Cloud of Things for Sensing as a Service: Sensing Resource Discovery and Virtualization

Sherif Abdelwahab*, Bechir Hamdaoui*, Mohsen Guizani†, and Taieb Znati‡
* Oregon State University, abdelwas,hamdaoui@eecs.orst.edu
† Qatar University, mguizani@ieee.org | ‡ University of Pittsburgh, znati@cs.pitt.edu

***Abstract*—We propose Cloud of Things for Sensing as a Service: a global architecture that scales up cloud computing by exploiting the global sensing resources of the highly dynamic and growing Internet of Things (IoT) to enable remote sensing. The proposed architecture scales out by augmenting the role of edge computing platforms as *cloud agents* that discover and virtualize sensing resources of IoT devices. Cloud of Things enables performing in-network distributed processing of sensing data offered by the globally available IoT devices and provides a global platform for meaningful and responsive sensing data analysis and decision making. We design cloud agents algorithmic solutions bearing in mind the onerous to track dynamics of the IoT devices by centralized solutions. First, we propose a distributed sensing resource discovery algorithm based on a gossip policy that selects IoT devices with predefined sensing capabilities as fast as possible. We also propose RADV: a distributed virtualization algorithm that efficiently deploys virtual sensor networks on top of a subset of the selected IoT devices. We show, through analysis and simulations, the potential of the proposed algorithmic solutions to realize virtual sensor networks with minimal physical resources, reduced communication overhead, and low complexity.**

I. Introduction

Remote sensing applications will evolve through on-demand sensing services provided by the global network of sensor equipped devices in our homes, factories, cities, and bodies known as the Internet of Things (IoT). Today in smartphones 'only', there are seven sensors on average per device including: magnetometer, barometer, light, heart, humidity, and temperature sensors that one can use as participatory sensors to carry out applications like short-term weather forecasting [1], [2]. The density of smartphones' sensors in London today exceeds 14,000 sensor per square kilometer[1]. By 2020, the global number of sensor-equipped and location-aware devices (e.g. wearable, smart home, and fleet management devices) will reach tens of Billions, potentially creating dense, dynamic, location-aware, and onerous to manage networks of devices that can realize the vision of providing a versatile remote sensing services, known as 'Sensing as a Service' [3], [4], [5].

We conjecture that employing IoT devices' sensing resources in a *cloud computing like platform* to support remote sensing applications may be an effective approach to realize the Sensing as a Service vision [3]. The idea is to dynamically augment and scale up existing cloud resources (compute, storage, and network) by exploiting sensing capabilities of devices through cloud agents near the network edge to form a *global* system named the *Cloud of Things* (see Figure 1). The Cloud of Things is a geographically distributed infrastructure with cloud agents elements that continuously discover and pool sensing resources of IoT devices to be used by cloud users on-demand. This infrastructure provides elastic sensing resources that scale and shrink according to remote sensing applications demands, providing an optimized and controllable sensing resource utilization and pricing based on measurable usage.

Cloud of Things shifts the current conventional usage of cloud platforms in remote sensing from a 'collect sensor data now and analyze it later' scenario to a usage scenario that *directly provides meaningful information from in-network processing of sensing data by IoT devices*. Without such conjectured infrastructure, remote sensing users can still gain access to sensing resources through conventional cloud back-end systems (see [6], [4]), with less opportunities to scale out sensing applications over the globally available sensing resources and with intolerable performance to applications that require responsive exploitation and fusion of sensing data and agile in-network decisions (e.g. localization [7] and estimation [8]).

Cloud of Things infrastructure - Cloud platforms near the network edge already exist in different forms such as smartphones, personal computers, gateways, and servers to offer computation offloading to nearby devices in real-time (e.g. cloudlets [9], [10] and edge computing platforms [11], [12]). *We envision a new role of edge platforms as cloud agents that incorporate IoT devices as sensing resources (Figure 1) to scale up the conventional cloud with global and location specific sensing resources. We propose cloud agents algorithmic solutions that provide*: (1) fast discovery of devices' dynamic sensing resources in specific geographical area, and (2) optimized devices virtualization to serve as virtual sensor networks by exploiting their discovered sensing resources.

Cloud agents implement remote sensing applications as virtual sensor networks to be deployed on virtualizable IoT devices in a geographical area that perform distributed in-network processing of sensing data. These virtual sensor networks may employ devices' sensing resources discovered by multiple cloud agents. Conventional cloud platforms provide a unified interface to cloud users to seamlessly use such global sensing resources from anywhere and at anytime while hiding complexities and supporting interaction between cloud agents. In Cloud of Things, IoT devices become surrogates of federated sensor networks (i.e administrated by a single organization) that can potentially reduce the total cost of ownership for remote sensing applications.

In this paper we propose a Cloud of Things architecture for Sensing as a Services and efficient algorithms for sensing resource discovery and optimized devices virtualization. We discuss the technical challenges and our envisioned usage scenario of the proposed architecture in Section II. In Section III, we first propose a sensing resource discovery algorithm that uses a gossip policy for propagating a sensing task requirements to devices (or their virtual instance at the edge

[1]The population density in London exceeds 4,000 inhabitants per square kilometer and the UK smartphone penetration reaches 55%.

cloud) and selects feasible devices to execute the task while responding to dynamic changes of devices as fast as possible. Then, we propose RADV, an efficient virtualization algorithm, that deploys a virtual sensor network corresponding to the sensing task on top of a subset of the selected devices with minimal physical resources. Finally, we numerically evaluate our proposed algorithms in Section IV and conclude this paper in Section V.

II. Architecture Usability and Challenges

The proposed Cloud of Things architecture allows cloud users to run remote sensing tasks, with certain specifications, virtually on any sensor-equipped IoT devices (see Figure 1). For example, a cloud user can profile pollution changes in cities from real-time temperature and CO2 concentration measurements collected by sensors in vehicles with defined precision and accuracy. The architecture consists of three main elements: IoT devices, first tier clouds, and cloud agents. **IoT devices** are sensor-equipped devices that can serve both specific and general purpose remote sensing applications, including for example smart factory, smart grid, and smartphone devices. An IoT device can communicate directly with proximate devices. **First tier clouds** are conventional cloud platforms that provide unified interfaces to users to access the system and hide complexities underlying realization of sensing services. Throughout, we refer to a first tier cloud as simply 'cloud'. Finally, **Cloud agents** are trusted and resource-rich elements near the network edge that are well-connected to the Internet and to conventional cloud platforms. Cloud agents can be as powerful as supercomputers, or as flexible as smartphones according to the types of devices they serve and the computing resources these devices demand. Throughout, we refer to a cloud agent as simply 'agent'.

This architecture offers new sensing features and service level guarantees with several benefits. Deploying agents (cloud agents) close to devices improves responsiveness to sensing task requests and enables access to a globally available sensing resources. From the devices' viewpoint, cloud resources can be split into local resources (agents' resources) and global resources (clouds' resources) that can improve resiliency by migrating sensing tasks as the states of the devices, which carry out the sensing task, change. A cloud (first tier cloud) also acts as liaison to support coordination between distributed agents, while these agents can rapidly capture dynamics of the devices (e.g. utilization, connectivity, and availability). This approach simplifies analytics and big data with possible direct device access for agile in-network data processing and decision making. The proposed architecture finally allows the design of network-aware and performance-optimized cloud procedures.

A. Usage Scenario and System Model

A cloud (first tier cloud) handles *sensing tasks* initiated by a user with a unified interface (**step 1** in Figure 1). A *sensing task* defines physical parameters (e.g pollution changes) that the user wishes to estimate in a defined geographical area during a predefined time with certain sensing capabilities of the IoT devices carrying out the task. The sensing task objective can be in two forms: information retrieval of raw sensor measurements from devices (e.g. surveillance videos), or execution of distributed algorithms on a virtual sensor network deployed on multiple interconnected devices (e.g. online thermal analysis or target localization). We represent a *sensing task* by the triple, $\langle g, c, \delta \rangle$, where g denotes the number of virtual sensors requested to perform the sensing task and the two parameters c and δ define the center and the radius of a geographical area of interest to the user's remote sensing application.

The cloud translates a sensing task to a corresponding *sensing task request* that it sends to its agents. A *sensing task request* defines the set, V, of g virtual sensors to be deployed on g connected devices, which are all located within distance δ from the area center c. For each virtual sensor $j \in V$, the cloud defines a *minimum sensing capability*, $R(j)$, that can, for e.g., represent the minimum storage capacity, the minimum CPU computing power, and/or the minimum amount of time that devices (to carry out the sensing task) shall fulfill. The cloud may also choose a suitable virtual topology that interconnects the virtual sensors so that they can execute distributed algorithms for in-network processing of devices' sensors measurements. The criteria used to choose a certain topology is beyond the scope of our work and we focus on three possible virtual topologies: complete, cyclic, and star. For a given topology, let E denote the set of virtual links connecting the virtual sensors and $\Upsilon = (V, E)$ be the graph data structure that represents the *virtual sensor network* of the virtual sensors (connected according to the given virtual topology). After translating a sensing task to its corresponding *sensing task request*, the cloud sends this request (i.e. the graph data structure Υ) to its agents (**step 2** in Figure 1).

Agents manage a large number of interconnected *IoT devices*. A *device* i, at time t, maintains its geographical location denoted by $loc(i)$ and its current sensing capability denoted by $C(i)$. $C(i)$ defines the currently allowed sensing time, available processing capacity, and/or available memory capacity that the i-th device can allocate, at time t, to fulfill the minimum sensing capability demanded by a virtual sensor j (i.e. $R(j)$) to be deployed on i. *The sensing capability of a device can correspond to local device's resources (i.e. CPU, memory, storage, and sensors) or to resources at the edge cloud (agent) that the device may opportunistically use through computation offloading mechanisms.* We assume that two devices can directly communicate with each other if they are within a transmission radius, r. We model the network of all n devices, connected to a single agent, as the Euclidean geometric random graph, $\mathcal{G} = (S, L)$, where S denotes the set of n devices, and L denotes the set of all links connecting the devices.

Agents handle sensing task requests under agreed Service Level Agreements (SLAs) with users through the cloud. An SLA generally consists of: i) a maximum time within which the sensing task must be completed, ii) a feasible selection of IoT devices to carry out the sensing task under certain tolerances of the results accuracy, and iii) a maximum task rejection rate defined as the ratio of the number of failures to handle sensing task requests to the total number of requests. The cloud translates an SLA to parameters that agents can use in their algorithmic solutions to discover sensing resources and virtualize devices efficiently. Defining all possible parameters that reflect any SLA is beyond the scope of this work and we will consider only two parameters. One parameter, which we defined earlier, is the *minimum sensing capability* $R(j)$ of the j-th virtual sensor. The other parameter is the maximum allowed path length, $h_{\max}$, between any pair of virtual sensors. The value of $h_{\max}$ limits the number of devices/hops a message, exchanged between virtual sensors, can go through. A virtual link between

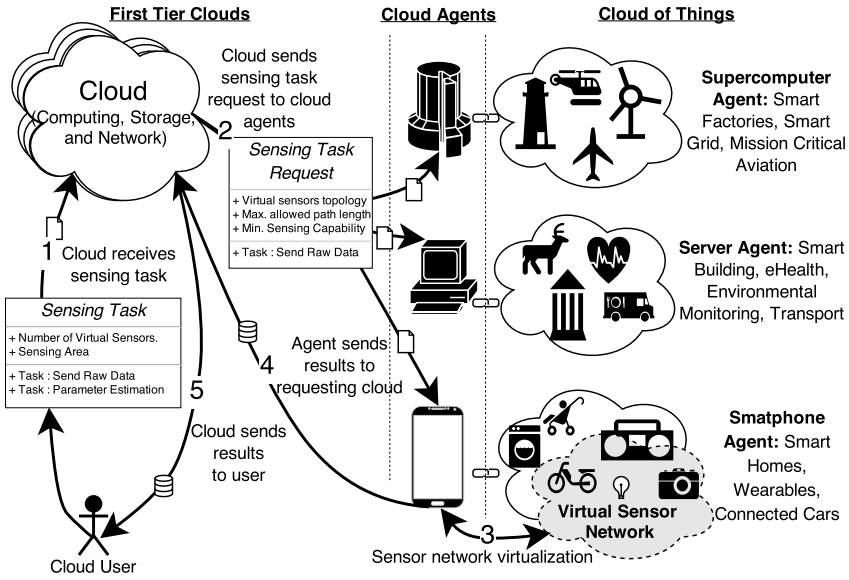

Figure 1: Sensor network virtualization in Sensing as a Service by different cloud agents near the edge. First tier clouds are conventional cloud computing platforms, and cloud agents are edge computing platform with evolved rule for Sensing as a Service. Arrows and numbers illustrate messages flow and sequence of the proposed usage scenario.

two virtual sensors may map to devices that do not necessarily deploy a virtual sensor and the virtual sensor network just use these devices for message forwarding. We use $h_{\max}$ to impose an upper limit on these intermediate devices for two reasons. First, restricting the number of intermediate devices shall bound the sensing task performance in an SLA. Second, agent's sensor discovery and virtualization algorithms shall consider minimal number of hops to deploy virtual links and generally minimal possible physical resources in sensor network virtualization to maximize the benefits of Sensing as a Service (**step 3** in Figure 1). We detail the implication of $h_{\max}$ on virtualization design objectives later in this section when we introduce our notion of 'optimal' virtualization and formulate the total devices' benefit in (3).

Finally in our proposed usage scenario, the successfully deployed virtual sensor network carries out the requested sensing task and reports the results to the agents, which make these results available to the cloud, and hence to the user. Figure 1 summarizes and illustrates the messages sequence and flow between the different architectural elements.

B. Technical Challenges and Solutions Objectives

The proposed Cloud of Things architecture and usage scenario envision designing algorithmic solutions with specific objectives, given the following set of challenges:

Sensing Resource Discovery: In the sensor network virtualization (**step 3** in Figure 1), an agent searches for devices with sensing capabilities that meet the sensing task requirements specified by the virtual sensor network data structure Υ. For a given Υ, the agent discovers devices' sensing capabilities and searches for a subset of devices, $S' \subset S$, such that a device $i \in S'$, if it is geographically located within δ distance from the center c, and the discovered sensing capability $C(i)$ satisfies the minimum sensing capability $R(j)$ demanded by at least one virtual sensor $j \in V$. For each device i we define its virtual domain, $\mathcal{D}(i)$, as

$$\mathcal{D}(i) = \begin{cases} \{j \in V : C(i) \geq R(j)\} & \text{if } \|loc(i) - c\| \leq \delta \\ \emptyset & \text{otherwise,} \end{cases} \quad (1)$$

hence

$$S' = \{i \in S : |\mathcal{D}(i)| > 0\}. \quad (2)$$

The design objective of a sensing resource discovery algorithm is to construct the virtual domains, $\mathcal{D}(i)$ for all $i \in S$, as fast as possible and with minimal communication overhead between the agent and the devices as well as between the devices.

The challenges related to sensing resource discovery arise from the large number of devices and their onerous to maintain dynamics. The large number of devices connected to an agent requires a scalable solution to discover devices' sensing capabilities and to decide if a device current state (e.g. connectivity to other devices) allows it to deploy a particular virtual sensor. Moreover, the dynamics and rapid changes in the whole network $\mathcal{G}$ including: devices availability, mobility, connection state, and resource utilization complicate maintaining devices' states in a centralized solution. We address these challenges by proposing a distributed sensing resources discovery algorithm that propagates the graph data structure Υ to devices in $\mathcal{G}$ using a gossip policy as detailed in Section III-A.

Virtualization: After performing the sensing resource discovery, an agent deploys the virtual sensor network, Υ, by means of devices virtualization. The virtualization task consists of finding: i) a set $A \subset S'$ of exactly g connected devices according to the virtual topology chosen by the cloud, and ii) a set $\mathcal{M}_A \subset \{(i,j) \in A \times V : j \in \mathcal{D}(i)\}$ of (device,virtual sensor) mapping pairs such that one virtual sensor maps to exactly one device and a device maps to one and only one virtual sensor in g. Also, the length $h(i,i')$ of any simple path connecting two distinct devices $i, i' \in A$ that maps a virtual link $(j,j') \in E$ must be less than or equal to $h_{\max}$. We refer to a $\{A, \mathcal{M}_A\}$ pair that satisfies the previous conditions as a *feasible virtualization* of the requested virtual sensor network Υ. Note that for any possible set A, there can exist multiple mappings, $\mathcal{M}_A$, and each can form a feasible virtulization. *The design objective of a virtualization algorithm is to find the 'optimal' feasible virtualization, $\{A, \mathcal{M}_A\}^*$, that uses the minimal possible physical resources (devices and physical links).*

We now define and introduce what an 'optimal' feasible virtualization means. We consider that the number of virtual sensors and the number of virtual links of a given $\Upsilon = (V, E)$ determine the cloud cost of providing the sensing service given by Cost$(\Upsilon) = \alpha|V| + \beta|E|$. The scalar α denotes an incentive paid to each device that maps a virtual sensor, and the scalar β denotes an incentive divided and paid to each device on a physical path that maps to a virtual link. An incentive could be monetary or could be in any other form (e.g., credits, services, etc.). On the other hand, the total devices' benefit from mapping the virtual sensors and the virtual links of Υ can be expressed as

$$\text{Benefit} = \sum_{(i,j)\in\mathcal{M}_A} \alpha\frac{C(i) - R(j)}{C(i)} + \sum_{(i,i')\in P} \beta\frac{h_{\max} - h(i,i')}{h_{\max}}, \tag{3}$$

where $h(i, i')$ is again the path length (in number of hops) of the path connecting the devices pair (i, i') mapping the virtual link between j and j' and $P = \{(i, i') \in A \times A : (i, j), (i', j') \in \mathcal{M}_A, (j, j') \in E\}$ denotes the set of all such pairs.

The total devices' benefit in (3) implies that the lesser the used physical resources, the greater the benefit to the devices. The first term of (3) captures the benefit loss of the i-th device from allocating resources to map a virtual sensor j. As the demanded minimum sensing capability $R(j)$ tends to be negligible to the sensing capability $C(i)$, i gets higher benefit as it invests less fraction of its resources (e.g. energy, CPU, or memory) to map j for the same incentive α. Similarly, the second term captures the benefit loss of devices i and i', which map the virtual sensors j and j' respectively. Such benefit loss results from mapping the virtual link between j and j' with more intermediate devices, as the same incentive β for the virtual link (j, j') divides on a greater number of devices (i.e. number of hops $h(i, i')$) compared to $h_{\max}$. The virtualization algorithm that we propose in Section III-B consists of finding an 'optimal' feasible virtualization that maximizes the total benefit given in (3). We refer to the optimal solution as $\{A, \mathcal{M}_A\}^*$. Clearly, finding $\{A, \mathcal{M}_A\}^*$ is a hard problem due to the factorial size of the solution space in n and to the same scale and dynamics challenges discussed earlier in the sensing resources discovery challenge.

III. Proposed Solutions for Sensing Resource Discovery and Virtualization

A. Sensing Resource Discovery

Although devices are directly accessible by cloud agents, contacting the devices at fine-grain time slots to discover their current sensing capabilities creates a significant communication and computation inefficiency for a large n. Such centralized approach requires exchanging $O(n)$ messages, in each time slot, while constructing virtual domains given by (1) requires $O(n)$ time. Moreover, activating devices periodically to update their current sensing capabilities to their cloud agent is power inefficient, especially if the devices are battery operated.

We propose to perform sensing resource discovery through a gossip based algorithm that requires a time complexity of $O(r^{-1}\log n)$ and an average $\Theta(1)$ messages per device. In this algorithm, an agent propagates information about a received sensing task request, Υ, using the following 'gossip policy'. The agent sends Υ to a randomly chosen device starting

while True **do**
  wait Δt
  $s \longleftarrow$ random neighbor
  if Υ is $\emptyset$ **then**
    solicit Υ from s
  else
    send Υ to s
  end if
  receive Υ' from s
  if $\Upsilon' = \Upsilon$ **then**
    stop sending Υ
  else
    Υ' is newer than Υ
    $\Upsilon \longleftarrow \Upsilon'$
    evaluate $\mathcal{D}(i)$
  end if
end while

i) active thread at device i

while True **do**
  receive Υ' or
  solicit request from s
  if Υ is not $\emptyset$ **then**
    send Υ to s
  end if
  if Υ' is new **then**
    $\Upsilon \longleftarrow \Upsilon'$
    evaluate $\mathcal{D}(i)$
  end if
end while

ii) passive thread at i

Figure 2: proposed sensing resources discovery gossip based threads at device i.

$t = 0$. Then, any device that receives Υ continue sending Υ to a random device of its direct neighbors until one neighbor acknowledges that it has already double received the same version of Υ in a previous step; by then the device stops sending Υ. The agent does not need to send Υ to each device as the utilized gossip policy allows devices to disseminate Υ autonomously and the network of devices is guaranteed to be connected with high probability, if each device is connected to k neighbors and $k \geq 0.5139 \log n$ [13]. Since $\mathcal{G}$ is a connected network, this simple gossip policy guarantees that Υ reaches all the devices in $O(r^{-1}\log n)$ time (see [14] for time complexity analysis of general gossip protocols in Euclidean geometric random graphs). Hence a device i can construct $\mathcal{D}(i)$ according to (1) once it receives Υ and the agent can discover sensing resources of devices that are capable of fulling the requirements of Υ as fast as possible with minimal communication overhead.

The agent and all its connected devices implement the active and passive threads shown in Figure 2. At the k-th time slot, let the device i be active and contacts a random neighbor device i' (i.e., $(i, i') \in L$) with probability $T_{i,i'} > 0$. The probability $T_{i,i}$ denotes the probability that i does not contact any other device. Let the $n \times n$ matrix $T = [T_{i,i'}]$ be a doubly stochastic transition matrix of non-negative entries [15]. A natural choice of $T_{i,i'}$ is

$$T_{i,i'} = \begin{cases} \dfrac{1}{d_i + 1}, & \text{if } i = i' \text{ or } (i, i') \in L, \\ 0, & \text{otherwise,} \end{cases} \tag{4}$$

where $d_i = |\{i' \in S : (i, i') \in L\}|$ is the degree of i.

When i contacts i', they exchange information as follows (Figure 2). The device i pushes Υ to i' only if i' does not have Υ, or pulls Υ from i' only if i does not have Υ. If i contacts i' and both devices have received Υ before, i stops contacting any other device.

The actual running time of the proposed algorithm depends on the choice of the transition matrix T and the communication range of the considered device-to-device communication technology. The running time is related to the mixing time of any

random walk on $\mathcal{G}$ [15], which suggests that there is an optimal value of $T_{i,i'}$ to minimize the mixing time and it is related to the second eigenvalue of the transition matrix. Moreover, in case of small r, the proposed algorithm is generally slow. Practically, this algorithm is suitable for device-to-device communication technologies that support communication ranges of few hundreds of meters, as in WiFi direct and LTE D2D and when $\mathcal{G}$ is sufficiently dense.

B. Virtualization

We propose RADV: a randomized and asynchronous distributed virtualization algorithm for Sensing as a Service which consists of three phases: (I) pruning of virtual domains $\mathcal{D}(i)$ for all $i \in S$ to limit the devices executing RADV to those that can feasibly deploy a given Υ, (II) construction of benefit matrices locally in devices in a distributed manner to allow maximizing the total devices' benefit (3), and (III) solving an assignment problems locally at some devices with $|\mathcal{D}(i)| > 0$ to find an optimal feasible virtualization. This approach results in multiple solutions each evaluated by a different device, and a cloud agent selects the solution with the maximum benefit. *These multiple solutions also allow the agent to offer better resiliency by enabling rapid migration of a virtual sensor in case of failure or sensing capability change of the device which maps the virtual sensor.*

We now present each of the four phases in details.

***Phase I**—Virtual Domain Pruning:* During this phase, we ensure that all virtualized devices maintain the topology described by E or a virtual sensor network Υ by allowing a device to receive the virtual domains of other devices and delete a virtual sensor j from its domain if there exists a virtual link (j, j') such that j' is not included in any other received domains. Let $D_s \subset \{\mathcal{D}(i) : i \in S\}$ denotes the set of domains that a device s has received at time k. Initially $D_s = \{\mathcal{D}(s)\}$ and $h(i, s) = 0$ for all $i \in S$[2]. Using the same transition matrix, T, defined in (4), s contacts only one of its neighbors s' at time k. Then, for all $\mathcal{D}(i) \in D_s : i \neq s'$, s pushes $\mathcal{D}(i)$ to s' only if s' did not receive $\mathcal{D}(i)$ before and $h(i, s) < h_{\max}$. Also, for all $\mathcal{D}(i) \in D_{s'} : i \neq s$, s pulls $\mathcal{D}(i)$ from s' only if s did not receive $\mathcal{D}(i)$ before and $h(i, s') < h_{\max}$. If no domain is exchanged between s and s' at time k, s stops contacting any of its neighbors. However, s may restart contacting its neighbors again if it updated D_s after time $k+1$. This part of the protocol is a multi-piece information dissemination gossip policy that requires simple modification to the threads in Figure 2.

When s constructs its D_s, it starts by pruning $\mathcal{D}(s)$. The pruning is performed by deleting a virtual sensor $j \in \mathcal{D}(s)$ (i.e., $\mathcal{D}(s) \leftarrow \mathcal{D}(s) \setminus \{j\}$) if none of the virtual sensors that are connected to j, $\{j' \in V : (j, j') \in E\}$, is not included in any received $\mathcal{D}(i)$, i.e. $j \notin \mathcal{D}(i) : \mathcal{D}(i) \in D_s$. This pruning rule ensures that the devices maintain the required topology E and the constructed benefit matrices shall result in a feasible virtualization.

The time required to spread information in the pruning phase is $O(r^{-1} n \log n)$. A device i examines g received virtual domains, each having at most g entries, hence requires $O(g^2)$ time to prune $\mathcal{D}(i)$. Since every device exchanges a maximum of n domains each of size $O(g)$, the average number of messages communicated per device is $O(n)$. However, as we restrict that messages to be communicated up to $h_{\max}$ hops, the average number of messages per device is typically small.

***Phase II**—Construction of Benefit Matrices:* As mentioned earlier, finding a feasible virtualization, $\{A, \mathcal{M}_A\}^*$, that maximizes the total benefit (3) is a hard problem due to the large size of the solution space. Therefore, this phase proposes an efficient way of solving this virtualization problem.

During this phase, each device s locally constructs its own set, $A^{(s)}$, of g candidate devices that s chooses to map the virtual sensors in V. Each device s also maintains g row vectors, $B_i^{(s)} \in \mathbf{R}^{1 \times g}$ and $i \in A^{(s)}$, that we define as the benefit vector of the i-th device seen by s, where the j-th element, $B_{i,j}^{(s)}$, denotes the benefit of mapping the device $i \in A^{(s)}$ to the virtual senor $j \in V$ as seen by s, and is given by

$$B_{i,j}^{(s)} = \begin{cases} \alpha \dfrac{C(i) - R(j)}{C(i)} + \beta \dfrac{h_{\max} - h(j,s)}{h_{\max}} & \text{if } j \in \mathcal{D}(i), \\ 0 & \text{otherwise.} \end{cases}$$

Our objective is to construct, for each $s \in S$, the benefit matrix $B^{(s)} = [B_{i \in A^{(s)}}^{(s)}]$ as fast as possible, and find a feasible virtualization, $\{A, \mathcal{M}_A\}$, that maximizes the total benefit,

$$\sum_{(i,j) \in \mathcal{M}_A} B_{i,j}^{(s)},$$

among all $s \in S$ without central knowledge of the complete $\mathcal{G}$ structure. Obviously, the path length between a device s and any other device i that s includes in its benefit matrix must not exceed $h_{\max}$. Finally, a device s shall include only the benefit vectors of the g devices with the largest possible benefit.

Each device s initially sets $A^{(s)} = A^{(s)} \cup \{s\}$ if $\mathcal{D}(s) \notin \emptyset$, sets $h(i, s) = 0$ for all $i \in S$, and sets

$$B_{s,j}^{(s)} = \begin{cases} \alpha \dfrac{C(s) - R(j)}{C(s)} + \beta, & j \in \mathcal{D}(s), \\ 0, & \text{otherwise.} \end{cases}$$

Also, s maintains a scalar, $b_s^{\min}$, defined as the minimum total benefit it has received from any other device

$$b_s^{\min} = \min_i \sum_{j \in V} B_{i,j}^{(s)},$$

where the minimum corresponding device is

$$i_s^{\min} = \operatorname*{argmin}_i \sum_{j \in V} B_{i,j}^{(s)}.$$

Initially, $b_s^{\min} = 0$ and remains so until $|A^{(s)}| = g$.

Using the same transition matrix, T, s contacts its neighbor s' only once at each time k. Then, for all $i \in A^{(s)} : i \neq s'$, s pushes the benefit vector $B_i^{(s)}$ to s' only if $h(i, s) < h_{\max}$ and

$$\sum_{j \in V} \left(B_{i,j}^{(s)} - \frac{\beta}{h_{\max}} \right) > b_{s'}^{\min}.$$

Also, for all $i \in A^{(s')} : i \neq s$, s pulls the benefit vector $B_i^{(s')}$ from s' only if $h(i, s') < h_{\max}$ and

$$\sum_{j \in V} \left(B_{i,j}^{(s')} - \frac{\beta}{h_{\max}} \right) > b_s^{\min}.$$

[2]Knowledge about other devices existence is not needed, and h is dynamically evaluated.

If no benefit vector is exchanged between s and s' at time k, s stops contacting its neighbors at time $k+1$. However, s may restart contacting its neighbors again if $B^{(s)}$ is updated after time $k+1$.

When s receives $B_i^{(s')}$, s updates $B_{i,j}^{(s)}$ as

$$B_{i,j}^{(s)} = \begin{cases} B_{i,j}^{(s)} - \dfrac{\beta}{h_{\max}} & \text{if } \ j \in \mathcal{D}(i), \\ 0 & \text{otherwise.} \end{cases}$$

If $i \notin A^{(s)}$, then we have two scenarios. In the first scenario, s has not received g benefit vectors, so $b_s^{\min} = 0$ and $|A^{(s)}| < g$, then s updates its set of candidate devices as $A^{(s)} = A^{(s)} \cup \{i\}$. In the other scenario in which $|A^{(s)}| = g$, s replaces the device, $i_s^{\min}$, that corresponds to the minimum total benefit with i so that $A^{(s)} = A^{(s)} \setminus \{i_s^{\min}\} \cup \{i\}$. On the other hand, if $i \in A^{(s)}$, then s updates $B_{i,j}^{(s)}$ if $\sum_{j \in V} B_{i,j}^{(s')} > \sum_{j \in V} B_{i,j}^{(s)}$. Finally, s updates $b_s^{\min}$, $i_s^{\min}$, and $h(i,s)$ as $h(i,s) = h(i,s') + 1$.

Finding a feasible virtualization that maximizes the benefit matrix $B^{(s)} = [\ B_{i \in A^{(s)}}^{(s)}\ ]$ instead of the total benefit in (3) makes the problem easier because every device has a different value for the benefit $B_{i,j}$ that depends only on the length of the physical path between i and s instead of the path lengths of all possible combinations of sensor pairs (i, i') that can map a virtual link. *Intuitively, this relaxation still leads to an optimal virtualization for star virtual topology or near optimal virtualization for other topologies. For a solution found by s, the worst case path length between any two devices (i, i') other than s in this solution is at most double the maximum of the path lengths between (i, s) or (i', s).* We evaluate the effectiveness of this relaxation in Section IV and show that our virtualization algorithm performs well for large and dense enough $\mathcal{G}$.

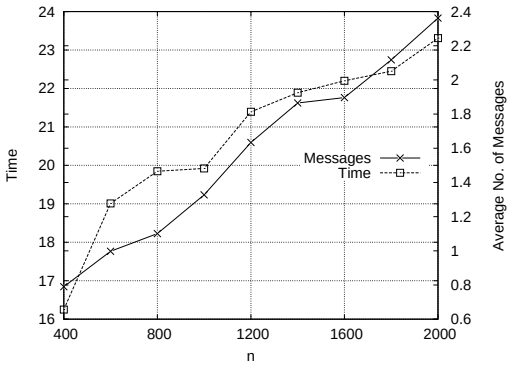

Figure 3: The average messages per device and the maximum time at which all devices in $\mathcal{G}$ prune their virtual domains and construct benefit matrices (in the time unit of Δt).

The time required to construct the benefit matrices is $O(r^{-1} n \log n)$. Figure 3 shows the total time and the average number of messages per device required during both the domain pruning and the benefit construction phases. The time in Figure 3 is linearithmic in n when the agent sends Υ initially to exactly one device, which can be significantly improved with a better implementation such that the agent continue sending Υ to random devices. As analyzed, the average number of messages per device is shown to scale linearly with n. The benefit construction phase dominates the number of message exchanged which is typically a very small fraction of n.

***Phase III**—Solving Local Assignment Problem:* After the reception of the g benefit vectors, s proceeds to this phase of the algorithm only if it stops communicating and $|A^{(s)}| = g$. Each device $s \in S$ with $|A^{(s)}| = g$ solves locally the following assignment problem:

$$\begin{aligned} \text{maximize} \quad & \sum_{i \in A^{(s)}} \sum_{j \in \mathcal{D}(i)} B_{i,j}^{(s)} m_{ij} \\ \text{subject to} \quad & \sum_{j \in \mathcal{D}(i)} m_{ij} = 1,\ i \in A^{(s)}, \\ & \sum_{\{i: j \in \mathcal{D}(i)\}} m_{ij} = 1,\ j \in V, \\ & m_{ij} \in \{0, 1\}, \end{aligned} \tag{5}$$

where m_{ij} is a binary variable indicating whether the i-th device maps to the virtual sensor j. The problem formulated in (5) is equivalent to the maximum weight perfect matching problem in a bipartite graph, allowing us to use the well known Hungarian method to solve it in $O(g^3)$ worst case time [16].

We can improve this time complexity to linear time, if we tolerate an error $\epsilon > 0$ of the resulting total benefit and relax the restriction of finding a perfect matching for large g; i.e it is not necessary to map all virtual sensors to devices. In such scenario, we can use the linear time $(1-\epsilon)$-approximation algorithm that is proposed recently in [17] to solve (5). Details of these algorithms are omitted due to space limitation (see [16], [17] for details).

In RADV, each device solves locally the optimization problem given in (5) using the Hungarian method and sends its obtained solution to the cloud agent. The agent then selects the solution that leads to the maximum total benefit, and keeps all other solutions for use in case virtual sensors migration is needed. The overall complexity of RADV is bounded by either solving the assignment problem at each sensor, or the time required to prune the virtual node domain, i.e. $O(\max\{r^{-1} n \log n, g^3\})$.

IV. Numerical Results

In this section, we numerically evaluate the performance of the proposed RADV algorithm implemented in our own simulator in Python, NetworkX, and Simpy. In our simulations, $\mathcal{G}$ and Υ are generated considering the parameters summarized in Table I. The topology of any Υ can either be complete, cyclic, or star. To evaluate RADV's solutions closeness to optimal, we consider a cloud agent that receives and services only one virtual sensing task request, Υ, at a time.

Table I: Simulation Parameters

Parameter	r	$C(i)$	$R(j)$	δ	$h_{\max}$
Value	0.1	$\sim U(50, 100)$	$\sim U(25, 50)$	0.2	20

We also consider the maximum total benefit that any virtualization algorithm can achieve to assess RADV performance. Such an upper bound is attained by mapping the virtual sensors of Υ to g devices with the maximum sensing capability and by assuming that virtual links always map to paths of exactly one hop. This simple bound has a theoretically wide optimization gap, as mapping virtual links to single-hop paths can be attained only for a large enough $\mathcal{G}$; where $\mathcal{G}$ has a clique of size g such that all devices in this clique can feasibly map virtual sensors of Υ (i.e. sensing capabilities of devices are always greater than demands, and sensing tasks are not restricted in a certain location).

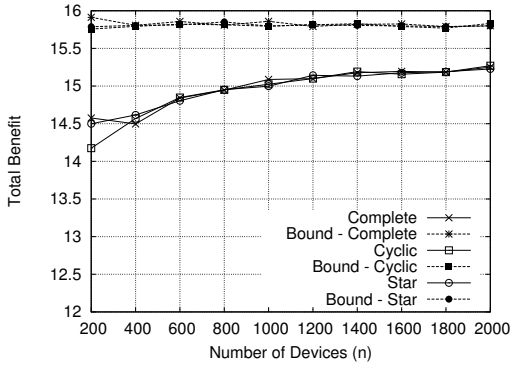

Figure 4: The total benefit achieved by RADV when compared to the upper bound for different topologies and number of devices. The performance gap of the star topology quantifies the looseness of the upper bound. The shown total benefit is very close to the upper bound as the network becomes denser.

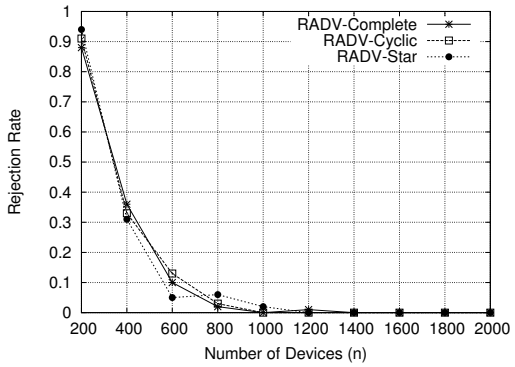

Figure 5: Virtual sensor network rejection rate encountered by RADV for different number of devices. For dense and unloaded network ($n > 1000$), RADV discovers sensing resources and finds a solution to the virtualization problem almost surely.

This suggested bound, despite the fact that it creates a wide optimization gap, enlightens sufficient insights about the closeness of RADV's solutions to optimal. Figure 4 numerically evaluates the total benefit in (3) that is achieved by RADV for different virtual topologies and compares it to the maximum total benefit bound we just introduced. As the network gets denser, RADV achieves a total benefit that is very close to the upper bound.

Figure 5 shows the rejection rate encountered with different Υ topologies and n values. As we only consider one Υ at a time, the results shown in this figure reflect the impact of Υ topology, the number of devices n, and the simulation parameters given in Table I on the rejection rate. Observe that the denser the $\mathcal{G}$, the lower the rejection rate, implying that the cloud is capable of granting higher number of requests as we discussed earlier.

V. Conclusion

We have described our initial research in Cloud of Things. We have shown the potential of Cloud of Things to scale cloud computing vertically by exploiting sensing resources of IoT devices to provide Sensing as a Service. We have proposed a global architecture that scales Cloud of Things horizontally by employing edge computing platforms in a new role as cloud agents that discover and virtualize sensing resources of IoT devices. We have described cloud agents technical challenges and design objectives for sensing resources discovery and virtualization that can dispatch offering virtual sensor networks deployed on IoT devices to cloud users with in-network processing capabilities. We have proposed our sensing resource discovery solution based on a gossip policy to discover sensing resrouces as fast as possible and RADV: our virtualization solution. We have shown through analysis and simulations the potential of RADV to achieve reduced communication overhead, low complexity, and closeness to optimal such that RADV employs minimal physical resources in devices virtualization with maximal benefit.

VI. Acknowledgment

This work was made possible by NPRP grant # NPRP 5-319-2-121 from the Qatar National Research Fund (a member of Qatar Foundation). The statements made herein are solely the responsibility of the authors.

References

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

[2] C. F. Mass and L. E. Madaus, "Surface pressure observations from smartphones: A potential revolution for high-resolution weather prediction?" *Bulletin of the American Met. Society*, vol. 95, no. 9, 2014.

[3] S. Abdelwahab, B. Hamdaoui, M. Guizani, and A. Rayes, "Enabling smart cloud services through remote sensing: An internet of everything enabler," *Internet of Things Journal, IEEE*, vol. 1, no. 3, June 2014.

[4] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Sensing as a service model for smart cities supported by internet of things," *Trans. on Emerging Telecomm. Technologies*, vol. 25, no. 1, 2014.

[5] S. Abdelwahab, B. Hamdaoui, and M. Guizani, "Bird-VNE: Backtrack-avoidance virtual network embedding in polynomial time," in *Proceedings of IEEE Global Telecommunications Conference*, 2014.

[6] X. Sheng, J. Tang, X. Xiao, and G. Xue, "Sensing as a service: Challenges, solutions and future directions," *Sensors Journal, IEEE*, vol. 13, no. 10, pp. 3733–3741, 2013.

[7] A. Rai, K. K. Chintalapudi, V. N. Padmanabhan, and R. Sen, "Zee: zero-effort crowdsourcing for indoor localization," in *Proceedings of the 18th annual international conference on Mobile computing and networking*. ACM, 2012, pp. 293–304.

[8] S. Abdelwahab, B. Hamdaoui, and M. Guizani, "Cloud-assisted remote sensor network virtualization for distributed consensus estimation," *arXiv preprint arXiv:1501.03547*, 2015.

[9] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *Pervasive Computing, IEEE*, vol. 8, no. 4, pp. 14–23, 2009.

[10] E. Koukoumidis, D. Lymberopoulos, K. Strauss, J. Liu, and D. Burger, "Pocket cloudlets," *ACM SIGPLAN Notices*, vol. 47, no. 4, 2012.

[11] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*. ACM, 2012, pp. 13–16.

[12] C. Shi, K. Habak, P. Pandurangan, M. Ammar, M. Naik, and E. Zegura, "Cosmos: computation offloading as a service for mobile devices," in *Proceedings of the 15th ACM international symposium on Mobile ad hoc networking and computing*. ACM, 2014, pp. 287–296.

[13] P. Balister, B. Bollobás, A. Sarkar, and M. Walters, "Highly connected random geometric graphs," *Discrete Applied Mathematics*, vol. 157, no. 2, pp. 309–320, 2009.

[14] D. Shah, *Gossip algorithms*. Now Publishers Inc, 2009.

[15] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *Information Theory, IEEE Trans. on*, vol. 52, no. 6, 2006.

[16] J. Munkres, "Algorithms for the assignment and transportation problems," *Journal of the Society for Industrial & Applied Mathematics*, vol. 5, no. 1, pp. 32–38, 1957.

[17] R. Duan and S. Pettie, "Linear-time approximation for maximum weight matching," *Journal of the ACM (JACM)*, vol. 61, no. 1, p. 1, 2014.