Hindawi
Wireless Communications and Mobile Computing
Volume 2018, Article ID 2796282, 18 pages
https://doi.org/10.1155/2018/2796282

Research Article

Opportunistic Mobile Sensing in the Fog

Rolando Menchaca-Mendez,[1] **Brayan Luna-Nuñez**,[1] **Ricardo Menchaca-Mendez**,[1] **Arturo Yee-Rendon**,[2] **Rolando Quintero**,[1] **and Jesus Favela**[3]

[1]*Instituto Politécnico Nacional, Mexico City, Mexico*
[2]*Universidad Autónoma de Sinaloa, Sinaloa, Mexico*
[3]*Centro de Investigación Científica y de Educación Superior de Ensenada, Ensenada, Baja California, Mexico*

Correspondence should be addressed to Rolando Menchaca-Mendez; rolando.menchaca@gmail.com

Received 21 May 2018; Accepted 25 July 2018; Published 8 August 2018

Academic Editor: Iván García-Magariño

The increasing adoption of mobile personal devices and Internet of Things devices is leveraging the emergence of a wide variety of opportunistic sensing applications. However, the designers of this type of applications face a set of technical challenges related to the limitations and heterogeneity of the hardware and software platforms and to the dynamics of the scenarios where they are deployed. In this paper, we introduce a Semantic-Centric Fog-based framework aimed at effectively and efficiently supporting this type of applications. The proposed framework is composed of local and distributed algorithms that support the establishment and coordination of sensing tasks in the Fog. First, it performs ontology-driven in-network processing to locate the most adequate devices to carry out a given sensing task and then, it establishes efficient multihop routes that are used to coordinate relevant devices and to transport the collected sensory data to Fog sinks. We present a set of theorems that prove that the proposed algorithms are correct and the results of a series of detailed simulation-based experiments in NS3 that characterize the performance of the proposed platform. The results show that the proposed framework outperforms traditional sensing platforms that are based on centralized services.

1. Introduction

Fog computing is a distributed paradigm for transporting, storing, analyzing, and acting on data generated by a swarm of heterogeneous networked devices such as Internet of Things (IoT) [1] devices and personal mobile devices that are located at the network edge [2–5].

Fog computing provides Cloud-like services implemented close to where data is generated. The purpose is manifold: (1) to provide stable resources to the swarm at the network edge: this way, edge devices do not have to rely solely on their limited resources; (2) to improve scalability by offloading data traffic from the core network [5, 6]: in the Fog computing model, only selected or preprocessed data is transported through the core network to the Cloud [4]; (3) to reduce response time: by analyzing and acting on time-sensitive data close to where it is generated, systems can eliminate a network round-trip time, reducing the response delay and jitter [7, 8]; (4) to improve privacy by storing privacy-sensitive data at the local premises [3, 5, 7–9]; and (5) to improve efficiency by distributing processing, storing, and communication functions anywhere between the Cloud and the swarm at the edge [7, 10].

The Fog computing paradigm provides an ideal platform for implementing sensing applications because it enables seamless integration between the unprecedented capabilities to monitor the physical world [11] of dedicated sensor networks, personal mobile, and IoT devices; and the scalable storage and high-performance computing capabilities for data analytics of the Cloud [8]. In fact, despite the large diversity of Fog applications (e.g., smart grid, smart traffic, and smart buildings [9]), all of them include a sensing component involving IoT devices [4], mobile devices, and/or sensor networks. However, the vast majority of current

sensing applications typically address a single scenario on a dedicated set of resources. While this approach provides performance guarantees and reliability, it prevents the explosion of possibilities that result from sharing data, hardware, and software services across applications [11].

In this paper, we present a platform for opportunistic sensing [12] in the Fog, where collections of heterogeneous networked devices (e.g., IoT devices, dedicated sensor networks, and personal mobile devices) can self-organize to collect relevant sensory data and to efficiently transport it to Fog sinks. The result is an adaptive platform that is able to opportunistically take advantage of the local sensing devices to support a wide diversity of sensing applications with optimized performance. Please note that opportunistic sensing and the Fog share a common fundamental research question, namely, how to distribute computational tasks over a dynamic set of heterogeneous resource-constrained wireless nodes [7].

Opportunistic sensing (OS) systems differ in important ways from traditional sensor networks, introducing new challenges but also opening new opportunities. While sensor networks are typically deployed as a well-known set of homogeneous devices, OS systems are usually composed of highly heterogeneous devices with diverse hardware characteristics [13]. Moreover, OS systems are much more dynamic in the sense that completely new types of networked sensors can come into play at any time during the system operation, and OS platforms should be able to seamlessly integrate them into the sensing tasks. Therefore, identifying the right set of devices, those that can produce the desired sensory data with the proper context and at minimum network cost become one of the most important and complex problems for the complete realization of an opportunistic sensing platform [12, 14]. The context of a sensor may include its current battery's energy level and its geographical location, but also, in the case of mobile devices such as smartphones, the set of applications running on the foreground, as well as instantaneous readings of their sensors that may indicate, for instance, whether a device is inside of a bag.

Another important difference between OS and traditional sensor networks is the degree in which human users are involved in the sensing tasks. In the *opportunistic sensing* paradigm, sensing applications can run in the background of mobile personal devices while opportunistically collecting data. In this type of scenarios, personal devices cannot be overloaded with continuous sensing tasks that may reduce the user experience by disrupting applications or depleting battery power because it may prevent users to participate in future sensing tasks [15]. This has motivated the development of collaborative sensing strategies, where two or more mobile devices share the sensing, processing, or communication load to save resources [16].

In a nutshell, the main advantage of opportunistic sensing platforms is that they can harness the sensing and communication capabilities of the static sensor networks deployed in a given environment, but also of mobile and IoT devices that happen to also be located at the same environment. This is all to provide sensing services to applications running on local devices but also to large-scale applications running on the Cloud.

The proposed framework for opportunistic sensing is based on *Semantic-Centric Sensing Foglets* (sensing Foglets for short) which are dynamic collections of local heterogeneous networked devices such as smartphones, traditional sensor networks, IoT devices, and mobile and desktop computers. These devices organize themselves to collect and deliver relevant sensory information to a designated local Fog data sink. For a given sensing request, the proposed framework locates and identifies the set of local sensing devices that are most fit to perform the task. This selection is based on the computation of a semantic distance function between semantic labels describing the requested sensor and the sensors installed on the local devices, the instantaneous context of the sensing devices, and the communication cost.

The main contributions of this paper are as follows: (1) an ontology-based semantic distance function, with an efficient implementation, that can be computed without accessing the whole ontology; (2) a semantic-driven distributed algorithm that uses in-network processing to locate a set of sensing devices that are able to perform a given sensing task at minimum network cost. This way, sensing tasks can be opportunistically carried out by a combination of mobile devices, IoT artifacts, traditional sensor networks and Fog devices; and (3) an effective and efficient distributed algorithm that instantiates *sensing Foglets* by establishing and maintaining multihop paths connecting the *best* sensing devices in the environment to Fog sinks and that implements *collaborative sensing schedules* where multiple devices can share the load of implementing a sensing task. These sensing schedules can reduce local network contention and congestion by balancing the network load. They can also be used to improve the quality of the sensory data by providing redundant sources of information.

The rest of this paper is organized as follows. In Section 2, we describe existing mobile sensing platforms, emphasizing the fact that most of these platforms are either based on centralized architectures where mobile nodes communicate directly to services on the Internet; or do not address the problem of finding the best set of sensing and communication nodes. In Section 3, we present the proposed opportunistic sensing architecture and formulate the problem of instantiating semantic-centric Foglets for sensing. In Section 4, we establish the correctness of the proposed algorithms. Section 5 presents the results of a series of detailed simulation-based experiments that show that the Semantic-Centric Sensing Foglets outperform traditional sensing platforms in terms of efficiency and effectiveness. Lastly, in Section 6 we present our concluding remarks and future research directions.

2. Mobile Sensing

Many sensing platforms have been proposed in recent years. Here we present a small, but representative sample of that body of work. Sensing platforms can be classified as either centralized or distributed depending on the way the mobile nodes interact among them. Representatives of centralized

platforms are METIS [17], PRISM [18] Medusa [18], and InCense [19], while representatives of distributed platforms are COUPON [20] and the work by Ngai et-al [21].

METIS [17] is a sensing platform that offloads sensing tasks to sensors embedded in the environment with the goal of conserving energy. METIS assumes the presence of a centralized rendezvous point, which is used by smartphones to query the infrastructure about the available sensing resources and their capabilities. Offloading decisions are based on the estimated energy cost when sensing is performed on the phone and the prediction of the energy cost when sensing is performed by sensors in the environment. Unlike the proposed platform, METIS does not provide support to generate sensing schedules where multiple devices can be involved in a given sensing task. In [22], the authors present a human-based data-muling system where smartphones are used to collect data from sensor networks. Then, smartphones use a 3G cellular link or 802.11 to upload their collected data to a centralized server. EEMSS [23] and Jigsaw [24] are two sensing platforms that focus on energy efficiency through the optimized use of the smartphones' sensors. These proposals, however, consider neither the problem of sensor selection nor collaboration among smartphones.

PRISM [25] is based on a client-server architecture where a PRISM server accepts sensing jobs from application servers; then, these jobs are deployed by pushing an application into smartphones that comply with a set of predicates. The PRISM runtime platform implements a software sandbox where the sensing application is executed. This sandbox also provides functionality for resource metering, for preventing applications to retain sensed data, and for allowing users to establish policies on the type of applications that they are willing to run on their phones. Once data is collected, it is transmitted back to the PRISM server through a wireless WAN link. AnonySense [26] also uses a client-server architecture where users of smartphones can volunteer to accept sensing tasks and send back anonymous reports. Tasks are accepted based on the acceptance condition defined by the task issuer and on the local policies of the phones. The main focus of AnonySense is to preserve privacy in opportunistic sensing environments. Medusa [18] provides a high-level language for developing crowd-sensing tasks which are specified as a sequence of steps that are executed by the Medusa runtime system, which is structured as a set of services that run on the Cloud and on the phones. These services are in charge of coordinating the execution of the sensing task between the smartphones and a cluster on the Cloud. InCense [19] is a general purpose mobile phone sensing platform for deploying sensing campaigns through a visual programming paradigm. The architecture of InCense is based on a centralized *context-aware server* that coordinates smartphones in order to improve the quality of the sensed data and reduce energy consumption. Neither PRISM, AnonySense, Medusa, nor InCense has support for direct peer-to-peer communication between smartphones. These four proposals are complementary to the one presented here in the sense that they tackle related but orthogonal aspects of mobile opportunistic sensing.

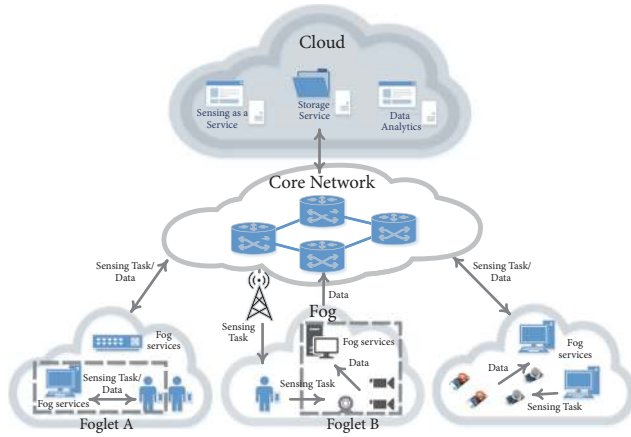

Figure 1: Cloud/Fog-based opportunistic sensing architecture.

COUPON [20] is a cooperative sensing and data forwarding framework that incorporates a cooperative sensing scheme and two store-carry-and-forward forwarding schemes with data fusion. In COUPON, the area of interest is divided into grid cells whose size is defined by the application requirements. Time is also divided into time slots that are further divided into sampling periods. Cooperative sensing consists of nodes reporting to their neighbors their coverage tables that contain two-tuples consisting of grid cell identifiers and the time that the grid cell has been covered during a time period. Using this information, nodes may decide not to sense and transmit redundant information regarding a given cell in the current time period. The authors implicitly assume that nodes are homogeneous and capable of sensing the required variable with the appropriate resolution at any time. Lastly, in [21] the authors propose a context-aware sensing data dissemination framework where smartphones are either used as sensors or as data mules that opportunistically collect data from stationary sensors through short-range communications.

3. Semantic Fog for Opportunistic Sensing

Figure 1 depicts the proposed Cloud/Fog-based opportunistic sensing architecture that aims at integrating the storage and processing power of the Cloud with the sensing capabilities of any networked device that happens to be located in a region that is of interest for a sensing application. In the proposed architecture, sensing applications running either on a mobile device (e.g., a personal health-care application) or on the Cloud (e.g., a large-scale public-health sensing application) can issue sensing tasks to Fog environments requesting to monitor a given set of variables, during a specified period of time and in a region of interest. As a response to such request, the devices in the Fog environment will organize themselves to create a *Foglet* composed of interconnected devices that have the capabilities to fulfill the sensing task by delivering the requested data to a designated Fog data sink. This is at minimum network cost in terms of congestion and contention. Under this paradigm, the sensing Foglets

are in charge of providing the sensing capabilities needed to implement personal, community, and large-scale sensing applications that run either on a mobile device or on the Cloud. It is important to mention that, even though our platform is specifically designed to instantiate Foglets for sensing, the same architecture can be used to provide other types of Fog services.

Figure 1 shows two examples of the way in which the proposed opportunistic sensing Fog environments operate. In the first example, Foglet A (composed of a smartphone and a personal computer) is instantiated as a response to a sensing task issued from the Cloud requesting to measure the noise level at the local environment. In the example, the smartphone is selected because it is equipped with an adequate sensor (e.g., a microphone), has the right context (e.g., it has enough energy in its battery, is not stored and is located in the region of interest), and is located just one-hop away from the designated Fog sink (the same personal computer that received the request). In the second example, Foglet B is instantiated as a response to a sensing task issued by an application running on a mobile device located in the local environment. This sensing task requests three devices with video cameras to record a video for a designated time period. The task of this example also designates a local Fog storage service as a sink so that the mobile device can move out of the environment without interrupting the data collection process.

For the sake of completeness, Figure 1 also shows two other Cloud services, which are relevant to opportunistic sensing platforms, namely, a storage service and a data analytics service.

In the following subsections, we present the system model and formally formulate the problem of instantiating an effective and efficient semantic-driven sensing Foglet.

3.1. System Model. We use a time varying graph $G(t) = (V(t), E(t))$ to model the topology of a dynamic network composed of a time varying set $V(t)$ of heterogeneous nodes (e.g., desktop computers, mobile and IoT devices, and sensor nodes) that at time t are located in a given Fog environment. Nodes in the Fog environment interact with each other by means of wireless links that are modeled by edges $(u, v) \in E(t)$. Two nodes $u, v \in V(t)$ are connected by a link $(u, v) \in E(t)$ at time t, if and only if $dist(p^t(u), p^t(v)) \leq range$. Where $range$ is the transmission radio range, $p^t : V(t) \longrightarrow \mathbb{R} \times \mathbb{R}$ is a function that assigns to each node $u \in V(t)$ a position in the (x, y)-plane at time t and $dist$ is the Euclidean distance between two points. In practice, individual nodes know their own instantaneous position only, usually through GPS or from other positioning services.

Since nodes are heterogeneous, they can be equipped with different types of sensors that provide them with a specific set of capabilities. Such sensors are described by means of a sensor ontology $\mathbb{O}_{\mathbb{S}} = (C_S, R_S)$ that contains concepts $c \in C_S$ describing the different types of sensors and a set of unidirectional relations $(x, y) \in R_S$ between concepts $x, y \in C_S$. For this work, we assume that the relations in R_S, as well as the relations in the other ontologies, are *is-a* relations and that

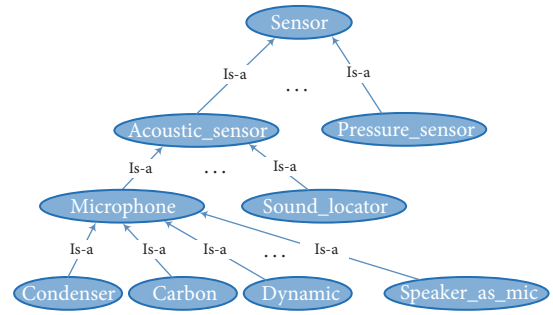

Figure 2: A small fraction of the sensor ontology.

the ontologies are organized as hierarchies. Figure 2 presents a small portion of the sensor ontology $\mathbb{O}_{\mathbb{S}} = (C_S, R_S)$ that shows the branch that describes the acoustic sensors. In the figure, the relation *(Microphone, Acoustic_sensor)* indicates that a *Microphone* is a type of *Acoustic_sensor*, and, for instance, that a node equipped with a microphone is adequate to fulfill a request for an *Acoustic_sensor*.

The particular sensors of a given device (also referred as node) $u \in V(t)$ are specified by a function $\eta : V(t) \longrightarrow \mathcal{P}(C_S)$, where $\mathcal{P}(C_S)$ denotes the power set of the concepts C_S in the ontology $\mathbb{O}_{\mathbb{S}}$. This means that a node can have any subset (e.g., a *Pressure_sensor* and an *Acoustic_sensor*) of the sensor types described in the ontology $\mathbb{O}_{\mathbb{S}} = (C_S, R_S)$. Please note that in practice, the information of function η is distributed among the nodes in the environment, namely, that each node only knows the information regarding their local sensors.

Similarly, a function $\kappa^t : V(t) \longrightarrow \mathbb{R}^+ \times C_A \times \{inside, outside\}$ defines the individual instantaneous node context in terms of a 3-tuple $(\epsilon, app, e_c) \in \mathbb{R}^+ \times C_A \times \{inside, outside\}$ where $\epsilon \in \mathbb{R}^+$ is the remaining node energy. For the case of personal devices, $app \in C_A$ is the type of the application currently running on the foreground and $e_c \in \{inside, outside\}$ indicates whether the device is stored (e.g., in a pocket or a bag). As in the case of the types of sensors, an application ontology $\mathbb{O}_{\mathbb{A}} = (C_A, R_A)$ is used to describe the different types of applications that can be executed by the devices. For instance, for a smartphone s that at time t is stored in a bag, has 50% of remaining battery charge and is not currently running an application on the foreground, we would have $\kappa^t(s) = (0.5, -, inside)$.

A sensing task is implemented by means of a set of *sensing requests* that are issued to the Fog by one or more devices. A *sensing request* issued by node u at time t is a 5-tuple of the form $\langle f_{id}, r_s, R, n, sink \rangle$, where f_{id} is the identifier of the request, $r_s \in C_S$ is the requested type of sensor, $R = \langle pos, d_{max}, l_{max}, \phi \rangle$ is a 4-tuple that defines the *restrictions* that a node needs to fulfill in order to be considered a *feasible* data source, n is the maximum number of devices that will generate data, and *sink* is the identifier of the Fog data sink. The restrictions R include the geographic area of interest of the sensing request (a disk of radio d_{max} with center at $pos = p^t(u)$), the maximum number of hops (l_{max}) that the request will be propagated, and a context predicate

$(\phi = (\epsilon, app, e_c) \in \mathbb{R}^+ \times C_A \times \{inside, outside\})$ that defines the required device context, which includes the acceptable remaining energy level ϵ, the type *app* of application running on the foreground of the sensing device and whether the device can be stored. When $n > 1$, the requesting node will be in charge of coordinating a collaborative sensing schedule where the selected nodes will monitor the required variables at intervals defined by a sensing schedule.

3.2. Semantic Distance. A *semantic distance function* between two concepts of an ontology $\mathbb{O} = (C, R)$, denoted by $sd_{\mathbb{O}}$, is a function $sd_{\mathbb{O}} : C \times C \longrightarrow \mathbb{R}^+$ that assigns a positive real value to any pair of concepts depending on how taxonomically close the two concepts are in the ontology [27] (here we have dropped the subindex S of $\mathbb{O}_{\mathbb{S}} = (C_S, R_s)$ because we are not referring to a particular ontology). We use such a function to assess how appropriate is a given sensing hardware to fulfill a sensing request. In particular, if the value of the semantic distance between the requested type of sensor and the sensing hardware installed on the device is small, we say that the device is adequate to fulfill the sensing request. From the example of Figure 2, a node equipped with either a *condenser_microphone* or a *carbon_microphone* is adequate to fulfill a request for a *Microphone* because both concepts are more concrete instances of the more abstract concept *Microphone*.

The definition of semantic distance between two concepts can be extended in a number of ways to obtain a semantic distance $sD_{\mathbb{O}} : C \times \mathcal{P}(C) \longrightarrow \mathbb{R}^+$ between a concept $c \in C$ and a set of concepts $C_i \subseteq C$. In this work, we propose (1) that simply returns the smallest semantic distance between concept c and any of the concepts $j \in C_i$. The idea is that a device is adequate to fulfill a request if the type of any of its hardware components is semantically close to the requested type of sensor. To exemplify this, assume that a node i is equipped with the set of sensors C_i = {*Pressure_sensor, Acoustic_sensor*}, and that a request for a *Pressure_sensor* arrives to the Fog environment. Then, since $sD_{\mathbb{O}}$(*Pressure_sensor*, {*Pressure_sensor, Acoustic_sensor*}) = 0 we can say that node i is adequate to fulfill the request.

Precise specifications of the way in which the proposed semantic distance function is computed are presented in Sections 3.5 and 3.6. Section 3.5 presents the formulation of the semantic distance function when the ontology is codified as a graph, and Section 3.5 presents an equivalent formulation that uses prefix-based labels as parameters.

$$sD_{\mathbb{O}}(c, C_i) = \min_{j \in C_i} sd_{\mathbb{O}}(c, j) \tag{1}$$

3.3. Problem Formulation. From the previous definitions, we can state the *problem of instantiating a sensing Foglet* as follows.

Definition 1. Given an environment composed of a time varying set of devices $V(t)$ located at positions designated by function p^t, a sensor ontology $\mathbb{O}_{\mathbb{S}} = (C_S, R_S)$ describing the different types of sensing hardware, an application ontology $\mathbb{O}_{\mathbb{A}} = (C_A, R_A)$ describing the different types of applications running on the devices, a function η that describes the sensing hardware installed in the devices in $V(t)$, a node context function κ^t that describes the instantaneous context of each device, and a sensing request $SRq = \langle f_{id}, r_s, R = \langle pos, d_{max}, l_{max}, \phi \rangle, n \rangle$ issued by node $u \in V(t)$ at time t; find a set of devices $S \subseteq V(t)$, such that the following conditions hold:

(1) The selected sensing nodes are located inside of the region of interest: $\forall s \in S$, $p^t(s)$ is a point located inside of the circumference of radio d_{max} with center at $pos = p^t(u)$.

(2) Nodes in S have the required energy, are running an adequate type of application on the foreground (if any), and are not stored if it may interfere with the sensing process: $\forall s \in S$, $\kappa^t(s).\epsilon \geq R.\phi.\epsilon \wedge$, $sd_{\mathbb{O}_{\mathbb{S}}}(R.\phi.app, \kappa^t(s).app) = 0 \wedge (\kappa^t(s).e_c = R.\phi.e_c \vee R.\phi.e_c = *)$.

(3) $\forall s \in S$ there is a path $p = u, u_1, \ldots u_i, s$ of length $l \leq l_{max}$ in $G(t)$.

(4) $|S|$ is as large as possible but $|S| \leq n$.

(5) Nodes in S are those equipped with the most adequate sensing hardware, or more specifically, the nodes that minimize

$$\sum_{s \in S} sD_{\mathbb{O}_{\mathbb{S}}}(r_s, \eta(s)) \tag{2}$$

where, as defined in Section 3.1, $\kappa^t(s).\epsilon$ denotes the remaining energy in the battery of node s at time t, $\kappa^t(s).app$ is the application that node s is running on the foreground (if any) at time t, and $\kappa^t(s).e_c$ indicates if node s is stored at time t. Similarly, ϕ is a predicate that defines the required device context which includes the acceptable remaining energy level ϵ, the type *app* of application running on the foreground, and whether the device can be stored e_c or if the latter is irrelevant for the request ($R.\phi.e_c = *$).

This way, a solution to the problem of instantiating a sensing Foglet is a set of nodes S currently in the environment, located in the area of interest (Condition 1), with the adequate context (Condition 2), equipped with the most suitable sensing hardware (Condition 5) and, a set of devices that connect the requesting node u with all the nodes in S through paths in $G(t)$ of length less than l_{max} (Condition 3). Other criteria can be used to compute the paths connecting u with the nodes in S, for instance, paths of Maximum-Minimum residual energy, paths composed of nodes not currently in use by their human users, or paths composed of nodes that have not recently participated in a sensing task.

Please note that the problem formulated at Definition 1 is a search problem where a subset $S \subseteq V(t)$ of feasible nodes is selected to perform a sensing task. Feasibility is defined by Conditions 1–4, whereas Condition 5 establishes that the *best* devices to fulfill the request are those equipped with the *most adequate* sensors for the request, namely, those with the smallest semantic distance between the requested sensor r_s and themselves.

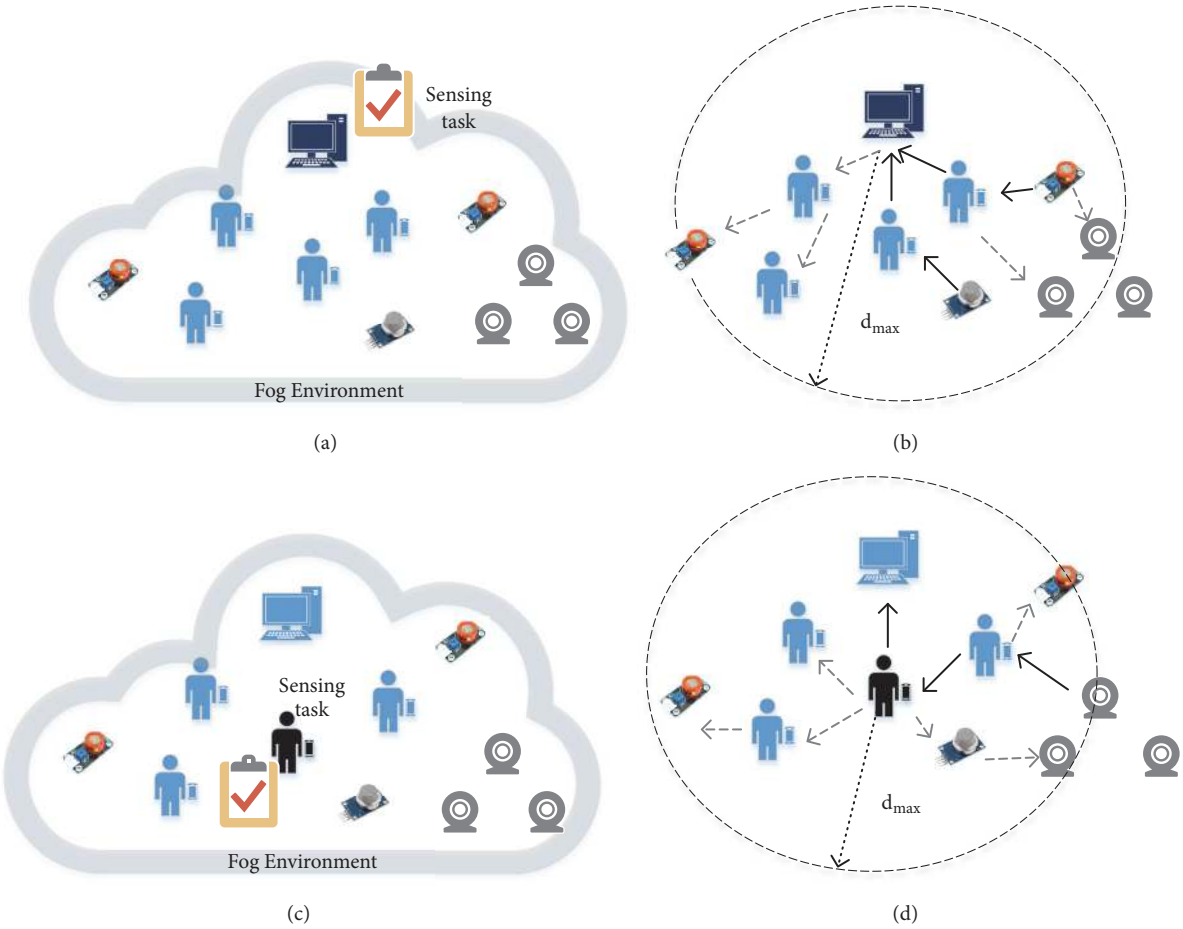

Figure 3: Semantic-driven Foglet formation. (a) A Fog ambient where a sensing task is received from the Cloud by a desktop Fog computer. The Fog ambient is populated by desktop computers, mobile and IoT devices, and sensor nodes. (b) A sensing Foglet composed of sensor and relay nodes, located inside of the region of interest, is instantiated on-demand as a response to the sensing request received by the desktop computer. (c) A Fog ambient where a sensing task is issued by a local mobile device. (d) The Foglet transports the collected sensory data to a desktop Fog computer. Dotted arrows indicate *parent-child* pointers that define BFS spanning trees that are used to collect information about the sensors in the environment. Solid arrows indicate wireless links used to transport sensory data from sensors to the designated sinks.

3.4. Semantic-Driven Foglet Formation. The proposed semantic-driven distributed framework for the instantiation of sensing Foglets is composed of a distributed *semantic-driven sensor selection algorithm*, a *sensor ontology*, an *application ontology*, a semantic distance function that can be computed without accessing the whole ontology, a distributed protocol for implementing collaborative sensing plans, and an interest-driven [28, 29] routing algorithm.

During the first phase of the sensor selection algorithm, a *Foglet Request* (*FReq*) is disseminated across the Fog environment establishing a breadth-first search tree rooted at the requesting device. The request is disseminated only among devices located inside of the region of interest defined by the request and up to the maximum number of hops (l_{max}). At the end of this phase, every node in the tree knows if it is a *feasible device*, namely, if it complies with the restrictions defined in the request.

During a second phase of the algorithm, starting at the leaves and up to the root, nodes inform their parents in the tree of the best feasible devices they have seen. A device is considered better than another device if the semantic distance between the requested type of sensor and any of the sensors in the device is smaller, or if the semantic distance is the same but the path connecting the root with the device is better in terms of hop length. This way, the node issuing the request receives only the information regarding the set of best feasible devices available in the Fog environment. With this information, the requesting node can determine if the selected devices are in fact adequate to implement an effective sensing Foglet.

Figures 3(a)–3(d) illustrate the previous idea. Figure 3(a) shows a Fog environment composed of a heterogeneous set of devices that are connected through wireless links. In the figure, a sensing task is received by a Fog server from the Cloud. As a response, the Fog server issues a series of Foglet requests that are disseminated inside the region of interest. If more than one device is selected to generate sensory data, the requesting node will coordinate the selected

devices to implement a collaborative sensing plan where the selected devices and the nodes connecting them share the computing and communication load. For instance, the requesting node can compute and deliver sensing schedules to the set of selected devices so that they can turn off their radios or sensing hardware to preserve energy and bandwidth. Alternatively, the sink can also request sensory information from all the selected devices to improve the quality of the information through the redundant sources. Figures 3(b) and 3(d) show the BFS-trees established over the nodes located inside of a circular region of interest of radio d_{max}.

Figure 3(c) illustrates the case where a sensing request is issued by an application running on a mobile device. The request in this example designates a Fog server as the data sink (Figure 3(d)) so that data can be collected even if the mobile device is turned off or if it moves out of the Fog environment. Once the best sensors have been identified, interest-driven routing [28, 29] is used to transport the sensory data to the designated sink (Figure 3(d)).

The following sections present detailed specifications of the proposed algorithms and protocols.

3.5. Asymmetric Semantic Distance. We propose an asymmetric semantic distance function $sd_{\mathbb{O}} : C \times C \longrightarrow \mathbb{R}^+$ defined over the concepts of an ontology $\mathbb{O} = (C, R)$ that can be efficiently computed, even without the need of accessing the whole ontology. The objective of this semantic distance function is to serve as the key metric to identify the best available sensor to fulfill a sensing request and, for the case of personal devices, to determine if an adequate application is running on the foreground. Equation (3) shows the definition of the proposed $sd_{\mathbb{O}}$ where we can observe that given a pair of concepts $i, j \in C$, the function $sd_{\mathbb{O}}(i, j)$ equals 0 if the requested concept i is an hypernymy (an ancestor) of concept j. This is because j *is-a* more specific instance of i and hence, it can be used to fulfill the request. Otherwise, the value of $sd_{\mathbb{O}}(i, j)$ is computed as the length of the path from i to the first common ancestor of i and j in the ontology, which is denoted by $d_h(i, j)$. This semantic distance function is similar to the one proposed by Rada et-al [30] in the sense that both functions are based on the hop distance between concepts in the ontology. The proposed distance, however, is an asymmetric version aimed at evaluating to what extent an instance of concept j can be used as an instance of concept i.

$$sd_{\mathbb{O}}(i, j) = \begin{cases} 0 & \text{if } i \text{ is an hypernymy of } j \text{ or } i = j, \\ d_h(i, j) & \text{otherwise.} \end{cases} \quad (3)$$

3.6. Asymmetric Semantic Distance over Prefix-Based Labels. In order to make the computation of (3) more efficient, every concept in $\mathbb{O}$ is mapped onto a prefix-based label that is derived from a finite alphabet Σ. A prefix label L_c for concept $c \in C$ is a word in Σ^* such that $L_c = L_p \oplus l$, where L_p is the prefix label of the parent concept of c in $\mathbb{O}$, $\oplus$ is the concatenation operator, and $l \in \Sigma$ is a suffix assigned to c, which is different to the suffixes assigned to its sibling concepts. The root concept of $\mathbb{O}$ is assigned to label L_0 that can be equal to any symbol $l \in \Sigma$. We use function $\Lambda : C \longrightarrow \Sigma^*$ to denote such a labeling. Please note that we can assign prefix labels to all the concepts in linear time by performing a level-order traversal of the ontology.

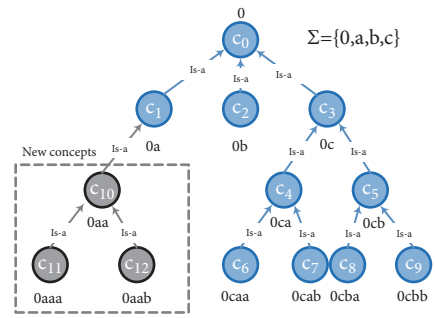

FIGURE 4: Example of a small ontology and its corresponding prefix-based labeling.

Now, we can reformulate (3) in terms of the prefix labels. We use $|L|$ to denote the length in symbols of label L and $L_1 \oslash L_2$ to denote the label composed of the largest common prefix between L_1 and L_2. The new formulation is shown in (4). Please note that (4) can be easily implemented using simple string operations.

$$sd_{\Lambda(\mathbb{O})}(L_i, L_j) = \begin{cases} 0 & \text{if } L_i \oslash L_j = L_i, \\ |L_i| - |L_i \oslash L_j| & \text{otherwise.} \end{cases} \quad (4)$$

As we have already mentioned, the main advantage of this formulation is that devices do not need to access the whole ontology in order to compute the semantic distance between concepts. They only need to perform simple string operations over the corresponding prefix-based labels. Moreover, devices can use (4) to correctly compute the semantic distance between the labels of their sensors and a label of a concept that was added to the ontology to accommodate a new type of sensor. This discussion also applies to the applications running on the devices.

Similar to the case of (1), we can define a semantic distance function $sD_{\Lambda(\mathbb{O})} : \Lambda(\mathbb{O}) \times \mathcal{P}(\Lambda(\mathbb{O})) \longrightarrow \mathbb{N}^+$ between a prefix label $L_i \in \Lambda(\mathbb{O})$ and a set of prefix labels $L_I \subseteq \mathcal{P}(\Lambda(\mathbb{O}))$, where $\mathcal{P}(\Lambda(\mathbb{O}))$ denotes the power set of prefix labels in the labeling Λ. This is shown in (5).

$$sD_{\Lambda(\mathbb{O})}(L_i, L_I) = \min_{L_j \in L_I} sd_{\Lambda(\mathbb{O})}(L_i, L_j) \quad (5)$$

Figure 4 shows a small ontology and its corresponding prefix-based labeling derived from the alphabet $\Sigma = \{0, a, b, c\}$. From the figure we can see that $sd_{\Lambda(\mathbb{O})}(L_{C_3}, L_{C_9}) = 0$ because $L_{C_3} \oslash L_{C_9} =$ "0c" $= L_{C_3}$. Similarly, that $sd_{\Lambda(\mathbb{O})}(L_{C_9}, L_{C_3}) = 2$ because $|L_{C_9}| - |L_{C_9} \oslash L_{C_3}| = |$"0cbb"$| -$ |"0c"$| = 2$. Now, let us assume that the concepts located inside of the dashed box (C_{10}, C_{11}, and C_{12}) were added to accommodate new types of sensors. We can see that a device equipped with a sensor of type C_9 can reply to a request

for a sensor of type C_{12} with a distance $sd_{\Lambda(\mathbb{O})}(L_{C_{12}}, L_{C_9}) = |\text{“0aab”}| - |\text{“0”}| = 3$ because it only needs the labels of the two concepts.

Please note that for any ontology $\mathbb{O} = (C, R)$ with a tree topology, the length $|L_i|$ of the prefix-based label of any $i \in C$ is in $O(h)$ where h is the height of the ontology tree. Therefore, the space needed to store a prefix-based label and the number of operations needed to compute Equation (4) are also in $O(h)$.

3.7. Semantic-Driven Sensor Selection Algorithm. As already mentioned, a sensing task is implemented by means of a set of *sensing requests* that are issued to the environment by one or more devices. A *sensing request* is a 5-tuple of the form $\langle f_{id}, r_s, R, n, sink\rangle$, where f_{id} is the identifier of the request, $r_s \in C_S$ is the requested sensor type codified as a prefix label, $R = \langle pos, d_{max}, l_{max}, \phi\rangle$ defines the restrictions that nodes need to fulfill in order to be considered feasible data sources, n is the maximum number of devices that will generate sensory data, and *sink* is the identifier of the sink node.

As shown in Algorithm 1, when a node receives a *sensing request* from the upper layers, it creates a new element in the set of Foglets (*Flet*, line (4)) that contains the fields and data structures that are used during the process of establishing and maintaining the sensing Foglet. All this information is soft state. An element of the Foglet set contains

(i) a unique Foglet identifier (f_{id}),

(ii) the identifier of the root node of the tree (*root*),

(iii) the identifier of the sink (*sink*),

(iv) the identifier of the parent of the current node in the tree (*parent*), which in the case of the node issuing the request is its own node identifier,

(v) a monotonically increasing sequence number (*sn*) that is further used to maintain the routes from nodes to the root node,

(vi) the requested sensor type (r_s),

(vii) the maximum number of devices that will generate sensory data (n),

(viii) the set of restrictions (R),

(ix) a subset of one-hop neighbors that are known to be children of the current node in the tree (*Child*),

(x) a subset of one-hop neighbors that are known not to be children of the current node in the tree (*nChild*),

(xi) the set of current one-hop neighbors (N_{ini}),

(xii) a set containing the replies (*Reps*) received from the children nodes in the tree,

(xiii) a flag that indicates if the node has already sent a *Foglet Reply* back to its parent (*replied*) and,

(xiv) a routing table (*RT*) that contains pairs $\langle nodeId, nextHop\rangle$, which are used to route packets from the root to the selected nodes.

A reply $r \in Reps$ is a 3-tuple that contains the identity of the node (*nodeId*) that originally issued the reply, the semantic distance (*distS*) between the requested sensor type and the sensors in the node, and the hop distance (*hdist*) from the root to that particular node.

Once the local state has been created, the root node transmits to its neighbors a *Foglet Request* which is an 8-tuple of the form $FReq = \langle f_{id}, root, sn, hdist, r_s, R, n, sink\rangle$, where f_{id} is the Foglet identifier, *root* is the identifier of the root node, *sn* is the root's sequence number, *hdist* is the hop distance that the *FReq* has traversed so far, r_s is the requested sensor type, R is the set of restrictions, n is the maximum number of devices that will be used to generate sensory data, and *sink* is the identifier of the sink. Note that the pair $\langle f_{id}, root\rangle$ uniquely identifies the Foglet that will be instantiated in response to the sensing request. Upon reception of a *FReq* from neighbor j, node i first checks if it is the first time it has received a request with the same $\langle f_{id}, root\rangle$ pair (line (8)). If so, it creates a new element in the set of Foglets and checks whether or not it has to relay the *FReq* message by verifying if it is inside the region of interest (by calling function $inRofI()$, line (11)) and, if the *FReq* has not reached the maximum number of hops ($R.l_{max}$). If node i has already received another Foglet Request with the same $\langle f_{id}, root\rangle$ pair, it checks if it is a leaf of the tree by comparing its set of nonchild neighbors (*nChild*) against its one-hop neighborhood at the time it received the *FReq* for the first time (line (30)).

Nodes know their one-hop neighborhood, denoted $N(u)$, by periodically exchanging hello packets that contain the identity of the nodes.

In order to ensure termination in the presence of packet loss due to either channel effects or topological changes, nodes start a timer (line (13)) after they have relayed a *Foglet Request*. The value of the timer is proportional to the difference between $R.l_{max}$ and the node's hop distance to the root. When this timer expires, and if the node has not already done so, the node sends its *Foglet Reply* back to its parent. This way, in the worst case, the tree will be contracted from leaves to root in a time proportional to $R.l_{max}$.

The contraction phase of the sensor selection algorithm starts when a *FReq* either reaches its maximum number of hops or when it is received by a node located outside of the region of interest. In the first case, the node considers itself a leaf of the tree and sends a *Foglet Reply* message back to its parent containing its own reply (line (19) of Algorithm 1). When the node does not fulfill the context restrictions defined in the Foglet Request ($validaC(R.\phi) = false$), it sets the value of the semantic distance to ∞ to indicate that it does not have a feasible sensor (line (22)). If the node is located outside of the region of interest ($inRofI(R.pos, R.d_{max}, p^{now}(i)) = false$), it also sends a *Foglet Reply* containing a value of semantic distance equal to ∞ (line (22)). A node can also detect that it is a leaf if it receives *FReq* from all of its current neighbors indicating other nodes as their parents (lines (27) and (30)).

As shown in Algorithm 2, when a node i receives a *Foglet Reply* $FRep = \langle f_{id}, root, rep\rangle$ from one of its children (node j), it stores the replies *rep* contained in the message in the *Reps* data structure and adds j's identifier to the *Child* data structure. Then, for each of the replies contained in

(1) **when** $SR = \langle f_{id}, r_s, R, n, sink\rangle$ **is received from** upper layer **do**
(2) $nChild \longleftarrow \emptyset$; $Child \longleftarrow \emptyset$; $Reps \longleftarrow \emptyset$; $N_{ini} \longleftarrow N(i)$;
(3) $RT \longleftarrow \emptyset$; $sn \longleftarrow sn + 1$;
(4) $Flet \longleftarrow Flet \cup \{\langle f_{id}, i, sink, sn, i, ..., N_{ini}, Reps, false, RT\rangle\}$
(5) **Broadcast**$(FReq = \langle f_{id}, i, sn, 0, r_s, R, n, sink\rangle)$;
(6) **end when**
(7) **when** $FReq = \langle f_{id}, root, sn, hdist, R, n\rangle$ **is received from** j **do**
(8) **if** $\langle f_{id}, root, *, ..., *\rangle \notin Foglet$ **then**
(9) $nChild \longleftarrow \{j\}$; $Child \longleftarrow \emptyset$; $Reps \longleftarrow \emptyset$; $N_{ini} \longleftarrow N(i)$; $RT \longleftarrow \emptyset$;
(10) $Flet \longleftarrow Flet \cup \{\langle f_{id}, root, sink, j, ..., RT\rangle\}$
(11) **if** $hdist + 1 < R.l_{max} \wedge inRofI(R.pos, R.d_{max}, p^{now}(i))$ **then**
(12) **Broadcast**$(FReq = \langle f_{id}, root, j, hdist + 1, r_s, R, n\rangle)$;
(13) **StartTimer**$(RepTOut = \langle f_{id}, root\rangle, (R.l_{max} - hdist + 1)\tau)$;
(14) **else**
(15) **if** $inRofI(R.pos, R.d_{max}, p^{now}(i)) \wedge validaC(R.\phi)$ **then**
(16) $f \longleftarrow Flet.getFoglet(f_{id}, root)$;
(17) $f.replied \longleftarrow true$;
(18) $FRep = \langle f_{id}, root, \{sD_{\Lambda(\mathbb{O})}(r_s, \eta(i)), i, hdist + 1\}\rangle$;
(19) **Unicast(parent)**$(FRep)$;
(20) **else**
(21) $FRep = \langle f_{id}, root, \{\infty, i, \infty\}\rangle$;
(22) **Unicast(parent)**$(FRep)$;
(23) **end if**
(24) **end if**
(25) **else**
(26) $f \longleftarrow Flet.getFoglet(f_{id}, root)$;
(27) **if** $parent \neq i$ **then** $f.nChild \longleftarrow f.nChild \cup \{j\}$;
(28) **end if**
(29) $Flet \longleftarrow (Flet \setminus \{f_{id}, root, *, ..., *\}) \cup \{f\}$;
(30) **if** $(f.N_{ini} \cap N(i))\backslash(f.Child \cup f.nChild) = \emptyset$ **then**
(31) $f.replied \longleftarrow true$;
(32) **if** $inRofI(f.R.pos, f.R.d_{max}, p^{now}(i)) \wedge validaCxt(f.R.\phi)$ **then**
(33) **if** $f.parent \neq i$ **then**
(34) $FRep = \langle f_{id}, root, \{sD_{\Lambda(\mathbb{O})}(r_s, h(i)), i, f.hdist + 1\}\rangle$;
(35) **Unicast(parent)**$(FRep)$;
(36) **else SendUpperLayer**$(f_{id}, sD_{\Lambda(\mathbb{O})}(r_s, \eta(i))$;
(37) **end if**
(38) **else**
(39) **if** $f.parent \neq i$ **then**
(40) **Unicast(parent)**$(FRep = \langle f_{id}, root, \{\infty, i, \infty\}\rangle)$;
(41) **elseSendUpperLayer**(f_{id}, ∞);
(42) **end if**
(43) **end if**
(44) **end if**
(45) **end if**
(46) **end when**

Algorithm 1: FogletMessageHandler.

the message reporting a semantic distance different to ∞, i creates an entry in the routing table (RT) establishing that node j can be used as a next-hop to reach the selected node through a shortest path (line (8)). If the receiving node has collected Foglet Replies from all of its children nodes that remain in its one-hop neighborhood, it selects the best received replies (line (14)) to create its own reply message that will be forwarded to either its parent in the tree or to upper layers if the current node is the root node. The node also includes its own information in the reply if it has a feasible sensor, namely, if $validaC(f.R.\phi) = true$ (line (11)).

3.8. Data Retrieval, Path Maintenance, and Collaborative Sensing Plans. Once the root node has received Foglet replies from all the nodes located inside the region of interest, it sends a *Data Control* ($DCtr = \langle nodeId, f_{id}, root, sensor, command, parameters\rangle$) packet to the selected nodes to initiate the sensory data flows. $DCtr$ packets are routed from the root to the intended nodes following the *nextHop*

(1) **when** $FRep = \langle f_{id}, root, rep\rangle$ **is received from** j **do**
(2) $f \longleftarrow Flet.getFoglet(f_{id}, root)$;
(3) **if** f.replied = true **then return**;
(4) **end if**
(5) $f.Reps \longleftarrow f.Reps \cup rep$;
(6) $f.Child \longleftarrow f.Child \cup \{j\}$;
(7) **for all** $r \in rep : r.distS < \infty$ **do**
(8) $f.RT \longleftarrow f.RT \cup f.RT\{\langle f.nodeId, j\rangle\}$;
(9) **end for**
(10) **if** $(f.N_{ini} \cap N(i)) \setminus (f.Child \cup f.nChild) = \emptyset$ **then**
(11) **if** $validaC(f.R.\phi) = true$ **then**
(12) $f.Reps \longleftarrow f.Reps \cup \{sD_{\Lambda(\mathbb{O})}(f.r_s, \eta(i)), i, f.hdist\}$;
(13) **end if**
(14) $Best \longleftarrow$getBest($f.Reps, f.n$); $f.replied \longleftarrow true$;
(15) **if** $f.parent \neq i$ **then Unicast(parent)**$(FRep = \langle f_{id}, root, Best\rangle)$;
(16) **else SendUpperLayer**$(f_{id}, Best)$;
(17) **end if**
(18) **end if**
(19) $Flet \longleftarrow (Flet \setminus \{f_{id}, root, *, ..., *\}) \cup \{f\}$;
(20) **end when**
(21) **when** $RepTOut = \langle f_{id}, root\rangle$ **is received from** itself **do**
(22) $f \longleftarrow Flet.getFoglet(f_{id}, root)$;
(23) **if** $f.replied = true$ **then return**;
(24) **end if**
(25) **if** $validaC(f.R.\phi) = true$ **then**
(26) $f.Reps \longleftarrow f.Reps \cup \{sD_{\Lambda(\mathbb{O})}(f.r_s, \eta(i)), i, f.hdist\}$;
(27) **end if**
(28) $f.replied \longleftarrow true$; $Best \longleftarrow$getBest($f.Reps, f.n$);
(29) **if** $f.parent \neq i$ **then Unicast(parent)**$(FRep = \langle f_{id}, root, Best\rangle)$;
(30) **else SendUpperLayer**$(f_{id}, Best)$;
(31) **end if**
(32) **end when**

Algorithm 2: FogletMessageHandler $_{cont...}$.

pointers stored at the routing tables (RT) during the contraction phase of the sensor selection algorithm. In a $DCtr$ packet, *sensor* is the prefix label of the selected sensor, $command \in \{startFlow, continueFlow, stopFlow\}$ instructs the sensor to start, continue, or end the data flow, and *parameters* include flow and sensor specific parameters such as data rate, sampling period, or resolution.

In the cases where the root node is also the sink, and for as long as it requires data from any device, the root node refreshes the routing structures used to connect it with those particular devices by means of *Sink Announcement* messages. These messages are periodically disseminated along the regions of interest related to all the Foglets for which the node acts as a sink. Sink Announcement messages establish an ordering over the nodes based on monotonically increasing sequence numbers, the hop distance to the root (sink) node, and the identifiers of the nodes. A node selects a neighbor as next-hop to the root if it reports the largest sequence number or the same sequence number but a shorter distance to the root.

In the cases where the designated sink is different from the root node, interest-driven routing is used to transport the sensory data and the data control messages.

The collaborative sensing plans are implemented by the sink nodes by means of *Data Control* packets. A sink can, for instance, implement a round robin schedule where a single node is selected at each time to sense a given variable in order to save battery. Alternatively, a sink can instruct more than one sensing device to sense the environment at the same time in order to have redundancy for the sake of increased data accuracy and resolution.

4. Analysis

In this section, we present a set of theorems and a corollary that establish the correctness of the algorithms presented in Section 3.7 and that (4) correctly computes the value of (3).

Theorem 2. *Given an ontology* $\mathbb{O} = (C, R)$ *with a tree topology and a prefix-based labeling* Λ *derived from a finite alphabet* Σ. *Then,* $sd_{\mathbb{O}}(i, j) = sd_{\Lambda(\mathbb{O})}(\Lambda(i), \Lambda(j))$ *for any* $i, j \in C$.

Proof. We have three cases. (1) i is an ancestor (hypernymy) of j. Since $\mathbb{O}$ is a tree, the only path connecting the root concept r with j includes concept i, and by construction, label $\Lambda(i)$ is a prefix of label $\Lambda(j)$. Therefore, $\Lambda(i) \oslash \Lambda(j) = \Lambda(i)$ and $sd_{\Lambda(\mathbb{O})}(\Lambda(i), \Lambda(j)) = sd_{\mathbb{O}}(i, j) = 0$. (2) j is an ancestor

(hypernymy) of i. We have that $\Lambda(i) \oslash \Lambda(j) = \Lambda(j) \neq \Lambda(i)$ and therefore $sd_{\Lambda(\mathbb{O})}(\Lambda(i), \Lambda(j)) = |\Lambda(i)| - |\Lambda(j) \oslash \Lambda(j)|$. Now, since j is an ancestor of i, it is also the first common ancestor of i and j and therefore $d_h(i, j) = |\Lambda(i)| - |\Lambda(j)|$ which is the length in hops of the path from i to j in the ontology. (3) Neither i nor j is an ancestor of the other. We also have that $\Lambda(i) \oslash \Lambda(j) \neq \Lambda(i)$. By construction of the prefix-based labeling Λ, the difference in length between the label $\Lambda(i)$ of concept i and the label of any of its ancestors is equal to its distance in the ontology; therefore, $|\Lambda(i)| - |\Lambda(i) \oslash \Lambda(j)|$ is equal to $d_h(i, j)$, because, by definition, $\Lambda(i) \oslash \Lambda(j)$ is the prefix label of the first common ancestor between i and j. □

For the following theorems and corollary, we assume that packets sent in unicast mode are reliably delivered to their intended destinations and that every node $i \in V(t)$ knows the constituency of its one-hop neighborhood at time t, which is denoted by $N^t(i)$. We will use $T = (V_T, E_T)$ to denote the BFS-three composed of active links at the end of the contraction process that was induced by the *parent* pointers established during the dissemination of the Foglet Request (*FReq*).

Theorem 3. *When the algorithm described in Algorithms 1 and 2 terminates, the set Best sent to upper layers is composed of a subset of the nodes of the BFS-tree $T = (V_T, E_T)$ that comply with the five conditions of Definition 1.*

Proof.

Condition 1. We have to show that $\forall i \in Best$, $(FReq.R.pos.x - p^{FReq}(i).x)^2 + (FReq.R.pos.y - p^{FReq}(i).y)^2 \leq (d_{max})^2$, where $p^{FReq}(i)$ denotes the position of node i at the time it received the *FReq*. From line (11) of Algorithm 1 we know that nodes located outside of the region of interest do not propagate the *FReq* and reply with a value of semantic distance equal to ∞ (lines (22) and (40) of Algorithm 1) to indicate that they are not feasible nodes. Therefore, no node $i \in Best$ was located outside of the region of interest at the time it received the *FReq*.

Condition 2. To show that the selected nodes have a feasible context, we have to show that $\forall i \in Best$, $\kappa^{FReq}(i).\epsilon \geq FReq.R.\phi.\epsilon \wedge sd_{\Lambda(\mathbb{O}_s)}(FReq.R.\phi.app, \kappa^{FReq}(i).app) = 0 \wedge (\kappa^{FReq}(i).e_c = FReq.R.\phi.e_c \vee FReq.R.\phi.e_c = *)$. Similar to the previous case, nodes call to $validaC(c.R.\phi)$ to check if their context $\kappa^{FReq}(i)$ comply with the restrictions defined in $FReq.R.\phi$ at the time the *FReq* was received. If not, they reply with a value of semantic distance equal to ∞ to indicate that they are not feasible nodes.

Condition 3. We have to show that at the time the Foglet is established, $\forall i \in Best$, the length of the path $root, \ldots, i$ connecting a node i to the root in the BFS-tree is less than or equal to $C.Req.R.l_{max}$. Assume there is a path $root, \ldots, i$ of length $|root, \ldots, i| > C.Req.R.l_{max}$. This is not possible because *FReq* packets are propagated at most $C.Req.R.l_{max}$ hops away from the root, and the paths connecting the root to the selected devices are reverse paths established using the parent pointers defined during the propagation of the *FReq*.

Condition 4. We have to show that $|Best|$ is as large as possible but $|Best| \leq n$. Assume $|Best| < n$ and that at the time the algorithm terminates, there is a node b that complies with Conditions 1, 2, and 3 in the BFS-tree but that $b \notin Best$. Since the algorithm terminates and unicast messages are reliable, the branch of the BFS-tree containing b should have completed its contraction process with the transmission of a *FRep* message to the root node containing a set *rep* of feasible nodes. Now, since at the root node $|Best| < n$, it should be the case that the number of feasible nodes in *rep* is also smaller than n. Otherwise, the extra nodes could be included in *Best*. On the other hand, the information regarding b should have been omitted, either by b itself or by one of its ancestors in the BFS-tree. This means that at some point during the contraction of the branch containing b, a node decided not to include b in the replay. But this is only possible if that node had information about other n feasible nodes which are better than b. From this point on, all the *FRep* messages in that branch would contain information of at least n feasible nodes which is a contradiction.

Condition 5. We have to show that $\forall S \in \mathcal{P}(V_T)$ such that every node in S complies with Conditions 1, 2, 3, and 4, we have that $\sum_{s \in Best} sD_{\Lambda(\mathbb{O}_s)}(r_s, h(s)) = \min_{S \in V_T}\{\sum_{s \in S} sD_{\Lambda(\mathbb{O}_s)}(r_s, h(s))\}$. Assume there is a feasible node $b \in V_T$ but not in *Best* at the root node, such that $sD_{\Lambda(\mathbb{O}_s)}(r_s, h(b)) < sD_{\Lambda(\mathbb{O}_s)}(r_s, h(s))$, for some $s \in Best$. Since the algorithm terminates and unicast messages are reliable, the branch of the BFS-tree containing b should have completed its contraction process with the transmission of a *FRep* message to the root node containing a set *rep* of feasible nodes. Since $b \notin Best$ at the root, it should be the case that at some point during the contraction of the branch containing b, a node decided not to include b in the replay. But this is only possible if that node had information about other n feasible nodes which are better than b. This leads to a contradiction since from this point on, only better nodes can be added to *FRep*s sent towards the root. □

Theorem 4. *All the branches of the BFS-tree $T = (V_T, E_T)$ are correctly contracted.*

Proof. By contradiction, assume that there is at least one branch of the BFS-tree $T = (V_T, E_T)$ that is not correctly contracted up to the root. Let node $s \in V_T$ be the first node that did not send a *FRep* message to its parent in T and let $p \in V_T$ be its parent node. Note that $s \neq p$ since we are assuming the branch did not finish its contracting process. Now, by lines (30) of Algorithm 1 and (10) of Algorithm 2, node s sends a *FRep* message to p when $N^{FReq}(s) \cap N^{current}(s) = S_{Child} \cup S_{nChild}$, where $N^{FReq}(s)$ is the one-hop neighborhood of s at the time it received the first *FReq*, $N^{current}(s)$ is the current one-hop neighborhood of s, S_{Child} is a set of nodes which are known to be children nodes of s in T, and S_{nChild} is a set of nodes which are known not to be children nodes of s in T. Since s adds nodes to S_{Child} when it receives a *FRep* message from them, the conditions of lines (30) of Algorithm 1 and (10) of Algorithm 2 state that a node contracts itself when it has collected *FRep* messages from all of his children nodes which are still within radio range. Since *FRep* messages

are transmitted using a reliable protocol, they always reach their intended destinations (if within radio range); therefore, the only way in which the contraction condition at s never becomes true is when one or more *FReq* messages stating a parent different to s are not correctly received at s. But, since s is not actually waiting for information from those nodes, it can send a *FRep* message when a timer expires as described by lines (21) to (31) of Algorithm 2. Now, we have to argue that the contraction timer at p expires after the contraction timer at s. Let τ_p be the time in which p started its contraction timer and $h\tau$ the value of that timer, with $h \geq 1$. Now, the time in which s started its own timer equals $\tau_p + \tau_{oneHopDelay}$ and its corresponding value equals $(h-1)\tau$, where $\tau_{oneHopDelay}$ is the maximum one-hop delay experienced by a message. Therefore, the expiration time of s' timer is $\tau_p + \tau_{oneHopDelay} + (h-1)\tau$ and this time can be earlier than p's expiration timer only if $\tau_{oneHopDelay} > \tau$. So, by setting the value of τ sufficiently large, the contraction timer at p will expire after the time in which the contraction timer at s expires. □

Corollary 5. *The algorithm described in Algorithms 1 and 2 terminates.*

Proof. This is a direct consequence of Theorem 4 because the algorithm terminates when all the branches of the BFS-tree complete their contraction process. □

5. Experimental Results

In this section, we present detailed simulation results comparing the performance of the Sensing Fog against that of a traditional opportunistic sensing environment in which sensing nodes periodically post their location, instantaneous context, and sensing capabilities to a well-known local *ambient server*. In this centralized environment, when a sensing request arrives at a node, that particular node forwards the request to the ambient server that replies back with a list of devices that can fulfill the requirements specified in the sensing request. Lastly, the requesting node contacts the selected devices to instruct them to generate sensory data. In this sensing environment, all the communications are supported by the AODV [31] routing protocol. This implementation mimics the behavior of many of the centralized environments described in Section II. The algorithms and protocols that implement both sensing environments were developed over the Network Simulator 3 (ns-3) [32] that provides realistic models of the whole protocol stack. In particular, ns-3 provides well-tuned versions of AODV and of the underlying MAC protocol and physical layer. The source code can be downloaded from https://github.com/blunan/Stratos-Distributed and https://github.com/blunan/Stratos-Centralized.

We use *search success ratio*, *satisfied-request ratio*, *packet delivery ratio*, *average reply delay,* and *overhead* as our performance metrics. The *search success ratio* is computed as the ratio between the number of times the sensing environments were able to locate the *best* set of devices (according to Definition 1) and the total number of sensing requests received by any node in the environment. The average *satisfied-request ratio* is computed as the ratio between the number of times the sensing environments located a *valid* set of devices (according to conditions 1 to 4 of Definition 1) and the total number of sensing requests received by any node in the environment. The *average reply delay* is the average time between the arrival of a sensing request and the reception, at the corresponding sink node, of the first data packet containing sensory data. The *overhead* is computed as the total number of bytes transmitted by the nodes in the environment, divided by the number of bytes of sensory data received at the sink nodes.

Table 1: Simulation environment.

Total nodes	100	Node placement	Random
Simulation area	$1000 \times 1000\text{m}^2$	Simulation time	100s
MAC Protocol	802.11b	Tx. rate	2Mbps
Mobility model	Random waypoint	Min.-Max. Vel.	1 - 4m/s
		Pause time	40s

All the algorithms were tested with IEEE 802.11 DCF as the underlying MAC protocol. Nodes move according to the random waypoint mobility model and each simulation was run for fifty different seed values. The confidence level for all the results presented in this section is 95%. Table 1 lists the details of the simulation environment.

The results are organized in two sets. In the first set (Sections 5.1-5.2) we evaluate the performance of the algorithms when the sensing requests require a single source of sensory data, whereas in Section 5.3 the sink nodes coordinate sensing plans that are implemented by a set of devices. In all the experiments, the value of the radio (d_{max}) of the region of interest is selected uniformly at random from the interval $[400m, 600m]$, the maximum number of hops (l_{max}) that the request is propagated equals four, and nodes are equipped with a set of sensors which are selected uniformly at random from the set of sensors defined in the sensor ontology. Nodes receiving sensing requests are also selected uniformly at random from all the nodes in the environment and the length of the data packets generated by the selected sensors is of 256 bytes. Both, the sensor type and the context included in the sensing request are also selected uniformly at random from the sensor and application ontologies and from all the possible values for node contexts, respectively. For the centralized environment, a node is selected as the ambient server uniformly at random at the beginning of the simulation. The ambient server remains static during the whole simulation time. All these values were selected to model scenarios as described in Section 3.

5.1. Performance with Increasing Proportion of Mobile Devices. In these experiments, we evaluate the performance of the sensing environments as the percentage of mobile nodes is increased from 25% to 100% of the nodes in the environment. Every node is assigned with two different types of sensors, there are four concurrent sensing requests, and selected sensors generate data flows composed of 10 data packets. The purpose of this scenario is to evaluate the ability of the

sensing environments to cope with an increasing number of topological changes.

From Figure 5(a) we can observe that the sensing Foglets clearly outperform the centralized environment in terms of both *search success ratio* and *satisfied-request ratio*. As shown in Figure 5(c), the reduced performance attained by the centralized environment is mainly due to the extra communication overhead induced by the nodes while updating its state at the centralized server. Under this situation, a significant number of requests (or replies) are lost due to contention and congestion, which is reflected as a reduced number of successfully replied sensing requests. Moreover, as the number of mobile nodes increases and more links break, the control overhead induced by AODV's route repair mechanisms also increases, creating even more contention and congestion. The net effect is a reduction in the number of successfully replied sensing requests. This is consistent with the fact that the values of the search success ratio and the satisfied-request ratio attained by the centralized environment are almost the same, which indicates that this environment almost always identifies the best sensor, but sometimes it is unable to deliver a reply back to the requesting node.

The results (Figure 5(c)) also show that the *on-demand* approach used by the Foglets is far less costly than the proactive approach of the centralized server. While the overhead induced by the Foglets is consistently less than 30, the overhead induced by the centralized environment reaches 2417. The sensing Foglets attain this level of efficiency because nodes can locally compute semantic distances using the proposed algorithm that is based on labels. In this case, however, there is a gap between the search success ratio and the satisfied-request ratio attained by the Foglets that indicates that they do not always locate the best sensor in the environment. This usually happens when the best node does not correctly receive a Foglet Request (sent in unreliable broadcast mode) and hence, it is not included in the BFS-tree.

Figure 5(b) shows that the average delivery ratio attained by the Foglets is consistently better than that of the centralized environment. Moreover, unlike the centralized environment, the figure also shows that the Foglets are fairly insensitive to the increase in the percentage of mobile nodes. This indicates that the routes connecting sensors to sinks computed and maintained by the Foglets are more effective than those computed and maintained by AODV.

Lastly, from Figure 5(d) we can observe that the *average reply delay* attained by the centralized environment is consistently better than that of the Foglets. This behavior was expected and the reason is as follows. First of all, it is important to recall that this metric considers the time between the moment in which the sensing request arrives at the environment and the time the first sensory data packet is received at the designated sink. Therefore, in the case of the sensing Foglets, this delay includes the time it takes to propagate the request within the environment, the time it takes to contract the BFS-tree established during the dissemination of the Foglet Request and the time it takes to transport the first data packet to the sink. Moreover, as described in Section 3.7 and Theorem 4, due to the conservative way in which the BST-Tree is contracted (needed to guarantee termination), the BFS-tree contraction delay is proportional to the height of the BFS-tree and to the value of τ which is a configuration parameter. On the other hand, due to the proactive approach of the centralized environment, all the information needed to locate a relevant sensor is already stored at the ambient server and hence, it can immediately reply back to an incoming request.

5.2. Performance with Increasing Number of Sensors per Node. In these experiments, we increase the number of sensors per node from 1 to 8, 50% of the nodes are mobile, there are four concurrent sensing requests, and selected sensors generate data flows composed of 10 data packets. The purpose of these experiments is to evaluate the ability of the environments to take advantage of having better-equipped devices.

Figure 6(a) shows the *search success ratio* and the *satisfied-request ratio* attained by the two environments. From the figure, we can observe that the Foglets are able to take advantage of having more nodes equipped with the requested sensors which is reflected in an increased *search success ratio*. This is not the case for the centralized environment where less than 50% of the requests are correctly fulfilled, even though the centralized server almost always identify the best sensor in the environment. As in the previous section, the main reason for this poor performance is the extra communication overhead induced by the proactive way in which the state of the sensors is periodically updated at the ambient server (Figure 6(c)) that tend to congest the network around the server increasing the probability of losing request and reply packets. In these experiments, the overhead induced by the Foglets is always less than 30, while the overhead induced by the centralized environment reaches 3002.

Figure 6(b) shows that the Foglets also outperform the centralized environment in terms of packet delivery ratio by delivering up to 30% more data packets at the sinks. From Figure 6(d) we can observe that the *average reply delay* attained by the centralized environment is consistently better than that of the Foglets. Again, the extra delay experienced by the Foglets is mainly due to the time it takes to contract the BFS-tree.

5.3. Collaborative Sensing Plans. In this set of scenarios, we evaluate the performance of the sensing environments when the sensing requests ask for a set of three devices that will collaborate in order to share the computation and communication load. The three devices implement a sensing plan where a single node is selected at a time to sense a required variable and to send the collected data back to the sink. In both sensing environments, data sinks are in charge of coordinating the sensing nodes by instructing them to start or stop sending sensory data. Every node is assigned with two different types of sensors, there are four concurrent sensing requests, and selected sensors generate data flows composed of 20 data packets.

From Figures 7(a)–7(d) we can observe that the sensing Foglets are capable of effectively and efficiently implementing a collaborative sensing task that is carried out by more

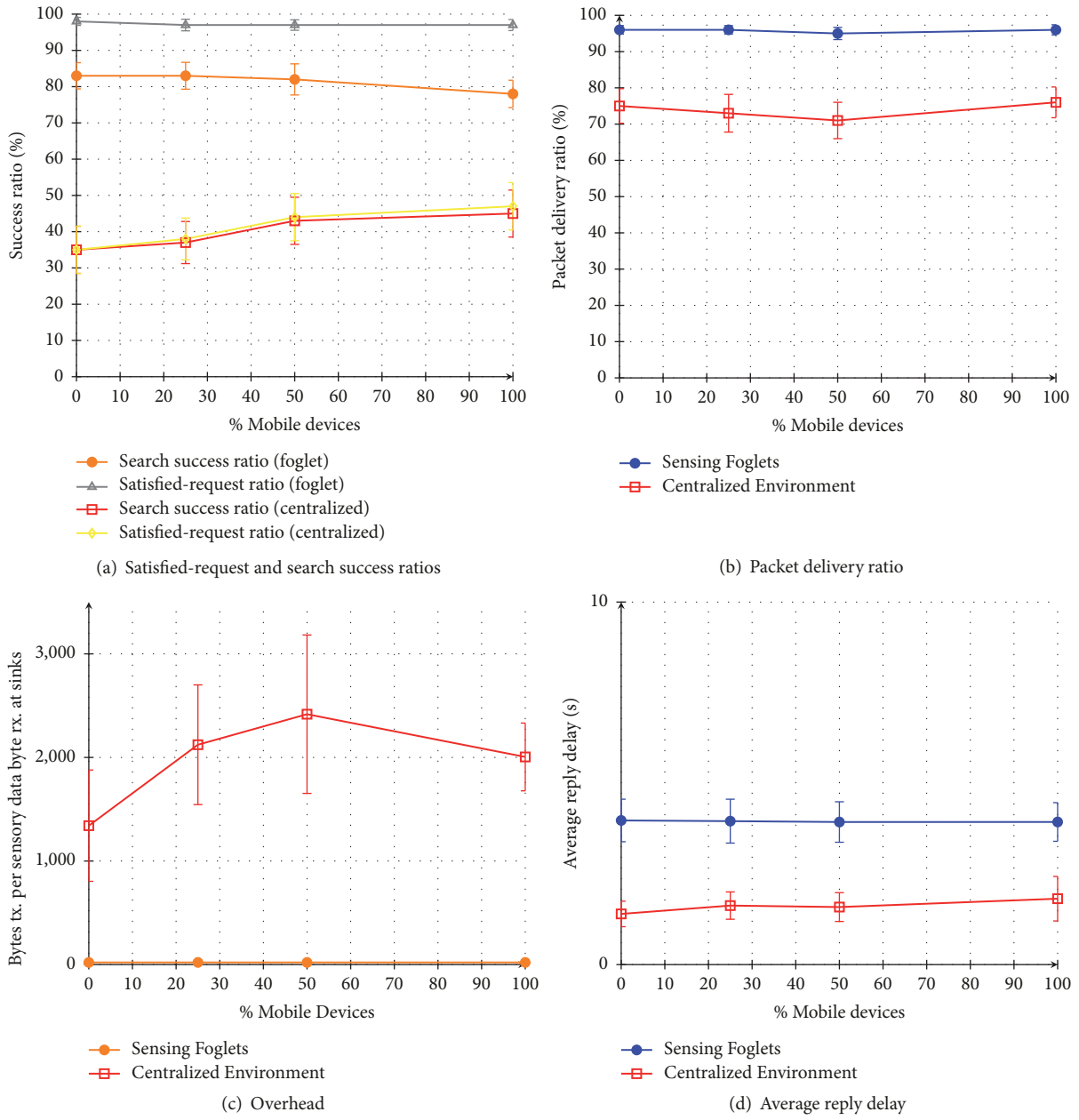

Figure 5: Performance with increasing proportion of mobile devices.

than one device. The results presented by these figures are consistent with those presented in previous sections; specifically, that the Foglets achieve higher search success ratio, satisfied-request ratio, and packet delivery ratio than the centralized environment. This is all while inducing two orders of magnitude less overhead per sensory data byte received at the designated sinks (up to 30 against up to 1408), but with larger average reply delays.

6. Discussion and Future Research

In this paper, we presented a new distributed framework for opportunistic sensing in the Fog, where collections of potentially highly heterogeneous devices organize themselves into *sensing Foglets* to fulfill a sensing request issued to the Fog environment. The proposed framework uses semantic-driven in-network processing to locate the devices that are most fit to perform a given sensing task. It also establishes and maintains multihop paths, connecting the selected sensors to a designated data sink, which are used to deliver flows of sensory data. The sensing Foglets are instantiated in response to sensing requests generated either by an application running on a device currently in the Fog environment or by an application running on the Cloud. In general, the sensing Foglets can extend the sensing capabilities of any device by taking advantage of other devices located at a relevant Fog environment.

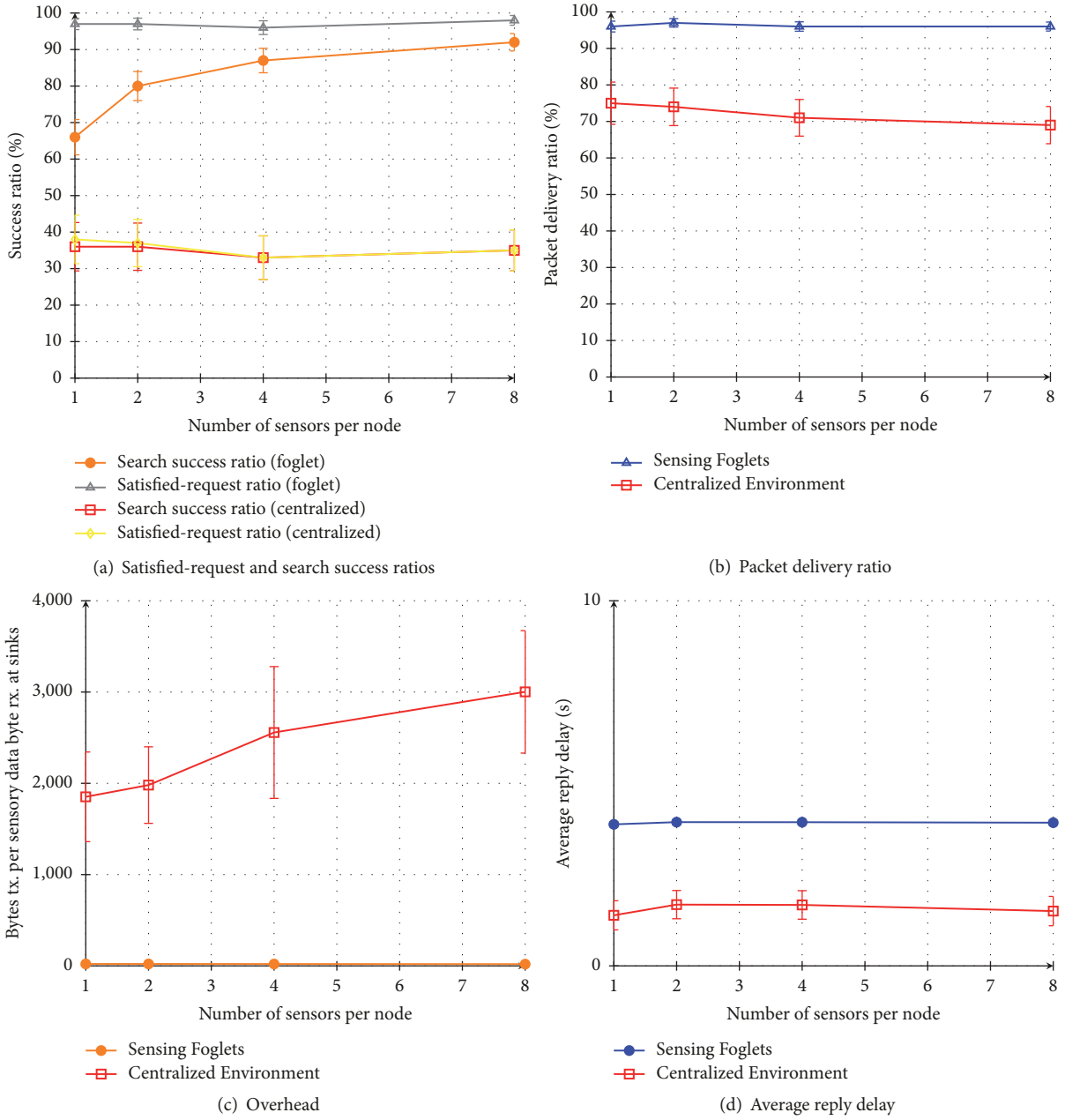

Figure 6: Performance with increasing number of sensors per node.

The fitness of a node is computed in terms of its instantaneous context and the semantic distance between its sensing hardware and the hardware specified in the sensing request. Semantic distances are computed locally at the nodes by means of a very efficient algorithm that takes prefix-based labels as input. The main advantage of this algorithm is that a node does not need to store the whole ontology in order to compute the semantic distance between the requested sensor and its own sensors. It only needs to store the prefix-based labels that codify the branches of the ontology that contain the concepts describing its sensing hardware. Moreover, the devices in the environment can correctly compute the semantic distance between its sensors and new types of sensors that are added to the ontology. This is without the need of updating any information at the devices.

We presented the results of a series of simulation-based experiments that show that the proposed sensing Foglets outperform traditional sensing platforms that are based on centralized services by correctly answering to more sensing request and at a lower communication cost. This is true in scenarios where sensing Foglets incorporate a single sensing device and in scenarios where each sensing Foglet includes a set of nodes that implement collaborative sensing plans in order to distribute the computation and communication load.

Future work includes the development of delay-tolerant sensing Foglets and the use of heterogeneous networks including vehicular ad hoc networks (VANETs).

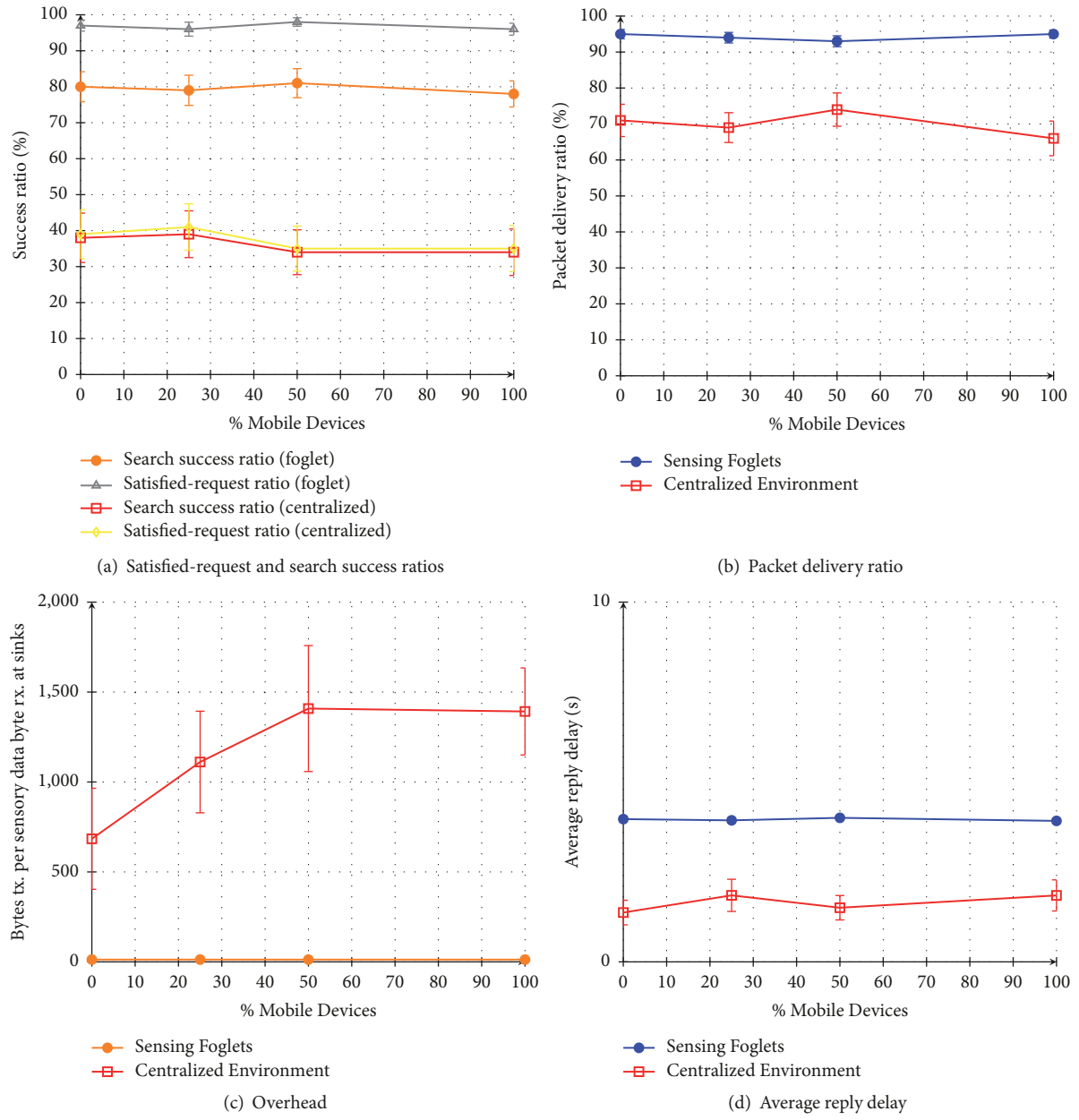

Figure 7: Collaborative sensing plans: performance with increasing proportion of mobile devices.

Lastly, it is important to highlight the fact that authentication, access control, rogue node detection, intrusion detection, privacy, and trust management are crucial aspects that need to be fully addressed before opportunistic sensing platforms are adopted by the general public. The good news is that the Fog computing paradigm provides promising solutions to address many of these security and privacy issues. For instance, Fog nodes can provide cryptographic services to devices that lack the processing power to perform complex computations [33]. Relevant services such as authentication and reputation-based trust management are also good candidates to be implemented by local Fog services running on stable and trusted computers [34]. Privacy policies can also be enforced by Fog nodes that decide which data can be transported outside of the local Fog environment.

Data Availability

The source code used to obtain the experimental results reported in Section 5 can be downloaded from https://github.com/blunan/Stratos-Distributed and https://github.com/blunan/Stratos-Centralized.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was sponsored in part by the University of California MEXUS-CONACyT program under Grant CN 15-1451, by the Mexican National Council for Science and Technology (CONACyT), and by the Mexican National Polytechnic Institute (IPN).

References

[1] L. Atzori, A. Iera, and G. Morabito, "The internet of things: a survey," *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[2] P. G. Lopez, A. Montresor, D. Epema et al., "Edge-centric computing: vision and challenges," *Computer Communication Review*, vol. 45, no. 5, pp. 37–42, 2015.

[3] Y. Yang, "FA2ST: Fog as a Service Technology," in *Proceedings of the 2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)*, pp. 708-708, Turin, July 2017.

[4] CISCO., *Fog computing and the internet of things: Extend the cloud to where the things are*, 2016.

[5] L. M. Vaquero and L. Rodero-Merino, "Finding your way in the fog: Towards a comprehensive definition of fog computing," *Computer Communication Review*, vol. 44, no. 5, pp. 27–32, 2014.

[6] M. Aazam and E.-N. Huh, "Fog computing and smart gateway based communication for cloud of things," in *Proceedings of the 2nd IEEE International Conference on Future Internet of Things and Cloud (FiCloud '14)*, pp. 464–470, Barcelona, Spain, August 2014.

[7] M. Chiang and T. Zhang, "Fog and IoT: an overview of research opportunities," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 854–864, 2016.

[8] A. V. Dastjerdi and R. Buyya, "Fog Computing: Helping the Internet of Things Realize Its Potential," *The Computer Journal*, vol. 49, no. 8, Article ID 7543455, pp. 112–116, 2016.

[9] I. Stojmenovic, S. Wen, X. Huang, and H. Luan, "An overview of Fog computing and its security issues," *Concurrency Computation*, vol. 28, no. 10, pp. 2991–3005, 2016.

[10] F. Jalali, K. Hinton, R. Ayre, T. Alpcan, and R. S. Tucker, "Fog computing may help to save energy in cloud computing," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 5, pp. 1728–1739, 2016.

[11] E. A. Lee, J. Rabaey, B. Hartmann et al., "The swarm at the edge of the cloud," *IEEE Design and Test*, vol. 31, no. 3, pp. 8–20, 2014.

[12] Q. Liang, X. Cheng, S. C. Huang, and D. Chen, "Opportunistic sensing in wireless sensor networks: theory and application," *IEEE Transactions on Computers*, vol. 63, no. 8, pp. 2002–2010, 2014.

[13] W. Z. Khan, Y. Xiang, M. Y. Aalsalem, and Q. Arshad, "Mobile phone sensing systems: a survey," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 1, pp. 402–427, 2013.

[14] H. Ma, D. Zhao, and P. Yuan, "Opportunities in mobile crowd sensing," *IEEE Communications Magazine*, vol. 52, no. 8, pp. 29–35, 2014.

[15] N. D. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, and A. T. Campbell, "A survey of mobile phone sensing," *IEEE Communications Magazine*, vol. 48, no. 9, pp. 140–150, 2010.

[16] X. Sheng, J. Tang, and W. Zhang, "Energy-efficient collaborative sensing with mobile phones," in *Proceedings of the IEEE Conference on Computer Communications, INFOCOM 2012*, pp. 1916–1924, USA, March 2012.

[17] K. K. Rachuri, C. Efstratiou, I. Leontiadis, C. Mascolo, and P. J. Rentfrow, "Smartphone sensing offloading for efficiently supporting social sensing applications," *Pervasive and Mobile Computing*, vol. 10, pp. 3–21, 2014.

[18] M.-R. Ra, B. Liu, T. F. La Porta, and R. Govindan, "Medusa: A programming framework for crowd-sensing applications," in *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services (MobiSys '12)*, pp. 337–350, ACM, June 2012.

[19] L. A. Castro, J. Favela, E. Quintana, and M. Perez, "Behavioral data gathering for assessing functional status and health in older adults using mobile phones," *Personal and Ubiquitous Computing*, vol. 19, no. 2, pp. 379–391, 2015.

[20] D. Zhao, H. Ma, S. Tang, and X.-Y. Li, "COUPON: a cooperative framework for building sensing maps in mobile opportunistic networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 2, pp. 392–402, 2015.

[21] E. C.-H. Ngai, M. B. Srivastava, and J. Liu, "Context-aware sensor data dissemination for mobile users in remote areas," in *Proceedings of the IEEE Conference on Computer Communications, INFOCOM 2012*, pp. 2711–2715, USA, March 2012.

[22] U. Park and J. Heidemann, "Data muling with mobile phones for sensornets," in *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems (SenSys '11)*, pp. 162–175, November 2011.

[23] Y. Wang, J. Lin, M. Annavaram et al., "A framework of energy efficient mobile sensing for automatic user state recognition," in *Proceedings of the 7th ACM International Conference on Mobile Systems, Applications, and Services (MobiSys '09)*, pp. 179–192, ACM, Kraków, Poland, June 2009.

[24] H. Lu, J. Yang, Z. Liu, N. D. Lane, T. Choudhury, and A. T. Campbell, "The Jigsaw continuous sensing engine for mobile phone applications," in *Proceedings of the 8th ACM International Conference on Embedded Networked Sensor Systems (SenSys '10)*, pp. 71–84, ACM, November 2010.

[25] T. Das, P. Mohan, V. N. Padmanabhan, R. Ramjee, and A. Sharma, "PRISM: platform for remote sensing using smartphones," in *Proceedings of the 8th Annual International Conference on Mobile Systems, Applications and Services (MobiSys '10)*, pp. 63–76, June 2010.

[26] M. Shin, C. Cornelius, D. Peebles, A. Kapadia, D. Kotz, and N. Triandopoulos, "AnonySense: a system for anonymous opportunistic sensing," *Pervasive and Mobile Computing*, vol. 7, no. 1, pp. 16–30, 2011.

[27] D. Sánchez, M. Batet, D. Isern, and A. Valls, "Ontology-based semantic similarity: A new feature-based approach," *Expert Systems with Applications*, vol. 39, no. 9, pp. 7718–7728, 2012.

[28] J. J. Garcia-Luna-Aceves and R. Menchaca-Mendez, "PRIME: An interest-driven approach to integrated unicast and multicast routing in MANETs," *IEEE/ACM Transactions on Networking*, vol. 19, no. 6, pp. 1573–1586, 2011.

[29] J. J. Garcia-Luna-Aceves, J. E. Martinez-Castillo, and R. Menchaca-Mendez, "Routing to Multi-Instantiated Destinations: Principles, Practice and Applications," *IEEE Transactions on Mobile Computing*, 2017.

[30] R. Rada, H. Mili, E. Bicknell, and M. Blettner, "Development and application of a metric on semantic nets," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 19, no. 1, pp. 17–30, 1989.

[31] C. E. Perkins and E. M. Royer, "Ad-hoc on-demand distance vector routing," in *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications (WMCSA '99)*, pp. 90–100, New Orleans, La, USA, February 1999.

[32] NS-3 Consortium., "Network Simulator 3 (NS-3)," Tech. Rep., 2015.

[33] A. Alrawais, A. Alhothaily, C. Hu, and X. Cheng, "Fog Computing for the Internet of Things: Security and Privacy Issues," *IEEE Internet Computing*, vol. 21, no. 2, pp. 34–42, 2017.

[34] M. Mukherjee, R. Matam, L. Shu et al., "Security and Privacy in Fog Computing: Challenges," *IEEE Access*, vol. 5, pp. 19293–19304, 2017.