance, when a node enjoys a high bandwidth and exhibits a low load, it can initiate $\mathcal{P}$. In such cases, the execution of $\mathcal{P}$ will not affect the performance of the node and it will not disturb the node from the execution of the assigned tasks.

In our model, we focus on two types of metrics related to (i) ***the network***, and, (ii) ***the performance of a node***. Network performance metrics can be categorized into [22]: (i) *availability*; (ii) *packet loss and error*; (iii) *delay*; (iv) *bandwidth*. On the one hand, nodes' performance can be easily obtained locally by measuring the tasks waiting for execution. On the other hand, the performance of the network can be obtained by specific tools/methodologies or through the use of operating systems' commands. For instance, in Linux one can retrieve the overall bandwidth with the use of nload, bmon, netload, etc commands applied on top of the kernel statistics. In an other example, Contiki supports the ping6 tool that is capable of reporting values related to the transmitted/received messages, the packet loss, etc. TinyOS supports the TOSSIM suite for getting MAC statistics. Other solutions involve theoretical models for e.g., the calculation of the available bandwidth [29,72] that refers to the maximum unused bandwidth at a link or end-to-end path. This depends on the link capacity and the traffic load during a certain time interval. The monitoring process of the aforementioned metrics adds an overhead, however, this overhead could be eliminated if data collection is characterized by a low frequency. Recent studies show that when the reporting frequency is low, the impact on the traffic performance increases [25]. The frequency of the monitoring process affects the throughput and the end-to-end delay. When the reporting frequency is set to a high value, i.e. 15 s, the impact is almost zero [25]. In this paper, we consider that nodes adopt a frequency that leads to almost zero overhead for network monitoring. An analytical study on the constraints of the network monitoring overhead is beyond the scope of the paper.

Consider the discrete time $\mathbb{T}$ with $t \in \mathbb{T}$. Let an advertisement message indicating a new update has been arrived in $n_i$. A new $UE_j$ starts and, at $t \in UE_j$, $n_i$ checks every $m_k$, $k = 1, 2, \ldots, |\mathcal{M}|$ and calculates the 'reward' that will gain if it initiates $\mathcal{P}$. The reward is based on the observed values and it is proportionally or inverse proportionally affected according to the type of $m_k$. For proportional metrics, the higher the value is, the higher the reward becomes (e.g., the bandwidth). For inverse proportional metrics, the lower the value is, the higher the reward becomes (e.g., the load of each node). We define the function $I_k$ that incorporates the information that $m_k$ is proportional or not:

$$I_k = \begin{cases} m_k & \text{if } m_k \text{ is a proportional metric} \\ 1 - m_k & \text{if } m_k \text{ is a non-proportional metric} \end{cases} \quad (1)$$

Based on $I_k$, $n_i$ calculates the reward $r_t$ as follows:

$$r_t = \frac{1}{|\mathcal{M}|} \sum_{k=1}^{|\mathcal{M}|} I_k \quad (2)$$

$r_t$ gives an indication about the current status of the node and the network. $r_t$ can be easily calculated and acts as a weighted

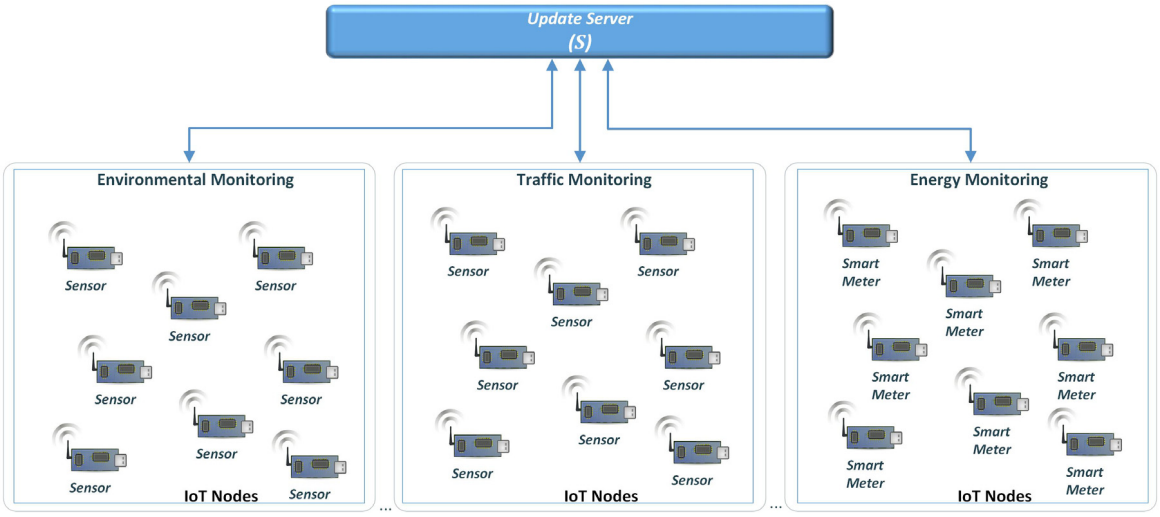



**Fig. 1.** An example architecture of the proposed model.

aggregation scheme where all metrics are of equal weight (the node pays equal attention on all of the observed metrics). $n_i$ tries to maximize $r_t$ and 'safely' conclude $\mathcal{P}$. Based on $r_t$, $n_i$ should decide when it is the appropriate time to initiate and conclude $\mathcal{P}$. Hence, at $t$, $n_i$ should take one of the following decisions:

- **D1**. Stop the monitoring process, initiate and conclude $\mathcal{P}$.
- **D2**. Continue the monitoring process without deviating from the current task fulfilment.

## 4. The time-optimized mechanism

### 4.1. Optimal stopping theory

The OST [56] could be adopted for determining the *best time* to take an action (decision) based on sequentially observed random variables. An optimal stopping problem is defined by the sequence of random variables $X_1, X_2, \ldots$ whose joint distribution is known and the sequence of real-valued reward functions $J_0, J(x_1), J(x_1, x_2), \ldots$. Let $(\Omega, B, P)$ be the probability space, and $\mathcal{G}_t$ be the sub-$\sigma$-field of $B$ generated by $X_1, \ldots, X_t$. We have the sequence of $\sigma$-fields $\mathcal{G}_1 \subset \mathcal{G}_2 \subset \ldots \mathcal{G}_t \subset B$. A stopping time is defined as a random variable $ST \in 0, 1, \ldots, \infty$ such that the event $ST = t$ is in $\mathcal{G}_t$. The aim is to choose an *optimal stopping time* $t^*$ to maximize the expected reward $\mathbb{E}[J_{t^*}]$. If there is no bound on the number of steps at which one has to stop, this is an *infinite horizon* problem and the optimal return can be calculated through the *optimality equation*. When there is a known upper bound on the number of steps, it is a *finite horizon* problem and the optimal return can be solved by *backward induction*.

### 4.2. Model analysis

In our infinite horizon problem, each node has not an upper time limit to conclude $\mathcal{P}$. At $t$, $n_i$ enjoys a 'disturbance' in the performance metrics depicted by the $m_k$ random values. We focus on independent metrics and do not consider any adaptation process, especially in the underlying network. The reward at $t$ is realized through a function $y_t = f_t(y_{t-1}, r_t), t \in UE_j, j = 1, 2, \ldots$. In our case, we adopt $y_t = max\{r_t\}$, as $n_i$ tries to find the maximum possible performance to conclude $\mathcal{P}$. Based on $y_t$, we consider the following reward function:

$$z_t(y_t, r_t) = \beta^t y_t. \tag{3}$$

It should be noted that no recall is permitted because $n_i$ cannot adopt any previous realizations of the performance metrics (node's and network's performance are dynamically updated). The discount factor $\beta \in (0, 1)$ affects $n_i$'s behaviour as follows. $n_i$ should delay the decision in the anticipation of a better $z_t$ when $\beta \to 1$. $n_i$ should not delay the decision when $\mathcal{S}$ requires an immediate conclusion of $\mathcal{P}$ ($\beta \to 0$). If $n_i$ never stops, the reward is considered equal to zero, thus, we assume $z_0 = z_\infty = -\infty$. The two main problems that should be solved are:

- Identify $t^*$ where the expected reward is maximized.
- Find an optimal stopping rule, such that $n_i$ terminates the monitoring process to maximize the expected reward $Z_t$, i.e., $\mathbb{E}[Z_t]$ ($Z$ is the random variable depicting the $n_i$'s reward).

We can treat the first problem as an infinite horizon problem where $n_i$ receives $z_t$ and it has no 'pressure' on the final decision. However, $\beta$ makes $n_i$ to conclude $\mathcal{P}$ in a 'rationale' time interval.

Once $n_i$ observes $z_t$, it decides whether to continue the process or not, by examining the expectation of the future reward without recall. In other words, $n_i$ has to find a $t^*$ where the supremum in Eq. (3) is attained i.e., $\sup_t \mathbb{E}[Z_t]$. Suppose that at some $t$, $n_i$ has observed $Z_t = z$ and it is optimal to continue the process. Then, at $t+1$, if $Z_{t+1}$ is still $z$, because $y_{t+1} \leq z$, it is optimal to continue due to the invariance of the problem in time [15]. Hence, based on the principle of optimality, this problem can be solved as an optimal stopping problem with discounted future reward and without recall. This means that the reward can be considered as $Z'_t = \beta^t Y_t = \beta^t max(R_t)$ and the problem assumes the same solution as the following problem: *Find a* $t^*$ *such that the* $\sup_t \mathbb{E}[Z'_t]$ *is attained.*

Consider the rewards $Z_0, Z_1, \ldots, Z_\infty$ where $Z_t = f(r_1, r_2, \ldots, r_t)$. The sequence $\langle r_t, \mathcal{F}_t \rangle$ is defined by a probability space $\Omega$, an increasing sequence of sub $\sigma$-algebras $\{\mathcal{F}_t\}_1^\infty$, the sequence of random variables $R_i$ ($r_i \in \mathcal{F}_t$) and $\mathbb{E}[r_i]$, $\forall i$. The following two assumptions should be true to have an optimal stopping time: **[A]** $\mathbb{E}[sup_t Z_t] < \infty$; **[B]** $lim\, sup_{t \to \infty} Z_t \leq Z_\infty$.

**Definition 1.** A stopping rule $S^*$ is the rule for which the following inequality holds true: $\mathbb{E}([Z_{S^*}]|\mathbb{F}_t) > Z_t$ a.s. on $\{S^* > t\}, \forall t$.

Ferguson proved the following theorems related to the existence of the optimal stopping rule $S^*$ [15], i.e.,

- **Theorem.** Assuming condition [A], for any stopping rule $S^*$, there is a regular stopping rule $S'$ such that $\mathbb{E}[Z_{S'}] \geq \mathbb{E}[Z_{S^*}]$.

- **Theorem.** Under [A] and [B] conditions, there exists a stopping rule $S^*$ such that $\mathbb{E}[Z_{S^*}] = M^*$ where $M^* = sup_S \mathbb{E}[Z_S]$.
- **Theorem.** Under [A], if an optimal stopping rule exists, in particular, if (B) holds true, then $S^*$ is optimal.

The interested reader could refer to [15] for more details.

**Theorem 1.** *For the model defined by Eq. (3), an optimal stopping time exists.*

**Proof.** We have $Z_t \leq \beta^t max\,(r_1, r_2, \ldots, r_t) \leq max\big(\beta r_1, \beta^2 r_2, \ldots, \beta^t r_t\big) \leq \sum_{j=1}^{\infty} \beta^j |r_j|$. Additionally, we have $\mathbb{E}[sup_t Z_t] \leq \sum_{j=1}^{\infty} \beta^j \mathbb{E}[|r_j|] = \frac{\beta}{1-\beta}\mathbb{E}[|r_j|] < \infty$. Based on the above, condition [A] is satisfied.

Furthermore, $lim\ sup_t Z_t \leq \lim_t \beta^t \sum_{j=1}^{t} |r_j| = \lim_t \beta^t \frac{\sum_{j=1}^{t} |r_j|}{t}$. From the law of large numbers, we get $\sum_{j=1}^{t} |r_j| \rightarrow \mathbb{E}[|r|]$ and $t\beta^t = 0$. Hence, $limsup_t Z_t \leq Z_\infty = 0$ and condition [B] is satisfied.

Based on the above, the optimal stopping rule and the optimal stopping time is given by the principle of optimality i.e., $t^* = \min\{t \geq 0 : Z_t \geq Z^*\}$. Hence, $Z^*$ is the expected return as defined by the optimal stopping rule. As discussed, the problem is invariant in time, thus, the principle of optimality will never require to recall a previous observation. Hence, the following equation holds true:

$$Z^* = \beta \mathbb{E}\left(max(Z_1, Z^*)\right). \tag{4}$$

### 4.3. Expected reward maximization

For simplicity in our calculations, we focus on proportional metrics in the interval [0,1]. Our model depends on the selection of the probability distribution for variables $R_t$ with realizations $r_t$. We rely on two probability distributions to provide a solution for our model, i.e., (a) the *Uniform* distribution and (b) the *Exponential* distribution. The selection of multiple distributions aims to cover as many real cases as we can. When applying the Uniform distribution, we assume that $r_t$s are of equal probability to be observed by $n_i$. By applying the Exponential distribution, we aim to focus on multiple scenarios where performance metrics are affected by the rate of the distribution. It should be noted that there is no reason to adopt a distribution 'favourite' to large values ($r_t \rightarrow 1$) as in these cases, the intelligent mechanism is useless.

By solving Eq. (4), we can get the reward limit $Z^*$ above which $n_i$ should stop the monitoring process and conclude $\mathcal{P}$.

**Lemma 1.** *If $m_k \sim U(0,1)$, with $k = 2$, the optimality equation $Z^* = \beta\left(Z^{*3} - \frac{Z^{*2}}{2} - \frac{Z^*}{2} + \frac{10}{6}\right)$, defines values that indicate the optimal stopping rule at $t$.*

**Proof.** We consider two cases: (i) $Z^* \leq 1$; (ii) $Z^* > 1$. In the former case (i.e., $Z^* \leq 1$), we should find the solution to the following equation: $\int_0^{Z^*} Z^* r dr + \int_{Z^*}^{1} r^2 dr + \int_1^2 r(2-r)dr$. By solving the integrals, we get $\beta\left(\frac{Z^{*3}}{6} + 2\right)$. In the latter case (i.e., $Z^* > 1$), we should find the solution to the following equation: $\int_0^1 Z^* r dr + \int_1^{Z^*} r(2-r)dr + \int_{Z^*}^2 r(2-r)dr$. By solving the integrals, we get $\beta\left(\frac{11}{6}Z^{*3} - Z^{*2} - Z^* + \frac{4}{3}\right)$. The two cases are of equal probability, thus, we get that $Z^* = \beta\left(Z^{*3} - \frac{Z^{*2}}{2} - \frac{Z^*}{2} + \frac{10}{6}\right)$.

**Lemma 2.** *If $m_k \sim Exp(\lambda_k)$, with $k = 2$, $R \sim Hypo(\lambda_1, \lambda_2, \ldots)$ [59], the optimality equation*

$$Z^* = \beta\left(\frac{\lambda_2}{\Lambda}\left(1 - e^{-\lambda_1 Z^*}\right) + \frac{\lambda_1}{\Lambda}\left(e^{-\lambda_2 Z^* - 1}\right) + \frac{\lambda_1\lambda_2}{\Lambda}\left(f(\lambda_1) - f(\lambda_2) + g(\lambda_2) - g(\lambda_1)\right)\right) \tag{5}$$

*with $\Lambda = \lambda_2 - \lambda_1$, $f(x) = \frac{e^{-x}(x+1)}{x}$ and $g(x) = \frac{e^{-Z^* x}(Z^* x + 1)}{x^2}$, indicates the optimal stopping rule at $t$.*

**Proof.** The pdf $h_R$ of the random variable $R$ i.e., a sum of exponentials with different rates, is given by [2]:

$$h_R(t) = \sum_{i=1}^{|\mathcal{M}|} \frac{\lambda_1 \ldots \lambda_{|\mathcal{M}|}}{\prod_{j=1, j\neq i}^{|\mathcal{M}|}} e^{\lambda_i t} I_{0,\infty}(t) \tag{6}$$

$\forall t \in \mathbb{T}$. By applying $h_R$ in Eq. (4) and through calculations, we conclude the recursive equation as the lemma indicates.

### 4.4. Estimation of the expected return

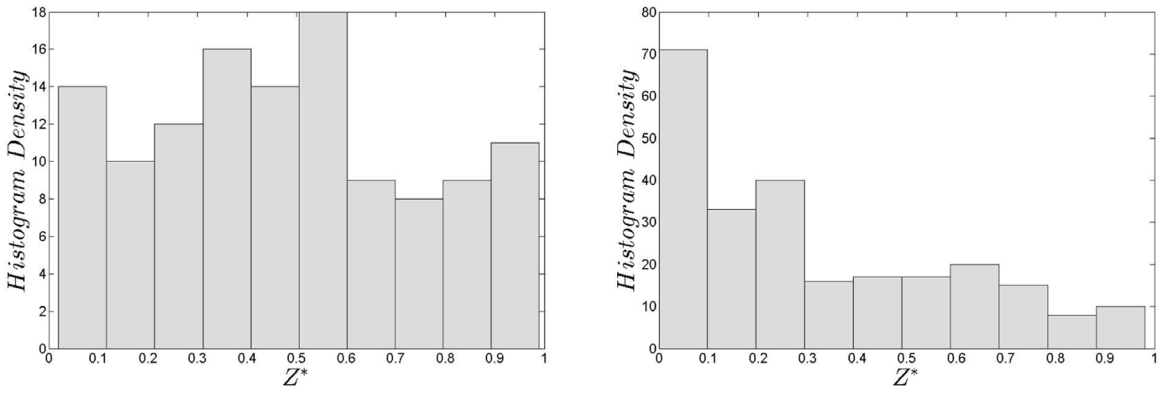
For evaluating the final value of $Z^*$, we adopt a Monte Carlo simulation. Monte Carlo methods (or Monte Carlo experiments) rely on repeated random sampling to obtain numerical results. In our case, we perform a large number of simulations to obtain the distribution of $Z^*$. In each simulation, we randomly generate the realizations of $\beta$. Accordingly, we record $Z^*$ that satisfies the equations provided by the aforementioned lemmas, thus, we can derive the final distribution. In Fig. 2, we present the histograms of the $Z^*$ distribution. It should be noted that mean values for $Z^*$ are 0.48 and 0.32 for the Uniform and the Exponential distribution, respectively.

## 5. Performance evaluation

### 5.1. Methodology and experimental setup

We elaborate on the performance of the proposed model. Through a high number of simulations, we evaluate the proposed **infinite horizon** *Optimal Stopping Scheme* (*OSS*). Various simulation scenarios are adopted to evaluate the performance of the *OSS* adopting the Uniform or the Exponential distribution for getting values for the adopted performance parameters. When the Uniform distribution is adopted, we aim to handle scenarios where $n_i$ monitors the system in an 'agnostic' manner. The term 'agnostic' means that $n_i$ considers that values for the performance metrics have an equal opportunity of occurring. When we adopt the Exponential distribution, we focus on high or low values depending on the rate $\lambda$. For instance, when $\lambda = 2.0$, $n_i$ monitors low values for the adopted parameters. The higher the $\lambda$ is, the smaller the values become. In this setting, we aim to evaluate different scenarios where the environment affects the performance metrics, thus, the decision of $n_i$. When $\lambda \rightarrow 1.0$, we assume that performance data exhibit low 'fluctuations' and $n_i$ could easily enjoy high performance values. When $\lambda$ is very high, we assume that the monitored performance exhibits 'heavy fluctuations' and $n_i$ cannot easily conclude high performance values.

$\mathcal{P}$, usually, involves large amounts of data, thus, for our study, we consider *the bandwidth* (let us define $b$ to depict the available bandwidth) as one of the most important network performance metrics. $b$ assesses the amount of data that a node can transfer through the network. Node performance parameters are related to its behaviour and load concerning the executed tasks (let us define the parameter $l$ to depict the current 'availability' of each node). When a node has a lot of tasks to execute, $l$ will be low. For simplicity, we do not focus on hardware related issues (e.g., memory size, CPU speed, storage used). However, our model can be easily extended to include more metrics into the decision scheme.




(a) *Uniform* (b) *Exponential*

**Fig. 2.** Histogram of the $Z^*$ distribution.

### 5.2. Evaluation metrics & simulation setup

We report on the performance of the *OSS* concerning two aspects: **(i)** the time required for deciding the initiation of $\mathcal{P}$; and **(ii)** the quality of the decision in terms of the network and node's performance (our model aims to identify the highest possible value for the observed metrics that secures the optimal performance). We report on the optimal stopping time $t^*$ and 'stopping' values of $b$ and $l$, i.e., $b^*, l^*$. The optimal performance is realized when $t^* \to 0$ and $b^*, l^*$ are the maximum possible for the specific update epoch $UE_j$. Without loss of generality, we consider $b, l \in [0, 1]$, thus, the optimal performance is achieved when $b^* \to 1, l^* \to 1$.

We define the $\tau$ metric to depict the time required to conclude a stopping decision. As $\tau \to 0.0$, $n_i$ requires limited time to conclude $\mathcal{P}$; the opposite stands when $\tau \to \infty$. The quality of the result (i.e., the maximum possible $b^*$ and $l^*$ for the epoch $UE_j$) is evaluated by the $\gamma$ and $\delta$ metrics. The following equations hold true:

$$\gamma = \frac{\sum_{i=1}^{E} b^*}{E} \tag{7}$$

$$\delta = \frac{\sum_{i=1}^{E} l^*}{E} \tag{8}$$

where $E$ is the number of the experiments. $\gamma$ depicts the average bandwidth while $\delta$ depicts the average 'availability' of each node. The higher the $\gamma$ and the $\delta$ are, the better performance the *OSS* exhibits. The optimal performance is achieved when $\gamma \to 1.0$, $\delta \to 1.0$. We also define the $\omega$ metric depicting the percentage of nodes deciding to initiate $\mathcal{P}$ at each $t$. $\omega$ shows how many nodes decide to start $\mathcal{P}$ at the same time. The following equation holds true:

$$\omega = \frac{|\mathcal{N}_s^t|}{|\mathcal{N}|} * 100\% \tag{9}$$

where $|\mathcal{N}_s^t|$ is the number of nodes deciding to initiate $\mathcal{P}$ at $t$. Recall that the stopping decision and the initiation of $\mathcal{P}$ are based on the network performance as well as the availability of each node. As we are based on the random behaviour of nodes, thus, it is an exceptional case to have all the available nodes deciding to initiate $\mathcal{P}$ at the same time. This is because, even if nodes observe the same network performance, their availability could differ. It is very difficult to have all nodes with the same load, the same complexity in their tasks to take the same decision for the initiation of $\mathcal{P}$. In any case, apart from the dynamic nature of their internal status, when a number of nodes decide to initiate $\mathcal{P}$, the remaining nodes will enjoy a different network performance affecting their future decisions. The adoption of $\omega$ aims at revealing this randomness in nodes' behaviour that will prove that nodes' decisions are 'distributed' in time and add value to the proposed model compared to a broadcasting scenario.

We compare the proposed model (i.e., *OSS*) with a distributed *deterministic model DM*. The *DM* concludes $\mathcal{P}$ immediately when an update message is received only if $b$ and $l$ are over a pre-defined threshold. In our simulations, this threshold is defined to be equal to 0.70 (also produced by simulations). The *DM* is the theoretical limit for our model concerning the time spent to derive a decision in combination with the assurance of a relatively high bandwidth and availability. The *DM* also incorporates the risk of not exceeding the pre-defined threshold, especially, in noisy and low performance networks. We also compare our model with the distributed *Moving Average Estimator* (*MAE*) [70] and the *Exponentially Weighted Moving Average* (*EWMA*) estimator [70]. *MAE* is a simple estimator that is widely used in measuring the link quality of wireless networks. The algorithm performs on top of a window of historical values and derives the average value. When the average is over a pre-defined threshold (it is defined to be equal to 0.70), *MAE* concludes $\mathcal{P}$. *EWMA* estimator is memory efficient, requiring a constant storage of the old estimates for any kind of history tuning. *EWMA* adopts a linear combination of infinite history, weighted exponentially. Again, when the *EWMA*'s result is also over a pre-defined threshold (i.e., 0.70), $\mathcal{P}$ is realized. Finally, we compare *OSS* with a centralized model, i.e., Deluge [28]. As *OSS* and Deluge have a completely different orientation (*OSS* is distributed while Deluge is centralized), we rely on the comparison of the required messages and the required time for concluding $\mathcal{P}$.

We also define the difference $D$ for each metric to depict the difference in the performance between the *OSS* and *DM*, *MAE*, *EWMA*. The following equation holds true:

$$D = \frac{P_{OSS} - P_{REF}}{P_{REF}} \cdot 100\% \tag{10}$$

where $P_{OSS}$ is the performance of the *OSS* and $P_{REF}$ is the performance of the reference model (i.e., $REF \in \{DM, MAE, EWMA\}$). $D$ is realized for each performance metric i.e., $\gamma$, $\delta$, $\tau$ and shows if *OSS* performs better than the reference models *DM*, *MAE*, *EWMA*. Depending on the metric, we expect to observe positive/negative values for $D$ as follows: (i) $D$ should be positive for $\gamma$ and $\delta$ metrics, thus, *OSS* achieves higher $b^*$ and $l^*$ than the corresponding reference model. The higher the $D$ is, the better the performance of *OSS* becomes; (ii) $D$ should be negative for $\tau$, thus, the *OSS* requires less time for the initiation of $\mathcal{P}$ than the corresponding reference model.

We evaluate our model for various realizations of $\beta$ and $\lambda_1, \lambda_2$ (we adopt different rates for the Exponential distribution for $b$ and $l$), i.e., $\beta \in \{0.2, 0.5, 0.95\}$ and $\lambda_1, \lambda_2 \in \{1, 5\}$. The higher the $\beta$ is, the higher the loss/cost becomes. The cost is related with

each observation and the loss that the $n_i$ enjoys when selecting values with low expected ranks. The higher the $\lambda_1$ and $\lambda_2$ are, the lower the values of $b$ and $l$ become. We also consider different scenarios for $b$ and $l$ by incorporating a set of $\lambda_1$, $\lambda_2$ combinations i.e., $\lambda_1 = \lambda_2 = 1$, $\lambda_1 = \lambda_2 = 5$, $\lambda_1 = 5, \lambda_2 = 1$, $\lambda_1 = 1, \lambda_2 = 5$. At $t$, we randomly generate $b$ and $l$ realizations and apply our model to record $t^*$, $b^*$ and $l^*$. For each set of experiments, we consider $E = 1000$ and get performance results for the aforementioned metrics.

We adopt two simulators, i.e., a simulator defined in Java and the Cooja simulator.[1] The former simulator 'behaves' as already described and delivers results for the envisioned metrics. The latter is a network simulator designed for WSNs. It builds on top of various types of nodes running the Contiki operating system. Several Contiki libraries can be compiled and loaded in the same Cooja simulation representing different kinds of sensor nodes. In this set of simulations, we assume the default bandwidth of the Cooja tool which is equal to 250 Kbps. For producing 'fluctuations' in the bandwidth of the network, we consider that every $\mu$ seconds, nodes send messages to a random peer. We get $\mu \in \{20, 120\}$ aiming to simulate two types of network load. When $\mu = 20$, there is an increased number of messages in the network apart from the messages related to $\mathcal{P}$. If $\mu = 120$, the number of messages is lower than in the previous case.

### 5.3. Performance assessment

Initially, we report on the complexity of the proposed model which depends on the length of $UE_j$ 'fired' just after the reception of $\mathcal{P}$'s advertisement message. Nodes should monitor the adopted metrics and perform the required calculations (e.g., calculation of the reward). At $t$, the realization of each performance metric is stored in a list of historical values, e.g., $L_B$ for $b$ and $L_L$ for $l$. The size of $L_B$ and $L_L$ could be at most equal to $UE_j$, thus, the computational complexity of the proposed model is $O(max(UE_j)^2)$. The storage requirements of the proposed *OSS* are $O(UE_j)$.

We report on the *probability density estimation* (pde) of $t^*$, $b^*$ and $l^*$. In Fig. 3, we plot the $pde(t^*)$ for $\beta \in \{0.2, 0.5, 0.95\}$. In general, $t^*$ is below 35 which indicates that the stopping decision requires at most 35 steps. Recall that $\beta$ applies 'pressure' on nodes to conclude $\mathcal{P}$ as soon as possible. In any case, $t^*$, derived by adopting the Uniform distribution, is lower than $t^*$ derived through the adoption of the Exponential distribution. In Fig. 4, we observe that $\lambda$ does not affect the realization of $t^*$, however, nodes require more than 1000 steps (the highest value) to conclude $\mathcal{P}$. Apart from that, the combination of $\lambda_1$ and $\lambda_2$ does not affect $t^*$. These results show that many fluctuations in the realization of $b$ and $l$ as depicted by the Uniform distribution make the proposed model to immediately conclude $\mathcal{P}$.

In Figs. 5 and 6, we depict our results concerning $b^*$. $b^*$ is kept at high levels close to 1.0 no matter the $\beta$ values. We observe some slight variations when $\beta = 0.95$. In such cases (i.e., $\beta = 0.95$), nodes enjoy limited loss for waiting better $b$ and $l$ under the risk of facing a low performance network in the future. A high $\beta$ makes the node to pursue high values for $b$ and $l$ even in the burden of the time (increased time for concluding $\mathcal{P}$). In the case of the Exponential distribution (Fig. 6), we observe that $\lambda_1$ and $\lambda_2$ do not significantly affect the performance of our model. *OSS* enjoys similar bandwidth at the stopping decision as such a decision is taken in combination with the $l$ realizations. This means that the focus of the proposed model is not derived exclusively on $b$ but also on $l$. The decision making tries to pay equal attention on both parameters before deciding to conclude $\mathcal{P}$. In our simulations in Cooja (see Fig. 7, we observe similar results for $\mu = 20$ and $\mu = 120$. The average $b^*$ is 0.33 for $\mu = 20$ and 0.40 for $\mu = 120$. The median values are 0.21 and 0.37, respectively.

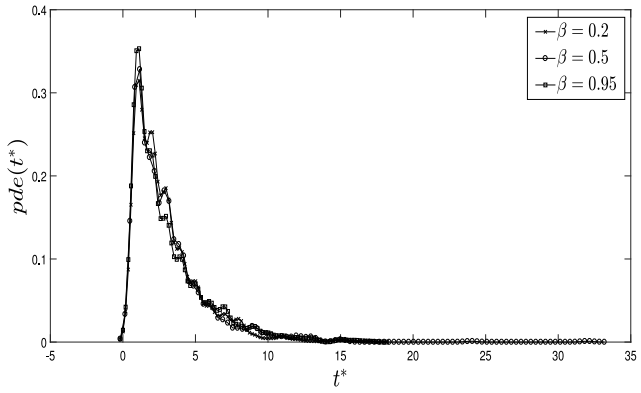



**Fig. 3.** PDE of $t^*$ adopting the Uniform distribution.

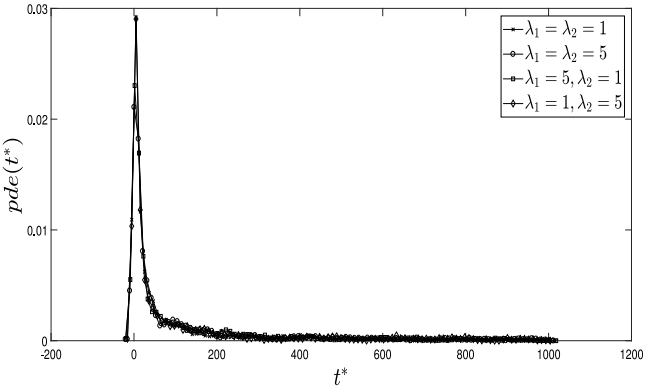



**Fig. 4.** PDE of $t^*$ adopting the Exponential distribution.

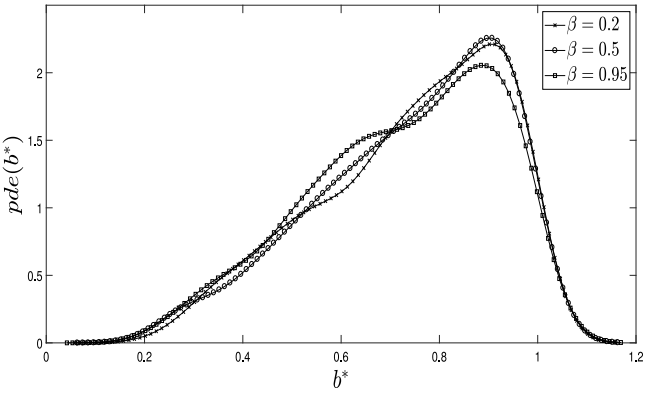



**Fig. 5.** PDE of $b^*$ adopting the Uniform distribution.

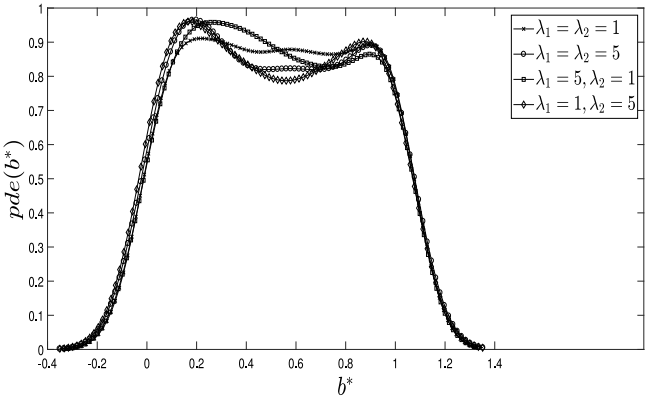



**Fig. 6.** PDE of $b^*$ adopting the Exponential distribution.

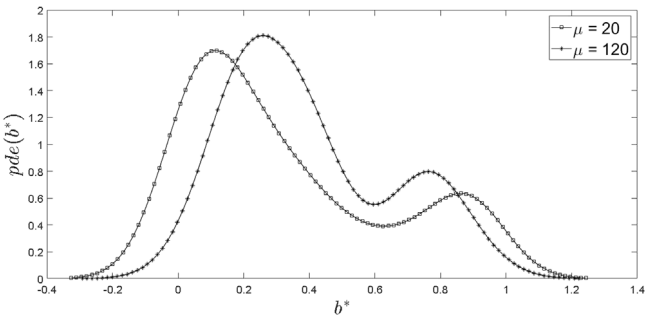



**Fig. 7.** PDE of $b^*$ as delivered by the Cooja simulator.

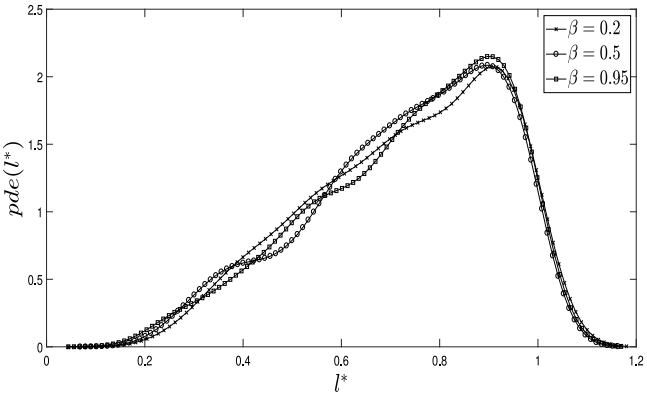



**Fig. 8.** PDE of $l^*$ adopting the Uniform distribution.

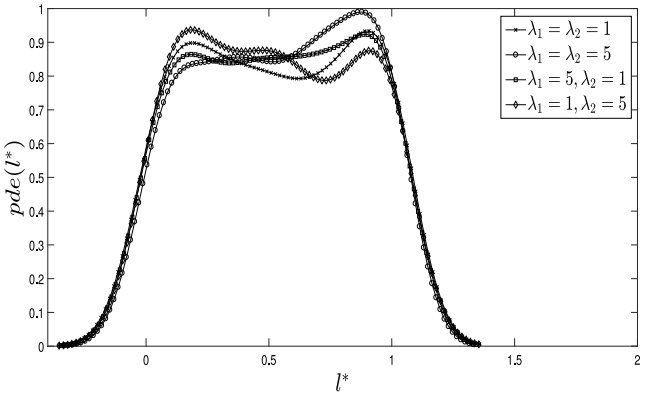



**Fig. 9.** PDE of $l^*$ adopting the Exponential distribution.

Our results related to $l^*$ are presented in Figs. 8 and 9. We observe similar results as in the experimental outcomes for $b^*$. Fig. 8 shows our results when the Uniform distribution is adopted while Fig. 9 depicts our results for the Exponential distribution. In the case of the Exponential distribution (Fig. 9), we observe that $\lambda_1$ and $\lambda_2$ do not significantly affect the performance of the *OSS* as already discussed for the $b^*$ outcomes. In the Cooja simulations, we get an average $l^*$ equal to 0.61 and 0.79 for $\mu = 20$ and $\mu = 120$, respectively. The medians are 0.71 and 0.88. We observe that the availability of nodes is high, thus, they could be able to support $\mathcal{P}$ (see Fig. 10).

$\gamma$ and $\delta$ results are presented in Figs. 11 and 12. When the Uniform distribution is adopted (Fig. 11), any increment in $\beta$ will slightly increase $\gamma$ and $\delta$. In any case, $\gamma$ and $\delta$, are kept above 0.70

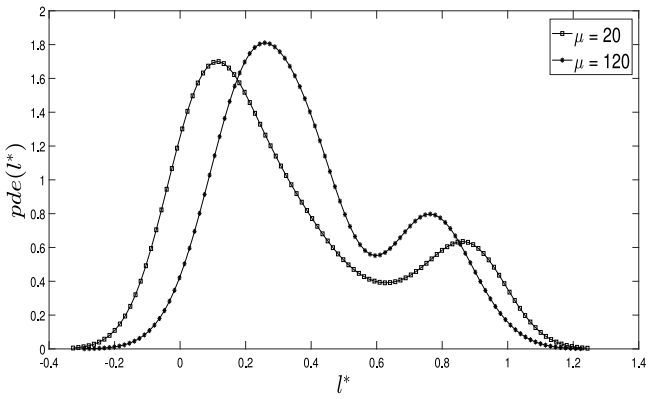



**Fig. 10.** PDE of $l^*$ as delivered by the Cooja simulator.

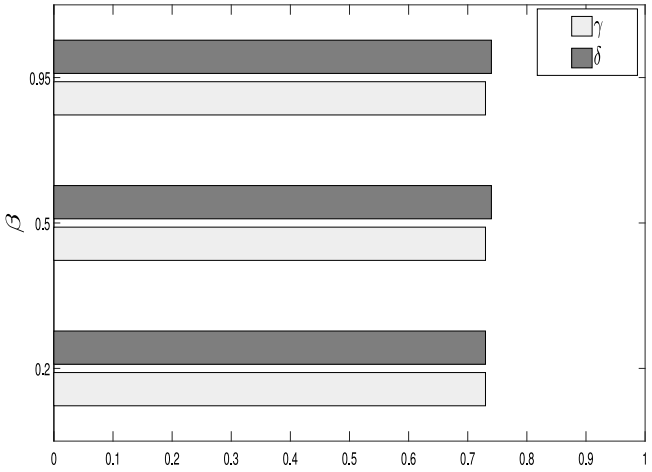



**Fig. 11.** Results for $\gamma$, $\delta$ adopting the Uniform distribution.

for $\beta \in \{0.2, 0.5, 0.95\}$. Each node enjoys high $b^*$ and $l^*$ when it decides to stop the monitoring process and conclude $\mathcal{P}$. When the Exponential is the distribution that $b$ and $l$ follow, we observe that $\gamma$ and $\delta$ are below 0.60. $\gamma$ and $\delta$ get low values compared to the results related to the Uniform distribution. Our model exhibits better performance when the monitored data are characterized by many fluctuations due to the randomness of the observations. Recall that the Exponential distribution exhibits an attitude to low or high values depending on the $\lambda$ we choose.

Let us now report on the comparison of the *OSS* with the *DM*, *MAE* and *EWMA*. In Table 2, we present the comparison between *OSS* and *DM*. Recall that the *DM* stops just after the reception of the message indicating the presence of an update in $\mathcal{S}$ and only when $b$ and $l$ are over the pre-defined thresholds. We observe that the *OSS* outperforms the *DM* concerning $\tau_D$ which means that it requires less time to conclude $\mathcal{P}$. However, this is realized in the burden of $\gamma$ and $\delta$. The *OSS* outcomes are less qualitative except when $\beta = 0.5$ and the metric under consideration is $\gamma$. The aforementioned results stand for the scenario where the Uniform distribution is adopted. In Table 3, we present our results delivered when the Exponential distribution is adopted. The *OSS* outperforms the *DM* for the entire set of the examined metrics. The reason is that the Exponential distribution leads $b$ and $l$ to have limited fluctuations, thus, the *DM* hardly finds both parameters over the pre-defined thresholds.

The comparison *OSS* vs *MAE* is presented in Tables 4 and 5. We observe that the *OSS* outperforms the *MAE* for the entire set of metrics. The difference is high when the Exponential distribution

1 http:anrg.usc.educontikiindex.phpCooja_Simulator.

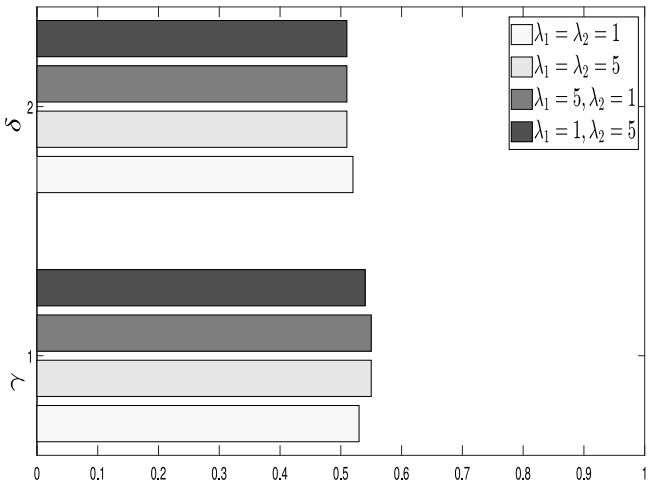



**Fig. 12.** Results for $\gamma$, $\delta$ adopting the Exponential distribution.

**Table 2**
*OSS–DM* comparison adopting the Uniform distribution.

| $\beta$ | $\gamma_D$ (%) | $\delta_D$ (%) | $\tau_D$ (%) |
|---|---|---|---|
| 0.2 | −8.75 | −8.75 | −40.00 |
| 0.5 | 8.75 | −7.50 | −50.00 |
| 0.95 | −7.59 | −8.64 | −40.00 |

**Table 3**
*OSS–DM* comparison adopting the Exponential distribution.

| $\lambda_1$ | $\lambda_2$ | $\gamma_D$ (%) | $\delta_D$ (%) | $\tau_D$ (%) |
|---|---|---|---|---|
| 1 | 1 | 3.92 | 4.00 | −71.15 |
| 5 | 5 | 10.00 | 2.00 | −69.81 |
| 5 | 1 | 12.24 | 4.08 | −70.91 |
| 1 | 5 | 12.50 | 6.25 | −69.09 |

**Table 4**
*OSS–MAE* comparison adopting the Uniform distribution.

| $\beta$ | $\gamma_D$ (%) | $\delta_D$ (%) | $\tau_D$ (%) |
|---|---|---|---|
| 0.2 | 32.73 | 32.73 | −99.66 |
| 0.5 | 32.73 | 37.04 | −99.66 |
| 0.95 | 35.19 | 37.04 | −99.67 |

is adopted. The increased performance related to $\gamma$ and $\delta$ metrics is accompanied by the limited time required to conclude $\mathcal{P}$. In addition, *MAE* requires more time than the *OSS* to reach the final decision no matter the cost of observations. *MAE* is affected by the adopted averaging scheme on top of the most recent observations for $b$ and $l$. The averaging mechanism leads to a faulty future estimations for the examined parameters. Recall that the *OSS* is not taking into consideration the past observations as nodes and network performance are not characterized by stability. We assume a very dynamic environment where the performance of the network and nodes are updated in a continuous manner. The aforementioned observations become more intense in the scenario where the Exponential distribution is adopted. By selecting a $\lambda$ to derive e.g., low values ($\lambda \rightarrow 0.0$) the difference between the two models (i.e., *OSS–MAE*) becomes high. In such cases, *MAE* cannot easily conclude a high average for $b$ and $l$ based on the most recent observations, thus, it exhibits limited performance. The *OSS* incorporates into the decision mechanism the estimation about the future reward based on the estimation of the future $b$ and $l$ values. When the Exponential distribution results in high $b$ and $l$ (the second row of Table 5), the difference is reduced (compared to the remaining scenarios).

**Table 5**
*OSS–MAE* comparison adopting the Exponential distribution.

| $\lambda_1$ | $\lambda_2$ | $\gamma_D$ (%) | $\delta_D$ (%) | $\tau_D$ (%) |
|---|---|---|---|---|
| 1 | 1 | 120.83 | 136.36 | −28.57 |
| 5 | 5 | 120.00 | 112.50 | −83.51 |
| 5 | 1 | 129.17 | 121.74 | −83.67 |
| 1 | 5 | 125.00 | 121.74 | −82.65 |

**Table 6**
*OSS–EWMA* comparison adopting the Uniform distribution.

| $\beta$ | $\gamma_D$ (%) | $\delta_D$ (%) | $\tau_D$ (%) |
|---|---|---|---|
| 0.2 | 48.98 | 48.98 | −99.70 |
| 0.5 | 43.14 | 51.02 | −99.70 |
| 0.95 | 43.14 | 48.00 | −99.70 |

**Table 7**
*OSS–EWMA* comparison adopting the Exponential distribution.

| $\lambda_1$ | $\lambda_2$ | $\gamma_D$ (%) | $\delta_D$ (%) | $\tau_D$ (%) |
|---|---|---|---|---|
| 1 | 1 | 140.91 | 147.62 | −85.00 |
| 5 | 5 | 129.17 | 131.82 | −84.00 |
| 5 | 1 | 150.00 | 142.86 | −84.00 |
| 1 | 5 | 145.45 | 131.82 | −83.00 |

In Tables 6 and 7, we present our results concerning the comparison between *OSS* and *EWMA*. In these results, we observe that the *OSS* also outperforms the *EWMA*. The average difference is 33.55% and 35.06% for $\gamma$ and $\delta$, respectively (the Uniform distribution is adopted for $b$ and $l$). For the Exponential distribution, the average difference is 123.75% and 123.09% for $\gamma$ and $\delta$, respectively. The highest difference is observed for $\beta = 0.2$ ($\gamma$ metric) and $\beta = 0.5$ for the $\delta$ metric when the Uniform distribution is adopted. The scenario involving $\lambda_1 = 5$ and $\lambda_2 = 1$ leads the *OSS* to exhibit the highest difference with the *EWMA* concerning $\gamma$. The scenario involving $\lambda_1 = \lambda_2 = 1$ leads to the highest difference concerning $\delta$.

In Figs. 13 and 14, we see our results related to the $\omega$ metric. In these experiments, we instruct 100 nodes to take a decision in 100 time steps. We get similar results, when the Uniform distribution is adopted (see Fig. 13). The use of the Uniform distribution 'allocates' the nodes to the first decision rounds aligned with the results delivered for the optimal stopping time. Nodes are forced to conclude the process as already explained in the provided experimental results. In this set of experiments, the maximum number of nodes deciding to conclude $\mathcal{P}$ at the same $t$ is 32. We also observe that the use of the Exponential, distributes the stopping time in the available interval. Except from the last decision round (i.e., $t = 100$), in the remaining rounds only a limited number of nodes decide to initiate $\mathcal{P}$. In all the experimental scenarios, the maximum number of nodes taking their decision at the same time is 36 out of 100 nodes. Even in this case, the proposed model saves resources for the conclusion of $\mathcal{P}$ compared to a broadcasting scenario.

Finally, we compare the *OSS* with the centralized system Deluge concerning the number of the required messages and time to conclude $\mathcal{P}$. Our results for *OSS* are retrieved through the use of the Cooja simulator. We adopt the same experimental scenario described in [28] and compare our model with the basic form of Deluge. Deluge's basic form involves every node occasionally advertising the most recent version of the data object it has available to whatever nodes that can hear its local broadcast. Nodes identifying a difference between the advertised data object and their local copy, they may request it from their neighbours. Nodes receiving requests then broadcast the requested data. Nodes receiving the new data objects, advertise the newly received data in order to propagate it further. Additionally, if a node has not completely received its data after making a number of requests,

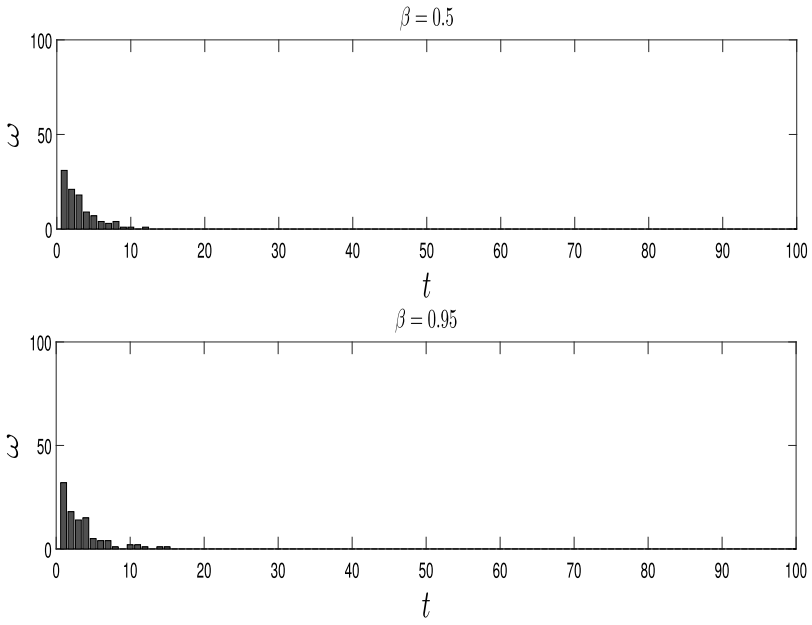



**Fig. 13.** $\omega$ results when adopting the Uniform distribution.

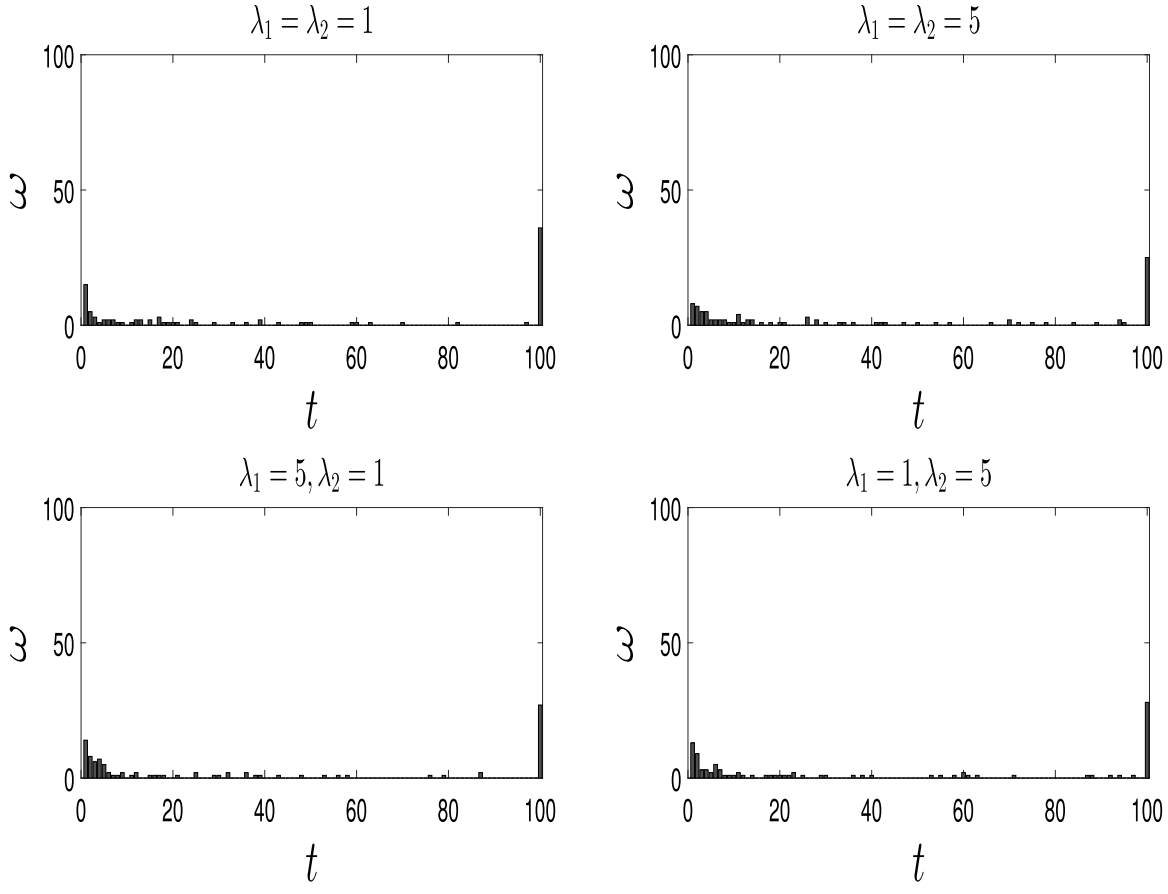



**Fig. 14.** $\omega$ results when adopting the Exponential distribution.

it searches for a new neighbour to request data. Compared to Deluge, we have to notice that the proposed model is not affected by the communication model and topology of the network. Every message has 1104 bytes per page and each data packet has a payload of 23 bytes. In Deluge, the required number of messages for the distribution of 20 pages in 75 nodes is equal to 9,966. In the *OSS*, we need one advertisement message (1023 in Deluge), 75 request messages (789 in Deluge) and 75 times the size of each update. The total number of messages in the proposed scheme is 3,226. In addition, Deluge requires an average number 10.52 of requests while the *OSS* requires a single packet just to inform the $\mathcal{S}$ and start the downloading of $\mathcal{P}$. The next set of experiments

concerns the time required to conclude $\mathcal{P}$. In Deluge, the average completion time is equal to 18.60 with a deviation of 2.60 s. The *OSS* requires an average time of 1.26 and 0.39 s for $\mu = 20$ and $\mu = 120$, respectively. The deviation in *OSS* is 2.32 and 0.57 while the medians are 0.40 and 0.24. The theoretical limit for sending the aforementioned packets is 0.09 s adopting the default bandwidth of the Cooja simulator. For 75 nodes, Deluge requires less than 275 s while *OSS* needs (in average) 94.50 and 28.89 s for $\mu = 20$ and $\mu = 120$, respectively. In the case of the *OSS*, the deviation is 174 and 42.45, respectively. The medians are 30.10 and 17.90. We observe that due to the increased traffic in the network, the *OSS* could require an increased time to conclude $\mathcal{P}$.

## 6. Conclusions & future work

IoT and pervasive computing demand novel applications on top of the autonomous nature of independent nodes. In this paper, we propose a distributed, time-optimized, performance aware model that aims to assist the autonomous nodes to initiate and conclude an update processes. The proposed scheme alleviates the central servers from the burden of supporting complex protocols for the distribution of the updates while being aware of nodes' specific characteristics. Each node, independently, decides when it will conclude the update process according to the result of a monitoring process. The monitoring process aims to provide a view on the performance of the network and the node itself. When the performance is of high quality, there is a room for applying the updates without disturbing the node from the assigned tasks. The decision will be to realize the communication with the central server and conclude the update process. In contrast to centralized systems, the network is not flooded by update messages and nodes' performance remains at high levels. We adopt an infinite horizon time-optimized model applied on top of multiple performance metrics. The model results the time when the update process should be concluded taking into consideration the dynamic nature of nodes. The proposed mechanism is fully adapted on the performance of the network securing the uninterrupted application of the updates. Future extensions of our work involve the implication of an adaptive model fully aligned with the nodes needs. The adaptive model will try to handle the uncertainty related to the state of the environment and nodes behaviour. With this approach, we will offer a complete model for concluding updates either in short- or in long-term.

## Acknowledgments

This work is funded by the EU/H2020 Marie Sklodowska-Curie Action under the INNOVATE project; Grant #745829.

## References

[1] P. Abdulah, S. Waseem, R. Bai, I. Mohsin, Development of new water quality model using fuzzy logic system for Maysia, Open Environ. Sci. 2 (2008) 101–106.
[2] M. Akkouchi, On the conution of exponential distributions, J. Chungcheong Math. Soc. 21 (4) (2008).
[3] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, Wireless sensor networks: a survey, Comput. Netw. 38 (2002) 393–422.
[4] H.O. Al-Sakran, Intelligent traffic information system based on integration of Internet of Things and Agent technology, IJACSA 6 (2015).
[5] L. Atzori, A. Iera, G. Morabito, The internet of things: A survey, Comput. Netw. (2010) 2787–2805.
[6] S. Brown, C. Sreenan, Software updating in wireless sensor networks: A survey and Lacunae, J. Sensor Actuators 2 (2013) 717–760.
[7] R. Buyya, C.S. Yeo, S. Venugopal, J. Broberg, I. Brandic, Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility, Future Gener. Comput. Syst. 25 (2009) 599–616.
[8] Q. Cao, T. Abdelzaher, J. Stankovic, T. He, The liteos operating system: Towards unix-like abstractions for wireless sensor networks, in: Proceedings of the 7th International Conference on Information processing in sensor networks, IPSN '08, 2008.
[9] H. Cha, S. Choi, I. Jung, H. Kim, H. Shin, J. Yoo, C. Yoon, RETOS: resilient, expandable, and threaded operating system for wireless sensor networks, in: Proceedings of the 6th International Conference on Information Processing in Sensor Networks, IPSN '07, 2007.
[10] A. Chepuru, K.V. Rao, A study on security of iot in Intelligent Transport Systems Applications, IJARCSEE 5 (2015).
[11] J. Choi, On the energy efficiency and total bandwidth in channel-aware random access for WSNs, in: Proceedings of the IEEE Signal Processing for Communications Symposium, 2015.
[12] T. Dang, N. Bulusu, W. Feng, S. Park, DHV: A code consistency maintenance protocol for wireless sensor networks, in: Proceedings of the 6th European Conference on Wireless Sensor Networks (EWSN 2009), Cork, Ireland, 2009.
[13] A. Dunkels, B. Grnvall, T. Voigt, Contiki - a lightweight and flexible operating system for tiny networked sensors, in: Proceedings of the IEEE Workshop on Embedded Networked Sensors (Emnets-I), 2004.
[14] M. Erol-Kantarci, H.T. Mouftah, Wireless sensor networks for domestic energy management in smart grids, in: Proceedings of the 25th Biennial Symposium on Communications (QBSC), 2010, pp. 63–66.
[15] T.S. Ferguson, Optimal Stopping and Applications', Mathematics Department, UCLA, Available online https://www.math.ucla.edu/tom/Stopping/Contents.html (accessed November, 2016).
[16] J. Gao, Y. Xiao, J. Liu, W. Liang, C. Chen, A survey of communication/networking in smart grids, Future Gener. Comput. Syst. 28 (2) (2012) 391–404.
[17] A. Garcia-Saavedra, P. Serrano, A. Banchs, Energy-efficient optimization for distributed opportunistic scheduling, IEEE Commun. Lett. 18 (6) (2014) 1083–1086.
[18] A. Ghosh, S.K. Das, Coverage and connectivity issues in wireless sensor networks: a survey, Pervasive Mob. Comput. 4 (2008) 303–334.
[19] A. Gluhak, S. Krco, M. Nati, D. Pfisterer, N. Mitton, T. Razafindralambo, A survey on facilities for experimental Internet of Things research, IEEE Commun. Mag. 49 (2011) 58–67.
[20] C. Gouveia, A. Fonseca, New approaches to environmental monitoring: the use of ICT to explore unteered geographic information, GoeJournal 72 (2008) 185–197.
[21] J. Gubbi, R. Buyya, S. Marusic, M. Palaniswami, Internet of Things (IoT): A vision, architectural elements, and future directions, Elsevier Future Gener. Comput. Syst. 29 (2013) 1645–1660.
[22] A. Hanemann, A. Liakopoulos, M. Molina, D.M. Swany, A study on network performance metrics and their composition, Campus-Wide Inf. Syst. 23 (4) (2006) 268–282.
[23] B.M. Hardas, G.M. Asutkar, K.D. Kulat, Environmental monitoring using wireless sensors: A simulation approach, in: Proc. of the 1st International Conference on Emerging Trends in Engineering and Technology, 2008, pp. 255–257.
[24] E. Hatzikos, N. Bassiliades, L. Asmanis, I. Vlahavas, Monitoring water quality through a telematic sensor network and a fuzzy expert system, Expert Syst. 24 (3) (2007) 143–161, Blackwell.
[25] A. Hava, Y. Ghamri-Doudane, J. Murphy, A study on monitoring overhead impact on wireless mesh networks, in: Proceedings of the 8th International Wireless Communications and Mobile Computing Conference, IWCMC'12, Limassol, Cyprus, 2012, pp. 487–492.
[26] K. Henricksen, R. Robinson, A survey of middleware for sensor networks: State-of-the-Art and future directions, in: Proceedings of the International Workshop on Middleware for Sensor Networks, 2006, pp. 60–65.
[27] I.H. Hou, Y.E. Tsai, T.F. Abdelzaher, I. Gupta, AdapCode: Adaptive network coding for code updates in wireless sensor networks, in: Procedings of the IEEE 27th Conference on Computer Communications, 2008.
[28] J.W. Hui, D. Culler, The dynamic behavior of a data dissemination protocol for network programming at scale, in: Proceedings of the International Conference on Embedded networked sensor systems, SenSys, 2004, pp. 81–94.
[29] M.F. Ibrahim, M. Jamal, S. Yahya, Taib. N., Available bandwidth estimation in network-aware applications for wireless campus e-Learning system, J. Comput. Netw. Commun. 2012 (art. ID 380959) (2011).
[30] C. Intangonwiwat, R. Govindan, D. Estrin, Directed diffusion: A scalable and robust communication paradigm for sensor networks, in: Proceedings of the 6th Annual Conference on Mobile Computing and Networking, Boston, MA, USA, 2000, pp. 56–67.
[31] A. Juels, RFID Security and privacy: a research survey, IEEE J. Sel. Areas Commun. 24 (2006) 381–394.
[32] V. Katiyar, P. Kumar, N. Chand, An intelligent transportation system architecture using wireless sensor network, Int. J. Comput. Appl. 14 (2011).
[33] K.E. Kjaer, A survey of context-aware middleware, in: Proceedings of the 25th conference on IASTED International Multi-Conference: Software Engineering, 2007, pp. 148–155.
[34] K. Kolomvatsos, Time-optimized management of IoT nodes, Elsevier Ad Hoc Netw. 69 (2018) 1–14.
[35] K. Kolomvatsos, C. Anagnostopoulos, S. Hadjiefthymiades, An efficient environmental monitoring system adopting data fusion, prediction and fuzzy logic, in: Proceedings of the 6th International Conference on Information, Intelligence, Systems and Applications, Corfu, Greece, 2015.

[36] K. Kolomvatsos, C. Anagnostopoulos, S. Hadjiefthymiades, Intelligent contextual data stream monitoring, in: Proceedings of the 8th International Conference on Pervasive Technologies Related to Assistive Environments, Corfu, Greece, 2015.
[37] S.S. Kulkarni, M. Arumugam, Infuse: A TDMA Based Data Dissemination Protocol for Sensor Networks; Technical Report MSU-CSE-04-46 for the Department of Computer Science and Engineering, Michigan State University, East Lansing, MI, USA, 2004.
[38] S. Kulkarni, L. Wang, Mnp: Multihop network reprogramming service for sensor networks, in: Distributed Computing Systems, 2005, pp. 7–16.
[39] R. Kumar, A. Paul, U. Ramachandran, D. Kotz, On Improving Wireless Broadcast Reliability of Sensor Networks Using Erasure Codes', Mobile Ad-hoc and Sensor Networks, in: Lecture Notes in Computer Science, vol. 4325, Springer, Berlin, Heidelberg, 2006.
[40] H. Leligou, C. Massouros, E. Tsampasis, T. Zahariadis, D. Bargiotas, K. Papadopoulos, S. Voliotis, Reprogramming wireless sensor nodes, Int. J. Comput. Trends Technol. (2011).
[41] P. Levis, S. Madden, J. Polastre, R. Szewczyk, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, D. Culler, TinyOS: an operating system for sensor networks, in: Ambient Intelligence, Springer Verlag, 2004.
[42] P. Levis, N. Patel, D. Culler, S. Shenker, Trickle: a self-regulating algorithm for code propagation and maintenance in wireless sensor networks, in: Proceedings of the Symposium on Networked Systems Design and Implementation, Vol. 1, 2004.
[43] Y. Li, B. Li, Y. Zhang, A channel state information feedback and prediction scheme for time-varying underwater acoustic channels, in: Proceedings of the International Conference on Intelligent Transportation Big Data & Smart City, 2018, pp. 141–144.
[44] X. Li, R. Lu, X. Liang, X. Shen, J. Chen, X. Lin, Smart community: an Internet of Things application, IEEE Commun. Mag. 49 (2011) 68–75.
[45] W. Li, Y. Zhang, B. Childers, MCP: an energy-efficient code distribution protocol for multi-application WSNs, in: Proceedings of the 5th IEEE International Conference on Distributed Computing in Sensor Systems, 2009.
[46] K. Lin, P. Levis, Data discovery and dissemination with dip, in: Proceedings of the International Conference on Information Processing in Sensor Networks (IPSN 2008), Washington, DC, USA, 2008, pp. 433–444.
[47] J.M. Lujano-Rojas, M. Monteiro, D. Dufo-Lopez, J.L. Bernal-Agustin, Optimum residential load management strategy for real time pricing (rtp) demand response programs, Energy Policy (2012).
[48] K. Maier, A. Hessler, O. Ugus, J. Keller, D. Westhoff, Multi-Hop Over-The-Air reprogramming of wireless sensor networks using fuzzy control and fountain codes, in: Self-Organising Wireless Sensor and Communication Networks, 2009.
[49] C. Miller, C. Poellabauer, Reliable and efficient reprogramming in sensor networks, ACM Trans. Sensor Netw. 7 (2010) 1–32.
[50] M. Miraoui, C. Tadj, C.B. Amar, Architectural survey of context-aware systems in pervasive computing environment, Ubiquitous Comput. Commun. J. 3 (3) (2008).
[51] A. Moraleda-Soler, B. Coll-Perales, J. Gozalvez, Link-aware opportunistic D2D communications: Open source test-bed and experimental insights into their energy, capacity and QoS, in: 11th International Symposium on Wireless Communications Systems (ISWCS), 2014.
[52] V. Naik, A. Arora, P. Sinha, H. Zhang, Sprinkler: A reliable and energy efficient data dissemination service for wireless embedded devices, in: Proceedings of the 26th IEEE International Real-Time Systems Symposium, Miami, FL, USA, 2005, pp. 5–8.
[53] R. Panta, I. Khalil, S. Bagchi, Stream: Low overhead wireless reprogramming for sensor networks, in: Proceedings of the International Conference on Computer Communications, INFOCOM, 2007, pp. 928–936.
[54] H.U. Park, J. jeong, P. mah, Non-invasive rapid and efficient firmware update for wireless sensor networks, in: Proceedings of the UBICOMP, Seattle, WA, USA, 2014.
[55] M.A. Pedrasa, T. Spooner, I.F. MacGill, Coordinated scheduling of residential distributed energy resources to optimize smart home energy services, IEEE Trans. Smart Grid 1 (2) (2010) 134–143.
[56] G. Peskir, A. Shiryaev, Optimal Stopping and Free Boundary Problems, ETH Zuerich, Birkhauser, 2006.
[57] M.I. Poulakis, A.D. Panagopoulos, P. Constantinou, Channel-aware opportunistic transmission scheduling for energy-efficient wireless links, IEEE Trans. Veh. Technol. 62 (1) (2013) 192–204.
[58] Y. Sang, H. Shen, Y. Inoguchi, Y. Tan, N. Xiong, Secure data aggregation in wireless sensor networks: A survey, in: 7th International Conference on Parallel and Distributed Computing, Applications and Technologies, 2006, pp. 315–320.
[59] A. Sen, N. Balakrishnan, Conution of geometrics and a reliability problem, Statist. Probab. Lett. 43 (1999) 421–426.
[60] F. Stann, J. Heidemann, RMST: Reliable data transport in sensor networks, in: Proceedings of the 1st IEEE International Workshop on Sensor Network Applications and Protocols, Vol. 11, Anchorage, AK, USA, 2003, pp. 102–112.
[61] T. Stathopoulos, J. Heidemann, D. Estrin, A remote code update mechanism for wireless sensor networks', Technical Report, Center for Embedded Networked Sensing, 2003.
[62] M. Stolikj, P. Cuijpers, J. Lukkien, Efficient reprogramming of wireless sensor networks using incremental updates and data compression, in: Proceedings of the IEEE International Conference on Pervasive Computing and Communications Workshops, 2013, pp. 584–589.
[63] S. Tilak, N. Abu-Ghazaleh, W. Heinzelman, A taxonomy of wireless microsensor network models, ACM Mob. Comput. Commun. Rev. 6 (2002) 28–36.
[64] M. Tory, T. Moller, Rethinking visualization: a high-level taxonomy, in: IEEE Symposium on Information Visualization, INFOVIS 2004, 2004, pp. 151–158.
[65] Y. Tseng, S.-Y. Ni, Y.-S. Chen, J.-P. Sheu, The broadcast Storm problem in a mobile ad hoc network, Wirel. Netw. 8 (2/3) (2002) 153–167.
[66] C.Y. Wan, A.T. Campbell, L. Krishnamurthy, PSFQ: A reliable transport protocol for wireless sensor networks, in: Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications, Vol. 28, Atlanta, Georgia, USA, 2002, pp. 1–11.
[67] M.M. Wang, J.N. Cao, J. Li, S.K. Das, Middleware for wireless sensor networks: A survey, J. Comput. Sci. Tech. 23 (3) (2008) 305–326.
[68] Z. Wang, C. Wang, W. Sun, Adaptive transmission scheduling in time-varying underwater acoustic channels, in: Proceedings of the MTS/IEEE OCEANS, 2015.
[69] E. Welbourne, L. Battle, G. Cole, K. Gould, K. Rector, S. Raymer, Building the Internet of Things using RFID the RFID ecosystem experience, IEEE Internet Comput. 13 (2009) 48–55.
[70] A. Woo, D. Culler, Evaluation of efficient link reliability estimators for low-power wireless networks, in: Tech. Rep. UCB/CSD-03-1270, EECS Department, University of California, Berkeley, 2003.
[71] L. Xiao, Internet of Things: a new application for intelligent traffic monitoring system, J. Netw. 6 (2011).
[72] Y. Yin, W. Wu, A real time measurement algorithm for available bandwidth, Int. J. Commun. Netw. Syst. Sci. 2 (2009) 746–753.
[73] Y. Yu, L.J. Rittle, V. Bhandari, J.B. Lebrun, Supporting concurrent applications in wireless sensor networks, in: Proceedings of the 4th International Conference on Embedded Networked Sensor Systems, SenSys, 2006.

**Kostas Kolomvatsos** received his B.Sc. in Informatics from the Dept. of Informatics at the Athens University of Economics and Business in 1995, his M.Sc. and his Ph.D. in Computer Science from the Dept. of Informatics and Telecommunications at the University of Athens (UoA) in 2005 and in the beginning of 2013, respectively. Currently, he serves as a Senior Researcher in UoA, as an Adjunct Lecturer at the University of Thessaly, Dept. of Computer Science and as a Research collaborator in the School of Computing, University of Glasgow. He has participated in several European and National research projects. His research interests are in the definition of Intelligent Systems adopting Machine Learning, Computational Intelligence and Soft Computing for Pervasive Computing, Distributed Systems and Large Scale Data Streams. He is the author of over 65 publications in these areas.