Cloud of Things for Sensing-as-a-Service: Architecture, Algorithms, and Use Case

Sherif Abdelwahab*, Bechir Hamdaoui*, Mohsen Guizani†, and Taieb Znati‡

⋆ Oregon State University, abdelwas,hamdaoui@eecs.orst.edu
† University of Idaho, 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 Internet of Things (IoT) to enable remote sensing. Cloud of Things enables in-network distributed processing of sensors data offered by the globally available IoT devices and provides a global platform for meaningful and responsive data analysis and decision making. We propose a distributed sensing resource discovery and virtualization algorithms that efficiently deploy virtual sensor networks on top of a subset of the selected IoT devices. We show, through analysis and simulations, the potential of the proposed solutions to realize virtual sensor networks with minimal physical resources, reduced communication overhead, and low complexity. We also design an uncoordinated, distributed algorithm that relies on the selected sensors to estimate a set of parameters without requiring synchronization among the sensors. Our simulations show that the proposed estimation algorithm, when compared to conventional ADMM (Alternating Direction Method of Multipliers), reduces communication overhead significantly without compromising the estimation error. In addition, the convergence time, though increases slightly, is still linear as in the case of conventional ADMM.**

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 smatphones '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 Fig. 1). The Cloud of Things is a geographically distributed infrastructure with cloud agent 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 up and down 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 remote sensing use of cloud platforms 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 a 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]).

A. 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 (Fig. 1) to scale up the conventional cloud with global and location specific sensing resources. We propose*

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

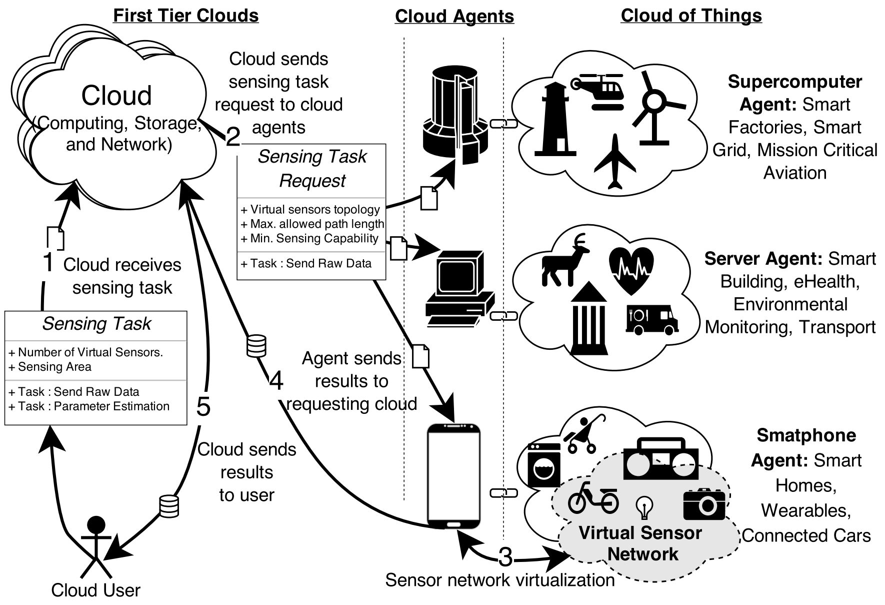
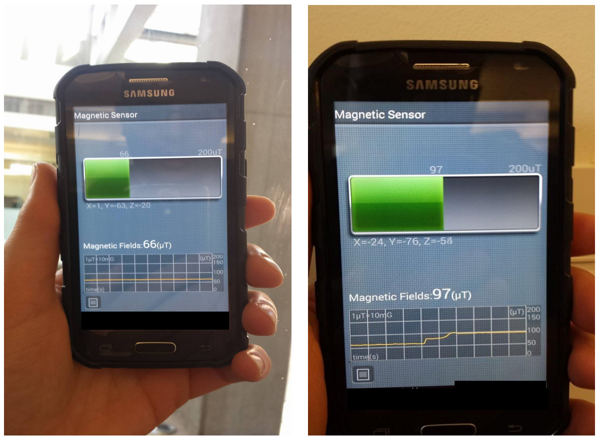

First Tier Clouds
Cloud Agents
Cloud of Things
Cloud (Computing, Storage, and Network)
Cloud sends sensing task request to cloud agents
2
Sensing Task Request
+ Virtual sensors topology
+ Max. allowed path length
+ Min. Sensing Capability
+ Task : Send Raw Data
1 Cloud receives sensing task
Sensing Task
+ Number of Virtual Sensors.
+ Sensing Area
+ Task : Send Raw Data
+ Task : Parameter Estimation
5
4
Agent sends results to requesting cloud
Cloud sends results to user
Cloud User
3
Sensor network virtualization
Virtual Sensor Network
Supercomputer Agent: Smart Factories, Smart Grid, Mission Critical Aviation
Server Agent: Smart Building, eHealth, Environmental Monitoring, Transport
Smatphone Agent: Smart Homes, Wearables, Connected Cars

Fig. 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.

algorithmic solutions that provide: (1) fast discovery of devices' dynamic sensing resources in specific geographical areas, (2) optimized device virtualization to serve as virtual sensor networks by exploiting the discovered sensing resources, and (3) efficient in-network processing of sensing data from unreliable but dense sensors in IoT devices.

Cloud agents implement remote sensing applications as virtual sensor networks to be deployed on virtualizable IoT devices in a geographical area. A virtual sensor network performs distributed in-network processing of sensing data such as: aggregation, feature extraction, belief propagation, and consensus estimation to serve applications such as: distributed computer vision, data analytics, or on-demand context awareness. These virtual sensor networks may employ devices' sensing resources that are discovered by the various 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.

However, the IoT devices usually incorporate cheap and unreliable sensors to serve specific task that is not intended for remote sensing applications. For example, augmented reality applications in smartphones make use of the measurements from magnetic field sensors.



(a) Magnetic field sensor reading close to a window (low energy environment).

(b) Same sensor reading close to a power source (high energy environment).

Fig. 2: Example of energy profiling from cheap magnetic field sensor in smartphones.

The same magnetic field sensors can be used to profile energy levels in different environment (see Fig. 2 for an example). The main problem with remote sensing applications based on IoT devices sensors is that individual measurements from the sensors of a single IoT device (e.g. magnetic field) are insufficient for a reliable sensing task. In Fig. 2, it is hard to distinguish the real context of a magnetic sensor reading changes;

whether it is a result of user proximity to the device, changes in the device's orientation, or presence in a high energy environment. Similar unreliability problems appear in traffic congestion estimation in navigation applications (e.g. Google maps), weather prediction from smartphones barometers, or indoor localization.

A virtual sensor network of a group of independent IoT devices can solve this problem through distributed consensus estimation. Instead of analyzing measurements from an individual sensor, the virtual network use independent sensor measurements from several devices and executes an efficient *in-network* distributed consensus estimation algorithm to be able to efficiently achieve its goal (e.g. estimating energy level in an environment surrounding a user). In this paper, we focus our analysis and evaluation on distributed consensus estimation as the sensing task under study.

B. Contribution and Organization

In this paper, we propose a system to perform in-network analytics, such as distributed parameter estimation, based on commodity IoT devices that act as surrogates of wireless sensor networks (i.e. virtual sensor networks). We design a virtualization algorithm that suits the use case of describing analytics as on-demand virtual sensor networks and the challenges of the conjectured architecture in Fig. 1. We also propose a distributed consensus parameter estimation algorithm to be executed by the optimized virtual sensor network. The distributed consensus algorithm provides a reliable, high quality parameter estimates from the low-quality and unreliable sensors in commodity IoT devices.

We discuss the technical challenges and our envisioned use case 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 cloud) and selects feasible devices to execute the task while responding to the 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. In Section IV, we propose RADE; an efficient estimation algorithm that relies on the virtual sensor network, formed by our proposed virtualization algorithm, to estimate a set of unknown parameters in a distributed way and without requiring synchronization among the IoT devices. We discuss several related work in Section VI. Finally, we numerically evaluate our proposed algorithms in Section V and conclude this paper in Section VII.

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 Fig. 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. **First tier clouds** are conventional cloud platforms that provide unified interfaces to users to access the system and hide complexities underlying the 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 simply as '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. Use case and System Model

Fig. 1 summarizes message sequence and flow between the different architectural elements. These are detailed as next.

1) First tier clouds: A cloud (first tier cloud) handles *sensing tasks* initiated by a user with a unified interface (**step 1** in Fig. 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: information retrieval of raw sensed data, or execution of distributed algorithms on a virtual sensor network deployed on multiple interconnected devices. We represent a *sensing task* by the triple, $\langle g, c, \delta \rangle$, where g denote 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.

2) Sensing task requests: The cloud translates a sensing task to a corresponding *sensing task request* that it sends to its agents. A *sensing task request* defines the virtual sensors set, V, 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*, $R(j)$. The minimum sensing capability represents the minimum storage capacity, the minimum CPU computing power, and/or the minimum amount of time that devices (to carry out the sensing task) must 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' sensed measurements.

For example, for aggregation and belief propagation algorithms, the cloud organizes the virtual sensor network as a spanning tree. A star topology can also be adopted for distributed algorithms that are implemented using the map-reduce or graph-processing paradigms. Although consensus algorithms, which are our main focus, can run with any arbitrary topology, we show that a complete topology results in faster convergence. For our evaluation, we focus on three common 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 Fig. 1).

3) The IoT devices capabilities: 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 also 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 denote the set of n devices, and L denote the set of all links connecting the devices. We assume that each sensor $i \in S$ is capable of estimating a vector of unknown parameters, $\theta \in \mathbf{R}^N$, through noisy sensors measurements, $x_i \in \mathbf{R}^M$. That is,

$$x_i = H_i \theta + u_i, \ i = 1, \ldots, n$$

where $H_i \in \mathbf{R}^{M \times N}$ is sensor i's sensing model (typically known to i only) relating x_i to θ, and u_i is an additive Gaussian noise with zero mean and variance σ_i^2. We assume that u_i and u_j are independent from one another for all $i, j \in S$. Because different sensors may have different sensing models and/or different measurement methods, it is very likely that different sensors have different estimates of θ. Also, we do not assume/require that the sensors are synchronized; that is, the consensus algorithms we develop in this paper to estimate θ are asynchronous.

4) Service Level Agreement (SLA) implications on agents configurations: Agents handle sensing task requests under agreed 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 consider only four parameters.

The first parameter is the absolute error of the estimated parameter θ, denoted by ϵ_{abs}. The second parameter is the relative error, ϵ_{rel}, of the parameter θ estimated by the different virtual sensors such that all sensors estimate θ within ϵ_{rel}. The third parameter, defined earlier, is the *minimum sensing capability* $R(j)$ of the j-th virtual sensor. The fourth parameter is the maximum allowed path length, $\bar{h}$, between any pair of virtual sensors. $\bar{h}$ limits the number of devices/hops a message, exchanged between virtual sensors, can go through.

A virtual link between 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 $\bar{h}$ 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 by an SLA. Second, the sensor discovery and virtualization algorithms shall use the least number of hops and the least possible physical resources when mapping the virtual network, so as to maximize the Sensing-as-a-Service benefits (**step 3** in Fig. 1). The implication of $\bar{h}$ on the virtualization design will be discussed later in this section.

5) Sensing task execution (consensus): Consensus estimation resembles the most commonly used sensing task relying on a collection of measurements from unreliable sensors. In consensus estimation, the sensing task is to estimate a set of parameters, θ, based on the measurements sensed by the IoT devices so that the estimated parameters are at most ϵ_{abs} away from their actual value, and so that all the sensors consent to the same estimate value of θ with a tolerance of ϵ_{rel} according to the SLA.

Without loss of generality, consider indexing the selected g sensors in the virtual sensor network as $1 \ldots g$ and let $x = \left[x_1^\mathsf{T}, \ldots, x_g^\mathsf{T}\right]^\mathsf{T}$, $H = \left[H_1^\mathsf{T}, \ldots, H_g^\mathsf{T}\right]^\mathsf{T}$, and $u = \left[u_1^\mathsf{T}, \ldots, u_g^\mathsf{T}\right]^\mathsf{T}$. The combined measurements can then be written as $x = H\theta + u$. One simple approach of estimating θ is to have the cloud agent first collect from each virtual sensor i its measurement vector, x_i, and its sensing model, H_i, and then solve the following Least Squares (LS) problem

$$\text{minimize} \quad \tfrac{1}{2}\|x - H\hat{\theta}\|^2 \tag{1}$$

where $\hat{\theta}$ is here the optimization variable. The unbiased maximum-likelihood (ML) estimate of θ is simply $\hat{\theta}_{\text{LS}} = \left(H^\mathsf{T} H\right)^{-1} H^\mathsf{T} x$.

B. Technical Challenges and Solutions Objectives

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

1) Sensing resource discovery: In the sensor network virtualization (**step 3** in Fig. 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 $j \in V$. We define the virtual domain, $\mathcal{D}(i)$, of a device i as

$$\mathcal{D}(i) = \{j \in V : C(i) \geq R(j), \|loc(i) - c\| \leq \delta\} \tag{2}$$

hence

$$S' = \{i \in S : |\mathcal{D}(i)| > 0\}. \tag{3}$$

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 and between the devices themselves.

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's 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 device availability, mobility, connectivity, and resource utilization, make it too difficult to maintain devices' states in a centralized manner. To address these challenges, we propose a distributed algorithm that propagates the graph data structure Υ to devices in $\mathcal{G}$ using a gossip policy as detailed in Section III-A.

2) 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 $\bar{h}$. 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 then to find the 'optimal' feasible virtualization, $\{A, \mathcal{M}_A\}^*$, that uses the least possible physical network resources.*

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, which is given by $\text{Cost}(\Upsilon) = \alpha|V| + \beta|E|$. The scalar α denote an incentive paid to each device that maps a virtual sensor, and the scalar β denote 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., credit, service, etc.). On the other hand, the total devices' benefit from mapping the requested virtual network, Υ, can be expressed as

$$\text{Benefit} = \sum_{(i,j)\in\mathcal{M}_A} \alpha \frac{C(i) - R(j)}{C(i)} + \sum_{(i,i')\in P} \beta \frac{\bar{h} - h(i,i')}{\bar{h}}, \tag{4}$$

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

The total devices' benefit in (4) implies that the lesser the used physical resources, the greater the benefit to the devices. The first term of (4) captures the benefit loss of the i-th device from allocating resources to map a virtual sensor j. As the minimum demanded sensing capability $R(j)$ becomes negligible (w.r.t. the sensing capability $C(i)$), i gets higher benefit as it invests lesser 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') is divided on a greater number of devices (i.e. number of hops $h(i,i')$) compared to $\bar{h}$. Theoretically, $\bar{h}$ can take a value up to the diameter of $\mathcal{G}$. However, this shall not work in practice as the diameter of $\mathcal{G}$ is assumed to be much greater than a user desired diameter Υ. The virtualization algorithm that we propose in Section III-B consists of finding an 'optimal' feasible virtualization that maximizes the total benefit given in (4). We refer to the optimal solution as $\{A, \mathcal{M}_A\}^*$. Clearly, finding $\{A, \mathcal{M}_A\}^*$ is hard due to the factorial size of the solution space in n and due to the challenges, discussed earlier, associated with the sensing resources discovery task.

3) Distributed consensus estimation: The virtual sensor network determined during the virtualization phase executes the distributed sensing algorithms. The simple solution to the LS problem, proposed in (1), requires that each virtual sensor exchanges its measurement vector and its sensing model with the cloud agent, which can create significant communication overhead. Therefore, we instead propose a decentralized approach that relies on the virtual sensor network to provide an estimation of the parameter vector θ. We rely on the recent results presented in [13] to develop our distributed estimation algorithm, which reduces communication overhead significantly when compared to the conventional Alternating Direction Method of Multipliers (ADMM) approach [14] in addition to not requiring synchronization among sensors. The proposed algorithm is presented in Section IV.

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-grained time slots to discover their current sensing capabilities creates significant communication and computation inefficiency for large n. Such a centralized approach requires exchanging $O(n)$ messages, in each time slot, while constructing the virtual domains, given by (2), requires $O(n)$ time. Moreover, activating devices periodically to update their current sensing capabilities to their cloud agents 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 at $t = 0$. Then, any device that receives Υ continues 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 $k \geq 0.5139 \log n$ [15].

```
while True do
    wait $\Delta t$
    $s \longleftarrow$ random neighbor
    if $\Upsilon$ is $\emptyset$ then
        solicit $\Upsilon$ from $s$
    else
        send $\Upsilon$ to $s$
    end if
    receive $\Upsilon'$ from $s$
    if $\Upsilon' = \Upsilon$ then
        stop sending $\Upsilon$
    else
        $\Upsilon'$ is newer than $\Upsilon$
        $\Upsilon \longleftarrow \Upsilon'$
        evaluate $\mathcal{D}(i)$
    end if
end while
```

i) active thread at device i

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

ii) passive thread at i

Fig. 3: proposed sensing resources discovery gossip based threads at device i.

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 [16] for time complexity analysis of general gossip protocols in Euclidean geometric random graphs). Hence a device i can construct $\mathcal{D}(i)$ according to (2) once it receives Υ and the agent can discover sensing resources of devices that are capable of satisfying 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 Fig. 3. At the k-th time slot, let the device i be active and contact a random neighbor device i' (i.e., $(i, i') \in L$) with probability $T_{i,i'} > 0$. $T_{i,i}$ denote 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 [17]. A natural choice of $T_{i,i'}$ is

$$T_{i,i'} = \begin{cases} \dfrac{1}{d_i + 1}, & \text{if } i = i' \text{ or } (i, i') \in L, \\ 0, & \text{otherwise,} \end{cases} \tag{5}$$

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

When i contacts i', they exchange information as follows (see Fig. 3). 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. If $\mathcal{G}$ is connected, the proposed protocol guarantees that Υ is delivered to all IoT devices.

The actual running time of the proposed algorithm depends on the choice of the transition matrix T and the communication range of the used device-to-device communication technology. The running time is related to the mixing time of any random walk on $\mathcal{G}$ [17], 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 present our proposed Randomized and Asynchronous Distributed Virtualization (RADV) algorithm. RADV consists of three phases: (I) pruning of virtual domains $\mathcal{D}(i)$ for all $i \in S$, (II) construction of benefit matrices in a distributed manner, and (III) solving assignment problems at virtual sensors. RADV results in multiple solutions each evaluated by a different sensor (device), and the cloud agent selects the solution with the maximum benefit.

Phase I*—Virtual Domain Pruning:* During this phase, we ensure that all virtualized sensors maintain the topology E by allowing a senor to receive the virtual domains of other sensors 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 $D_s \subset \{\mathcal{D}(i) : i \in S\}$ denote the virtual domains set that sensor s has 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 Eq. (5), 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 $h(i, s) < \bar{h}$. 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 $h(i, s') < \bar{h}$. If no information 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 $k + 1$.

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 virtualized sensors maintain the required topology E and the constructed benefit matrices shall result in a feasible virtualization.

Phase II*—Construction of Benefit Matrices:* As mentioned earlier, finding a feasible virtualization, $\{A, \mathcal{M}_A\}^*$, that maximizes the total benefit given in Eq. (4) 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. Specifically, we propose a method that solves this problem in a distributed manner and without requiring any synchronization among sensors, as described next.

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

During this phase, each sensor s locally constructs its own set, $A^{(s)}$, of g sensors that s chooses as virtualized sensors to assign to virtual sensors in V. Each sensor 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 sensor i seen by s, where the j-th element, $B_{i,j}^{(s)}$, denotes the benefit of assigning participatory sensor $i \in A^{(s)}$ to the virtual senor $j \in V$ as seen by s, and is given by

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

Our objective is then 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 knowing the $\mathcal{G}$ structure. Moreover, the path length between a sensor s and any other sensor i that s includes in its benefit matrix must not exceed $\bar{h}$. Finally, a sensor s shall include only the benefit vectors of the g sensors with the largest possible benefit.

Each sensor 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 sensor and written as

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

s also maintains the corresponding sensor,

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

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

Using the same transition matrix, T, defined in Eq. (5), 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 $h(i, s) < \bar{h}$ and

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

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

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

If no information 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

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

If $i \notin A^{(s)}$, then we have two scenarios. In the first scenario, s still has not received g benefit vectors, so $b_s^{\min} = 0$ and $|A^{(s)}| < g$, then s updates its set of candidate sensors as $A^{(s)} = A^{(s)} \cup \{i\}$. In the other scenario in which $|A^{(s)}| = g$, s replaces the sensor corresponding to the minimum total benefit, $i_s^{\min}$, 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 $B^{(s)} = [\ B_{i \in A^{(s)}}^{(s)}\ ]$ instead of the benefit given in Eq. (4) makes the problem easier because every sensor 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 virtualize a virtual link. Intuitively, this relaxation still leads to an optimal or near optimal virtualization, because if $\mathcal{G}$ is very large and connected, the number of sensors that are directly connected by a single physical link (clique) grows logarithmically in n and hence this number is larger than g almost surely as $g \ll n$. In such a case, it is sufficient to ensure that the length of the paths between i and s and between i' and s are the shortest possible ones to ensure that the length of the path between i and i' is also the shortest, as in this case, s, i, and i' reside in the same clique with high probability. We evaluate the effectiveness of this relaxation in Section V and show that our virtualization algorithm performs well even when the condition $g \ll n$ does not hold.

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

$$\begin{aligned} \text{maximize} \quad & \sum_{i \in A^{(s)}} \sum_{j \in \mathcal{D}(i)} B_{i,j}^{(s)} m_{ij} \\ \text{subject to} \quad & \sum_{j \in \mathcal{D}(i)} m_{ij} = 1,\ i \in A^{(s)}, \\ & \sum_{\{i : j \in \mathcal{D}(i)\}} m_{ij} = 1,\ j \in V, \\ & m_{ij} \in \{0, 1\}, \end{aligned} \tag{6}$$

where m_{ij} are binary optimization variables indicating whether the participatory sensor i is assigned to the virtual sensor j. The problem formulated in (6) is equivalent to the perfect maximum weight matching problem in a bipartite graph, and hence, we propose to use the classical Hungarian method to solve it (the worst case time complexity is $O(g^3)$ [18], [19]).

We can also tolerate an error $\epsilon > 0$ of the resulting total benefit and relax the restriction of finding a perfect matching for large g. This relaxation is reasonable when there are enough sensors involved in solving these local optimization problems, as in this case we can pick the best solution and discard those without a perfect matching. In such a scenario, we can also use a linear time $(1-\epsilon)$-approximation algorithm to solve (6) [20]. In this paper, we use the Hungarian method to solve our formulated optimization problems. Details of the algorithm are omitted due to space limitation; readers are referred to [18], [19], [20] for detailed information.

Each sensor solves locally the optimization problem given in (6) and sends its obtained solution to the cloud agent. This is done asynchronously. The cloud agent then selects the solution that leads to the maximum total benefit, and keeps all other solutions for later use in the event that the network dynamics invalidate the selected solution before the virtual sensing task completes.

Complexity and message overhead. We assume that the topology of $\mathcal{G}$, devices mobility, and sensing capability are not changed during the execution of the virtualization phase. The time required to spread Υ across the network is $O(r^{-1} \log n)$ [16]. It takes $O(g)$ worst case time to evaluate $\mathcal{D}(i)$ locally at sensor i. Also, the time required to spread information in the pruning and benefit construction phases is $O(r^{-1} n \log n)$. The pruning of the virtual domain $\mathcal{D}(i)$ requires node i to examine g received virtual domains, each having at most g entries. The worst case local running time of pruning is then $O(g^2)$. Finally, the local running time of the Hungarian method is $O(g^3)$. Hence, the overall complexity is $O(\max\{r^{-1} n \log n, g^3\})$.

The average number of messages communicated per sensor during the sensor search phase is $\Theta(1)$ and each message is $O(g)$ in size. During pruning of virtual domains, since every sensor exchanges a maximum of n domains each of size that is also $O(g)$, the average number of messages communicated per sensor is $O(n)$. However, because we restrict that messages to be communicated up to $\bar{h}$ hops for only a group of sensors that support the requirements of Υ, the average number of messages per sensor is typically small. Fig. 4 shows the total time and the average number of messages per sensor required during both the domain pruning and the benefit construction phases. The total time growth is linearithmic in n when Υ is sent to exactly one sensor and when $\mathcal{G}$ is connected. This time can, in practice, be decreased significantly if Υ is initially sent to multiple sensors. Additionally, the average number of messages per sensor is shown to scale linearly with n, and is typically a very small fraction of n.

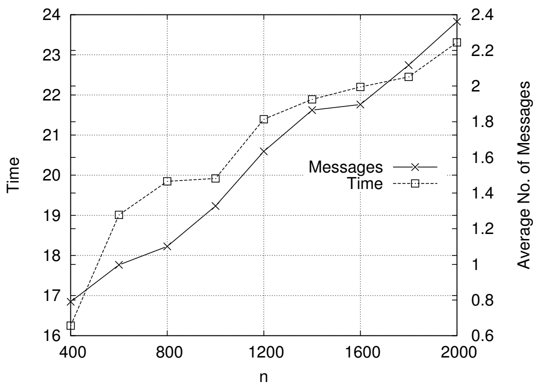


Fig. 4: Time (in number of iterations) and message overhead (in average number of communicated messages) resulting from constructing the benefit matrices, $g = 10$.

IV. Proposed Solutions for Distributed Estimation

After completing the sensor virtualization task, using RADV, the virtual sensors run an in-network parameter estimation algorithm to compute $\hat{\theta}$ distributedly. In this section, we present our proposed Randomized and Asynchronous Distributed Estimation (RADE) algorithm. We first follow the standard ADMM approach to derive primal, dual and Lagrangian variable update equations, then we describe the proposed RADE algorithm. For clarity of notation, in what follows, we refer to the set of g selected devices, determined by RADV, simply as A.

The centralized estimation approach given in (1) is first decomposed into g local estimates of θ (one $\hat{\theta}_i$ for each $i \in A$) while constraining the local estimates with the coupling constraints $\theta_i = \theta_j$ for all $(i,j) \in P$. This results in the following optimization problem:

$$\begin{aligned} &\text{minimize} && \tfrac{1}{2}\sum_{i\in A} \|x_i - H_i\theta_i\|^2 \\ &\text{subject to} && \theta_i - \theta_j = 0 \text{ for all } (i,j)\in P, \end{aligned} \tag{7}$$

where $\{\theta_i, i \in A\}$ are the optimization variables.

By introducing an auxiliary variable, z, we decouple the constraints in (7), so that $\theta_i - z = 0$ for all $i \in A$ [21]. However, this requires that z be shared among all the g virtual sensors. Instead, we introduce g auxiliary variables, z_i, and equivalently write the optimization problem as

$$\begin{aligned} &\text{minimize} && \tfrac{1}{2}\sum_{i\in A} \|x_i - H_i\theta_i\|^2 \\ &\text{subject to} && \theta_j - z_i = 0 \text{ for all } (i,j)\in P. \end{aligned} \tag{8}$$

Let $\lambda = \{\lambda_{i,j} \in \mathbf{R}^{N\times 1} : (i,j) \in P\}$ and $\rho = \{\rho_{i,j} \in \mathbf{R} : (i,j) \in P\}$ denote respectively the set of Lagrangian multipliers and the set of penalty parameters. The augmented Lagrangian is given by

$$\begin{aligned} \mathcal{L}_\rho(\theta, z, \lambda) = \sum_{i\in A} \Bigg[&\tfrac{1}{2}\|x_i - H_i\theta_i\|^2 \\ &- \sum_{j\in A:(i,j)\in P} \lambda_{i,j}^{\mathsf{T}}(\theta_i - z_j) \\ &+ \sum_{j\in A:(i,j)\in P} \tfrac{\rho_{i,j}}{2}\|\theta_i - z_j\|^2 \Bigg]. \end{aligned} \tag{9}$$

By setting the gradient w.r.t θ_i of Eq. (9) to zero and solving for θ_i, we get

$$\begin{aligned} \theta_i &= \left(H_i^{\mathsf{T}} H_i + \sum_{j\in A:(i,j)\in P} \rho_{i,j} I \right)^{-1} \\ &\quad \cdot \left(H_i^{\mathsf{T}} x_i + \sum_{j\in A:(i,j)\in P} (\lambda_{i,j} + \rho_{i,j} z_j) \right). \end{aligned}$$

Similarly, we solve for z_i by setting the gradient w.r.t to z_i to zero and rearranging the indices of the Lagrangian multipliers and the penalty parameters. It follows that

$$z_i = \frac{1}{g} \sum_{j\in A:(i,j)\in P} \left(\theta_j - \frac{1}{\rho_{j,i}} \lambda_{j,i} \right).$$

The former analysis leads to the conventional ADMM-based distributed consensus estimation algorithm given by

$$\begin{aligned} \theta_i^{(k+1)} &= \left(H_i^{\mathsf{T}} H_i + \sum_{j\in A:(i,j)\in P} \rho_{i,j} I \right)^{-1} \\ &\quad \cdot \left(H_i^{\mathsf{T}} x_i + \sum_{j\in A:(i,j)\in P} \left(\lambda_{i,j}^{(k)} + \rho_{i,j} z_j^{(k)} \right) \right), \\ z_i^{(k+1)} &= \tfrac{1}{g} \sum_{j\in A:(i,j)\in P} \left(\theta_j^{(k)} - \tfrac{1}{\rho_{j,i}} \lambda_{j,i}^{(k)} \right), \\ \lambda_{j,i}^{(k+1)} &= \lambda_{j,i}^{(k)} - \rho_{j,i} \left(\theta_j^{(k+1)} - z_i^{(k+1)} \right), \end{aligned} \tag{10}$$

where the superscript k denotes the value of the variable at the k-th iteration. This conventional ADMM algorithm, given in (10), requires synchronization and variable update among the sensors [22], [23]. Moreover, at each iteration k, each sensor i must send $z_i^{(k)}$ and $\theta_i^{(k)}$ to all other sensors it is connected to, so as to evaluate their $k+1$ primal, dual, and Lagrangian multipliers. When M is small, this algorithm incurs communication overhead that can be shown to be worse than the communication overhead incurred by centralized estimation methods. However, when M is large, the conventional ADMM algorithm incurs lesser communication overhead than what centralized estimation methods incur, but it still remains practically unattractive due to other weaknesses, detailed later in Section V.

Given the absolute and relative tolerances, ϵ_{abs} and ϵ_{rel}, specified by the SLAs, we define the primal and dual tolerances, controlling the convergence of the algorithm at iteration k, as

$$\epsilon_i^{pri}(k) = \sqrt{g}\epsilon_{\text{abs}} + \epsilon_{\text{rel}} \max(\|\theta_i^{(k)}\|, \| - z_i^{(k)}\|),$$

and

$$\epsilon_i^{dual}(k) = \sqrt{g}\epsilon_{\text{abs}} + \epsilon_{\text{rel}} \sum_{j \in A} \|\rho_{j,i}\lambda_{j,i}\|.$$

The tolerances, ϵ_i^{pri} and ϵ_i^{dual}, define the stopping criteria of sensor i; i.e., sensor i stops updating θ_i and z_i when

$$\|\theta_i^{(k+1)} - z_i^{(k+1)}\| < \epsilon_i^{pri}(k), \quad (11)$$

and

$$\|z_i^{(k+1)} - z_i^{(k)}\| < \epsilon_i^{dual}(k). \quad (12)$$

The stopping criteria of RADE are different from those of the conventional ADMM. Unlike the conventional ADMM where all sensors shall stop computations all at the same time using a common stopping criteria and common primal and dual tolerances, the stopping criteria (Eq. (11) and Eq. (12)) of RADE allow a sensor i to stop its computations asynchronously and independently from other sensors. However, these criteria are not enough to ensure asynchronous implementation, as synchronization is still required for dual and primal variable updates at iteration $k+1$ due to their dependencies on k.

To ensure full asynchronous implementation, we use the doubly stochastic transition matrix, $T \in \mathbf{R}^{g \times g}$, where $T_{i,j}$ is the probability that a sensor i contacts another sensor j at any iteration, for deciding the communications among sensors. We can have

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

where $d_i' = |\{j \in A : (i,j) \in P\}|$ is the degree of the virtual sensor, in Υ, that i virtualizes. At iteration $k+1$, sensor i may need to contact only one sensor j, unless both of i's stopping criteria, Eq. (11) and Eq. (12), are already satisfied. Whereas sensor j can be contacted by more than one sensor if j is not contacting any other sensor, even when both of j's stopping criteria are satisfied.

Upon contacting j, sensor i pushes $\theta_i^{(k)}$ to j only if i's primal stopping condition is not satisfied and pushes $z_i^{(k)}$ to j only if i's dual stopping condition is not satisfied. Also, i pulls $\theta_j^{(k)}$ from j only if j's primal stopping condition is not satisfied and pulls $z_j^{(k)}$ only if j's dual stopping condition is not satisfied. Finally, both i and j update their $k+1$ variables using the most recent values they received from other sensors.

Mean square error and convergence. The asynchronousness and randomization design of RADE do not impact the Mean Square Error (MSE) achieved by RADE when compared to ADMM. This is explained as follows. In both ADMM and RADE, the number of necessary dual and primal variables updates that are needed until convergence remains unchanged, so that convergence to the same estimate is guaranteed in both algorithms. Fig. 5 shows the MSE achievable under both RADE and ADMM when compared to LS under each of the three studied sensor network topologies: complete, star, and cycle. These results show the optimality of RADE that we intuitively discussed. All approaches have the same accuracy. But of course each of them does so at a different performance cost, as will be discussed later.

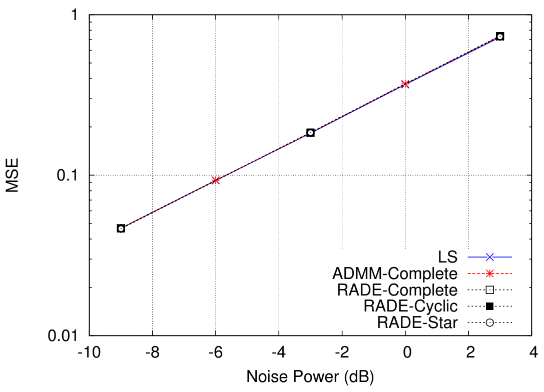


Fig. 5: MSE of RADE compared to those achieved under ADMM and LS at different noise power and for different virtual sensor network topologies, $g = 10$.

On the other hand, RADE exhibits a linear convergence rate ($O(1/k)$), similar to what the conventional ADMM does. Fig. 6 shows the number of time steps required for both RADE and ADMM to converge under different relative tolerance parameters, ϵ_{rel}. RADE convergence tends to be more restricted by the randomization nature of the algorithm for smaller values of ϵ_{rel}, which can be seen by the increasing number of steps as g increases if $\epsilon_{\text{rel}} = 10^{-2}$. ADMM generally requires a lesser number of steps to converge by relaxing the consensus constraint (through reducing ϵ_{rel}). However, as will be seen in the numerical results section later, this increase in the number of convergence steps is acceptable when considering the amount of communication overhead that the algorithm saves.

V. Numerical Results

In this section, we evaluate the performance of the proposed RADV and RADE algorithms through simulations. In our simulations, $\mathcal{G}$ and Υ, are generated using the parameters summarized in Table I. We evaluate the performance for a complete, cyclic, or star virtual sensor network topology, with a randomly chosen central location, c. We consider receiving and servicing only one virtual sensing task request at a time. The $\mathcal{G}$ topology and connectivity can change rapidly. For a single Υ, we assume that the network change rate is slow enough for the completion of the sensor search and virtualization phases. The absolute and relative tolerances, ϵ_{abs} and ϵ_{rel}, are set to 10^{-4} unless specified otherwise.

Fig. 7 shows the rejection rate encountered with different Υ topologies and n values. As we only consider

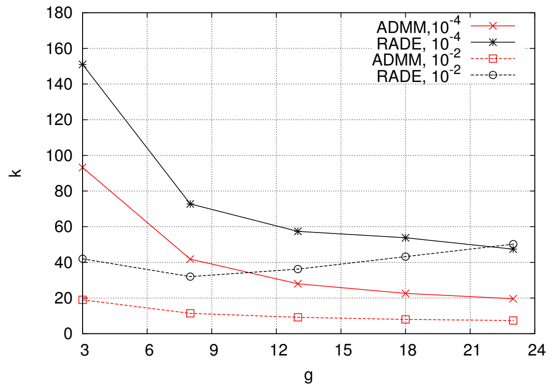


Fig. 6: Number of time steps (k) needed until convergence of RADE when compared to ADMM for complete topology under different relative tolerance values ϵ_{rel}.

TABLE I: Simulation Parameters

Parameter	g	r	$C(i)$	$R(j)$	δ	$\bar{h}$
Value	10	0.1	$\sim U(50, 100)$	$\sim U(25, 50)$	0.2	20

one single request at a time, the results shown in this figure reflect mainly the impact of the virtual sensor network topology, the number of sensors n, and the simulations parameters given in Table I on the rejection rate. The denser the network of IoT devices is, the lower the rejection rate, implying that the cloud is capable of granting higher number of requests.

One way of assessing the effectiveness of the virtualization algorithm is by measuring the difference between the total virtualization benefit given in (4) and the cost associated with the sensor virtualization introduced in Section III-B. For a given number of virtual sensors, the cost is mainly determined by the choice of the topology (star topology has the lowest cost and complete topology has the highest one). For

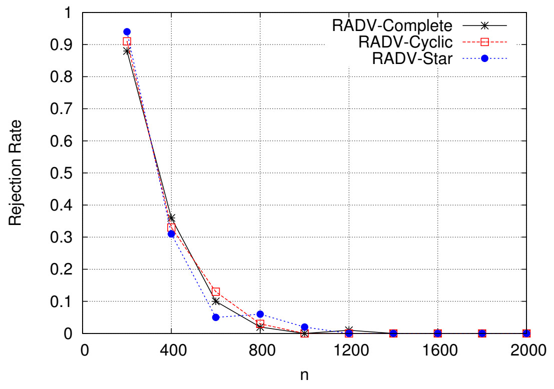


Fig. 7: Rejection rate encountered at different n.

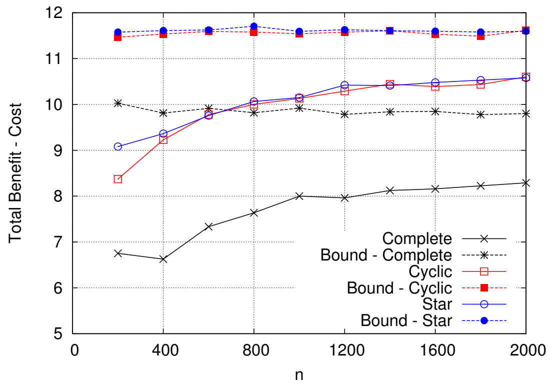


Fig. 8: Virtualization cost of RADV when compared to the upper bound under different topologies.

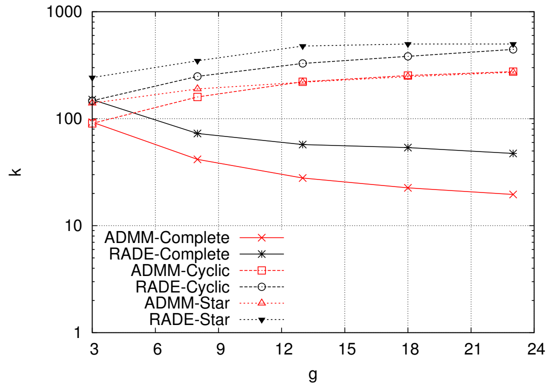


Fig. 9: Number of time steps until convergence of RADE when compared to ADMM under different topologies.

a given topology, the total benefit is maximized when each virtual sensor is assigned to the IoT devices with the maximum capacity and each virtual link is mapped to exactly one physical link. We refer to this maximized benefit as the upper bound.

In Fig. 8, we evaluate the virtualization effectiveness achieved by RADV under different virtual topologies. As the network gets denser, RADV achieves a Total Benefit − Cost that is very close to the upper bound. Since the lowest possible virtualization cost is with star or cyclic topologies, it is desired by the cloud to arrange each virtual sensing task in a star or a cyclic topology. This observation holds true for a more general topologies. On the other hand, convergence and communication overhead of the distributed estimation is also impacted by the cloud agent's choice of the virtual topology. This creates a design trade-off, as we will see in the next two paragraphs.

Fig. 9 shows the impact of the virtual topology choice on the convergence performance of RADE when compared to ADMM. If g is small (three to eight), the

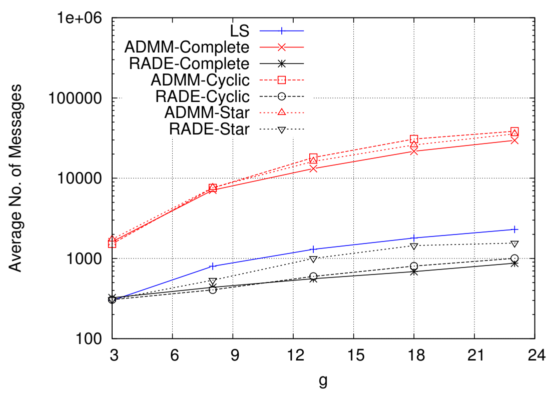


Fig. 10: Communication overhead when comparing RADE to ADMM and LS under different g values.

impact of the virtual topology on convergence of RADE and ADMM is minimal. This is because the degree of parallelism (number of virtual sensors active at the same time) is restricted by the small number of virtual sensors g. In such a scenario, it is convenient for the cloud agent to always arrange the virtual sensors in a star topology. However, as g increases, the impact of choice of the virtual topology becomes significant as the degree of parallelism is higher in a complete topology, enabling RADE to converge much faster as g gets larger. This convergence becomes slower with star and cyclic topologies. This is because in star and cyclic topologies, only few sensors are active at a time, making RADE and ADMM converge in a number of steps comparable to that of the ADMM's sequential implementation. In this later scenario, the cloud agent shall arrange the virtual sensors as a complete topology unless the SLA permits slower convergence.

Moreover, RADE converges in a higher number of steps when compared to the conventional ADMM. This is because in ADMM, all sensors are active at each time, and a sensor exchanges its updated variables with all of its neighbors, whereas in RADE, only disjoint sensor pairs are active at a time and variables are updated only between pairs of sensors. Nevertheless, we argue that this loss in speed of convergence for RADE is marginal when compared to the significant savings in communication overhead.

Fig. 10 shows the total number of $O(N)$ sized messages exchanged during estimation when comparing RADE, ADMM, and LS for $M = 100$. The number of messages exchanged by RADE is at least an order of magnitude less than the number of messages generated under ADMM. Also the communication overhead of RADE is less than the centralized LS especially as M becomes large. This savings in communication overhead is attributed to the asynchronous design of RADE in which messages among sensors are only exchanged if new values of a primal or dual variables are changed away from their specified tolerances.

VI. Related Work

A. *Network Virtualization*

Network virtualization techniques proposed in the past decade consist mainly of virtual network embedding algorithms, which instantiate virtual networks on substrate infrastructures [24], [25]. Most of these virtual network embedding algorithms are centralized (e.g. [5], [26]) due to the ease of deployment of centralized approaches in cloud platforms where the cloud provider desires to have full control on the physical network resources. Distributed network virtualization techniques are suited for Cloud of Things, given the size of the network, and are also proposed for applications in resource allocation in distributed clouds, wireless sensor network virtualization, and cloud network as a service [27], [28].

Beck et al. propose a hierarchical partitioning of any substrate network [29] and solve the network virtualization (virtual network embedding) problem on the scale of smaller partitions by delegating the problem to delegate nodes. In our context, their algorithm can progress in four steps: (1) partitioning the network of IoT devices, (2) assigning delegation nodes among the IoT devices that actually perform the network virtualization, (3) setting distributed lock trees to avoid inconsistent solutions among different delegation nodes, and (4) embedding the virtual sensor network within the scope of the delegation nodes. Several assumptions in this work prevent this method applicability in Cloud of Things. The authors require that a centralized node manage the IoT network topology to perform the partitioning. The centralized node partitions the IoT devices in groups that are highly interconnected. This requirement is very hard to achieve, if not impossible, in an Internet-scale network of IoT devices, not only because of the size of the network but also due to its highly dynamic nature that prevents tracking the states of the devices and their connectivity. Moreover, to apply the same method in cloud of things, the delegation nodes of Beck's method needs to learn a significant amount of information about the IoT devices in their neighborhood, which creates significant practical problems such as: timely information retrieval, privacy concern, and computation power.

Esposito and Ibrahim propose to model the network virtualization as a network utility maximization problem, where the utility is a general function that is measured on each hosting node (i.e. IoT device) [30]. They solve the problem distributivity using primal dual decomposition. To employ their algorithm in our Cloud of Things context, the non-convex constraints of the network virtualization problem need be relaxed. Such a relaxation is known to have a negative impact on the accuracy of the solution and may lead to false decisions [5]. Moreover, this method requires the definition of a single utility for the IoT device that shall not reflect the actual embedding cost due to network conditions, and only captures network conditions seen locally by the

IoT device. Finally, solving the problem using primal-dual decomposition requires cooperation between *all* IoT devices in the network, which prevents this method from scaling on large-sized networks without adequate network partitioning as approached by Beck et al. in [29].

In [31], [32], the authors propose a distributed virtual node mapping algorithm using consensus-auction. To apply it in our context, a utility function needs to be defined for each IoT device that is based on the devices local capacity and link bandwidth (known only to the device). The IoT devices then start bidding for virtual sensors to maximize its utility. Although the node mapping is very close to optimal, the overall result of the algorithm is not necessarily optimal since the link mapping uses only first hop link information of the substrate nodes. The authors also assume that paths are computed using $3-$shortest paths, which overlook other path diversity that can be found in a large substrate network.

The earlier methods are designed for federated data centers, and cannot be applied to the Cloud of Things context. In our approach, each device with a non-empty virtual mapping domain first solves the problem within a radius of $\bar{h}$ of the subgraph centered at the IoT device. Then, the agent selects the best solution by reducing different solutions found by the devices to a single best solution. Unlike the work of Beck et al. in [29], partitioning and delegation are an implicit process of the algorithm performed through the gossip policy used during the sensing resource discovery phase where only the IoT devices with a non-empty virtual mapping domain start to execute the next steps of the algorithm. This approach does not require updated knowledge about all the IoT devices and the topology of the network. The IoT devices perform virtualization by exchanging minimal information to construct their local benefit matrices and solving an assignment problem. Unlike [30], [31], [32], a benefit matrix constructed by a device captures all network information within a radius of $\bar{h}$ around that device, thereby improving the obtained solution closeness to optimal without overlooking important network characteristics. And unlike [30], [29], we do not restrict the link mapping to use $k-$shortest paths, thereby allowing the use of diverse paths in the network constructed during the randomized gossip policy used during the benefit matrices construction.

B. Distributed Estimation

Distributed parameter estimation approaches have been proposed in [14], [33], [34]. Estimation can for e.g. be carried out by first computing a local estimate at each virtual sensor and then perform a distributed weighted average of the local estimates [33]. This approach results in an ML estimate, but does not limit/bound the variation between mean square errors of local estimates. More recently, Paul et al. [14] propose a distributed estimation algorithm based on ADMM. Although this approach results in an optimal mean square error when compared to LS, it exhibits a significant in-network communication overhead that requires even more messages to be exchanged among sensors than that exchanged in the centralized LS. One approach also proposed in [14] to overcome this problem is to approximate the computation of primal and dual variables at each step of the algorithm by using predictions and earlier versions of these variables instead of sharing them at each iteration which marginally reduces the communication overhead. In addition to the increased communication overhead, conventional ADMM requires synchronous operation of the sensors. This is very challenging from a practical viewpoint, and does not scale well especially when applied in the Internet of Things (IoT) context. It has been shown recently that an asynchronous implementation of ADMM has $O(1/k)$ convergence [13]. Our proposed estimation algorithm is both asynchronous and distributed, and reduces communication overhead significantly when compared to the conventional ADMM approach [14].

VII. Conclusion and Discussion

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 gave a taxonomy of the potential sensing tasks, their applications, and there challenges. 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. We also proposed RADE for distributed consensus estimation as we believe it is one major sensing task in Sensing as a Service. Using simulation, we show that RADE reduces the communication overhead significantly without compromising the estimation error when compared to the traditional ADMM algorithm. We also show that the convergence time of our proposed algorithms maintain linear convergence behavior, as in the case of conventional ADMM.

VIII. 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. The authors would like to thank Thinh Nguyen and Mohamed Grissa for their early comments on this work.

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 Meteorological Society*, vol. 95, no. 9, pp. 1343–1349, 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," *Transactions on Emerging Telecommunications Technologies*, vol. 25, no. 1, pp. 81–93, 2014.

[5] S. Abdelwahab, B. Hamdaoui, and M. Guizani, "Bird-vne: Backtrack-avoidance virtual network embedding in polynomial time," in *Global Communications Conference (GLOBECOM), 2014 IEEE*. IEEE, 2014, pp. 4983–4989.

[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, M. Guizani, and T. Znati, "Cloud of things for sensing as a service: Sensing resource discovery and virtualization," in *2015 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2015, pp. 1–7.

[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] E. Wei and A. Ozdaglar, "On the o (1/k) convergence of asynchronous distributed alternating direction method of multipliers," in *Global Conference on Signal and Information Processing (GlobalSIP), 2013 IEEE*. IEEE, 2013, pp. 551–554.

[14] H. Paul, J. Fliege, and A. Dekorsy, "In-network-processing: Distributed consensus-based linear estimation," *Communications Letters, IEEE*, vol. 17, no. 1, pp. 59–62, 2013.

[15] 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.

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

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

[18] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[19] 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.

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

[21] D. P. Palomar and M. Chiang, "A tutorial on decomposition methods for network utility maximization," *Selected Areas in Communications, IEEE Journal on*, vol. 24, no. 8, pp. 1439–1451, 2006.

[22] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[23] G. Xu, H. Paul, D. Wuebben, and A. Dekorsy, "Fast distributed consensus-based estimation (fast-dice) for cooperative networks," in *Smart Antennas (WSA), 2014 18th International ITG Workshop on*. VDE, 2014, pp. 1–8.

[24] A. Fischer, J. F. Botero, M. Till Beck, H. De Meer, and X. Hesselbach, "Virtual network embedding: A survey," *Communications Surveys & Tutorials, IEEE*, vol. 15, no. 4, pp. 1888–1906, 2013.

[25] A. Belbekkouche, M. Hasan, A. Karmouch *et al.*, "Resource discovery and allocation in network virtualization," *Communications Surveys & Tutorials, IEEE*, vol. 14, no. 4, pp. 1114–1128, 2012.

[26] X. Cheng, S. Su, Z. Zhang, H. Wang, F. Yang, Y. Luo, and J. Wang, "Virtual network embedding through topology-aware node ranking," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 2, pp. 38–47, 2011.

[27] P. T. Endo, A. V. de Almeida Palhares, N. N. Pereira, G. E. Goncalves, D. Sadok, J. Kelner, B. Melander, and J.-E. Mångs, "Resource allocation for distributed cloud: concepts and research challenges," *Network, IEEE*, vol. 25, no. 4, pp. 42–46, 2011.

[28] H. Alshaer, "An overview of network virtualization and cloud network as a service," *International Journal of Network Management*, vol. 25, no. 1, pp. 1–30, 2015.

[29] M. Till Beck, A. Fischer, H. de Meer, J. F. Botero, and X. Hesselbach, "A distributed, parallel, and generic virtual network embedding framework," in *Communications (ICC), 2013 IEEE International Conference on*. IEEE, 2013, pp. 3471–3475.

[30] F. Esposito and I. Matta, "A decomposition-based architecture for distributed virtual network embedding," in *Proceedings of the 2014 ACM SIGCOMM workshop on Distributed cloud computing*. ACM, 2014, pp. 53–58.

[31] F. Esposito, D. Di Paola, and I. Matta, "On distributed virtual network embedding with guarantees," *Transactions on Networking*, 2014.

[32] F. Esposito and F. Chiti, "Distributed consensus-based auctions for wireless virtual network embedding," in *Communications (ICC), 2015 IEEE International Conference on*. IEEE, 2015, pp. 472–477.

[33] P. Tarrío, A. M. Bernardos, and J. R. Casar, "Weighted least squares techniques for improved received signal strength based localization," *Sensors*, vol. 11, no. 9, pp. 8569–8592, 2011.

[34] F. Garin and L. Schenato, "A survey on distributed estimation and control applications using linear consensus algorithms," in *Networked Control Systems*. Springer, 2010, pp. 75–107.