Vision: The Case for Symbiosis in the Internet of Things

Ahmed Saeed*, Mostafa Ammar*, Khaled A. Harras+, and Ellen Zegura*
*Georgia Institute of Technology
+Carnegie Mellon University
{amsmti3,ammar}@cc.gatech.edu, kharras@cs.cmu.edu, ewz@cc.gatech.edu

ABSTRACT

Smart devices are becoming more powerful with faster processors, larger storage, and different types of communication modalities (e.g., WiFi, Bluetooth, and cellular). In the predominant view of Internet of Things (IoT) architecture, all smart devices are expected to communicate with cloud services and/or user-held mobile devices for processing, storage, and user interaction. This architecture heavily taxes Internet bandwidth by moving large volumes of data from the edge to the cloud, and presumes the availability of low-cost, high-performance cloud services that satisfy all user needs. We envision a new approach where all devices within the same network are 1) *logically* mesh connected either directly through Bluetooth or indirectly through WiFi, and 2) cooperate in a *symbiotic* fashion to perform different tasks. We consider instantiating this vision in a system we call *SymbIoT*. We first present the design goals that need to be satisfied in SymbIoT. We then discuss a strawman system's architecture that allows devices to assume different roles based on their capabilities (e.g., processing, storage, and UI). Finally, we show that it is, indeed, feasible to use low-end smart device capabilities in a cooperative manner to meet application requirements.

Categories and Subject Descriptors

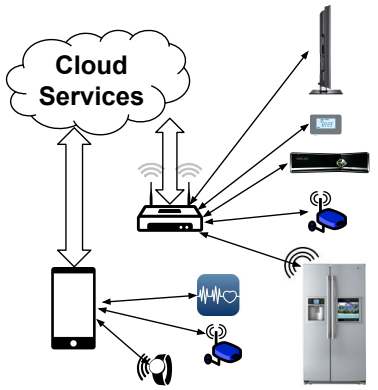
C.2.4 [**Computer Systems Organization**]: Computer-Communication Networks—*Distributed Systems*

Keywords

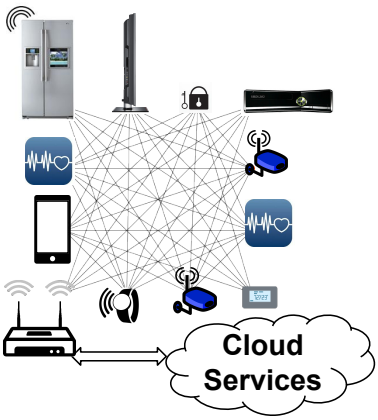
Internet of Things, Computation Offloading, Symbiosis

1. INTRODUCTION

The Internet of Things (IoT) is made up of everyday objects that are becoming more intelligent with enhanced processing, connectivity, and storage capabilities. These devices can be seen in both home and industrial or enterprise settings. Interestingly, a common feature of all smart devices is their need to interact with the user's cloud accounts and/or mobile devices [10, 13]. This interaction adds value to the user's experience in two different ways [6]: 1) Smart devices extend the accuracy, reach, or granularity of sensory inputs reported to the user's hand-held device and/or cloud services, 2) Smart devices extend the reach, functionality, and convenience of the user's interaction with the devices (e.g., smart TVs, hand-held device, smart fridges, etc). Within the current approach, depicted in Figure 1(a), smart devices survive off resources provided by mobile devices or cloud services. This hierarchical relationship taxes mobile device batteries and responsiveness by requiring constant connec-



(a) Current approach.



(b) Proposed approach.

Figure 1: Comparison between current approach that relies heavily on Internet connectivity and mobile devices, and the proposed approach where devices exist is symbiosis to lower the burden on Internet bandwidth and mobile devices.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.
MCS'15, September 11, 2015, Paris, France.
© 2015 ACM. ISBN 978-1-4503-3545-4/15/09 ...$15.00.
DOI: http://dx.doi.org/10.1145/2802130.2802133.

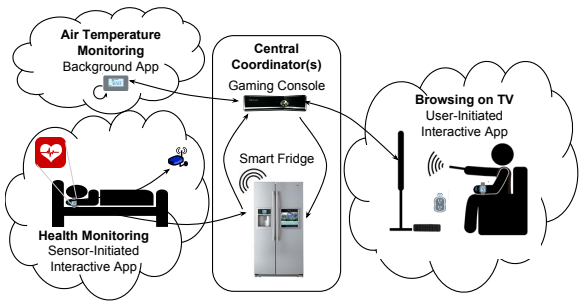


Figure 2: Operation Scenarios of SymbIoT.

tion to the phone. It can also be burdensome on Internet and cloud resources by streaming large volumes of data for processing and storage (e.g., cloud security IP cameras [1]).

We envision a new approach (Figure 1(b)) where all devices are 1) *logically* mesh connected either directly through Bluetooth or indirectly through WiFi, and 2) cooperate in a *symbiotic* fashion to perform different tasks. Our vision relies on the fact that smart devices are becoming more powerful with better processors, large storage, better connectivity, and some of them are even plugged in. It is also enabled by new industry initiatives to facilitate IoT device-to-device communication (e.g., Google's Weave [2]). In this vision, IoT devices exist in symbiosis, leveraging each other's strengths, to lessen the burden on the network and the user's mobile devices.

Figure 2 shows a scenario that instantiates our vision. In this example, a health monitoring application collects a person's vital signs, including EEG, ECG, and breathing patterns. These signals are analyzed by the nearby thermostat and fridge, and alerts are displayed on the TV in another room where the gaming console coordinates the whole process. In this example, powerful devices (e.g., gaming consoles) are used to orchestrate the distribution of tasks between other devices. The powerful devices also orchestrate network connectivity to allow only meaningful links of the mesh network to exist. Finally, these devices decide which of them are used for interaction with the user based on the user's location and the availability of UI enabled devices near the user (e.g., a TV screen).

In this paper, we answer the following three questions, that arise when considering the new approach, through instantiating this vision in a system we call *SymbIoT*.

1) *What are the goals of the symbiotic relationship between smart devices?* We consider goals such as lowering the burden on Internet bandwidth, trying to match the cloud's performance, and better utilizing available computing resources provided by these smart devices (§ 3).

2) *What is the architecture required to accommodate heterogeneous IoT devices in SymbIoT?* We propose an architecture where devices can assume different roles based on their capabilities and the user's preferences (§ 4).

3) *Is there enough processing power in lower end IoT devices to perform useful tasks?* We demonstrate that 5-20 Raspberry Pis, which represent lower-end smart devices, can match the end-to-end performance of cloud services under different RTT values. This means that a SymbIoT-like deployment can be useful for security IP camera deployments which are one of the most Internet bandwidth taxing applications (§ 5).

2. MOTIVATION FOR SYMBIOSIS

The current architecture for IoT cloud services aims for extending the current approach for Mobile Cloud Computing to encompass not only mobile devices, but also smart devices that form the IoT. This centralized architecture suffers several drawbacks which are all related to the centralized and hierarchical nature of the approach which 1) taxes the network, 2) requires that the centralized capacity scale with load, and 3) causes private data to travel through the network for processing and storage. In this section, we discuss each of these drawbacks along with how symbiosis between all connected devices can help in their mitigation.

1) Bandwidth: IoT is predicted to overwhelm the network's capacity within only the next three years. Cisco predicts that as the number of connected devices nears 8 billion, by 2018, the amount of IP traffic will increase four fold [8]. This is predicted to result in network congestion and poor quality connections. Moreover, IDC predicts that IoT generated traffic will turn 50% of networks from having excess capacity to being network constrained [15]. Symbiosis alleviates this problem by allowing devices to cooperate to process data generated in their vicinity. This allows only processed and extracted information to travel over backbone networks instead of raw data. While this approach does not solve the problem of the local networks, which suffer under both scenarios as raw data will still need to travel through local networks either to travel outside or internally, this approach allows for relieving backbone networks.

2) Latency: The rise of IoT is allowing for several applications that are important and will form a critical component of our daily lives in the future (e.g., health monitoring applications). Critical applications are usually sensitive to latency which has been a rich topic of research for mobile cloud computing [11]. This RTT-based degradation is likely to be even more pronounced as networks become more congested in the future [8]. Symbiosis follows a similar approach to solving latency problems, by bringing processing nodes closer to the edge. Moreover, symbiosis allows processing nodes to be within the same network and allows for an even better integration between devices beyond processing (i.e., devices can even use other nearby device's UI).

3) Privacy: As more devices get connected, finer grain details of everyone's personal lives will be recorded, directly and indirectly. Complete reliance on the cloud implies that all this data will travel over the network to be processed and stored on the cloud. This architecture severely jeopardizes personal privacy at an unprecedented level. Symbiosis allows raw data to remain useful to the user by processing it locally. It also allows users to make use of the cloud for further archiving or more fine grained processing, but only if that meets their privacy preferences. It can also provide logs that clearly identify data that travels outside the local network to keep the network owner updated of their privacy status.

4) Cost: Capacity issues, including network capacity and cloud's processing capacity, can be handled by expanding the infrastructure. Symbiosis allows for resolving the same issues by relying on devices that already exist on the network. In particular, instead of needing more servers, symbiosis allows for perceiving every new purchased smart device as a potential server. Smart devices don't necessarily use their processors continuously (e.g., smart light bulbs and thermostats). Using these devices' idle cycles can help reduce

the requirement for increasing cloud and network capacity and its associated costs.

Related Work: Several current hot topics in mobile cloud computing try to address several of these issues. However, to the best of our knowledge none of the approaches allow for addressing all four issues simultaneously. Fog Computing [3] and cloudlets [11] are two technologies that envision having more computation resources near the edge to lower latency and the burden on backbone networks. However, large scale deployments of such resources on the edge remains a significant challenge. Furthermore, relying on services of devices from outside the local network doesn't handle cost and privacy issues.

Another related line of work is the recently introduced concept of opportunistic cooperative computing. This endeavor explores challenges and potential benefits of systems where a mobile device offloads some of its tasks to nearby devices. Several aspects of this problem have been tackled including computational offloading [12], peripheral device sharing [6], and mobile devices clustering and offloading scheduling [5]. However, this work focuses primarily on the user's mobile device as the central piece of the system where all the attention of the user is directed.

The approach we propose to tackling this problem is closest to earlier work on resource allocation in sensor networks (e.g., [7]). The network of devices we are interested in has dramatically evolved in the past 10 years which we believe merits revisiting the problem based on the new motivation discussed in the previous section.

3. SYMBIOT DESIGN GOALS

One of the main questions we answer is what goals SymbIoT should achieve. The system goals, listed below in order of importance, derive from the motivation presented in the previous section.

1) Reducing Internet bandwidth consumption: One of the main concerns that are prevalent is the amount of data that is going to travel through the Internet generated by IoT devices (e.g., IP cameras, environment monitors, health monitoring systems, and traffic control and monitoring systems) and what the burden this represents on the network resources. SymbIoT allows the local processing of contextual information and the control of information traveling through the Internet. SymbIoT's control is based on the user's preference and the importance of the data (e.g., sending "no intruder detected" instead of streaming surveillance video).

2) Matching and improving on the cloud's performance: SymbIoT allows for improving the performance of smart applications by reducing the network delay (i.e., all tasks are executed locally) and employing more resources (i.e., more processing cores), as local devices get more powerful. These characteristics can even allow for improving the responsiveness of emergency systems and can enable applications that make use of the readily available contextual information, instead of typically having to wait for processing on the cloud.

3) Improving resource utilization: Most of the smart devices introduced in a modern setup are equipped with significant computational capabilities to handle peak utilization periods, partly due to the decreasing cost of equipping everyday objects with such resources [4]. This implies that these devices will be under-utilized most of the time, partly because the user is unlikely to use most of those devices simultaneously. SymbIoT aims at making idle device resources available to other devices that are in current use.

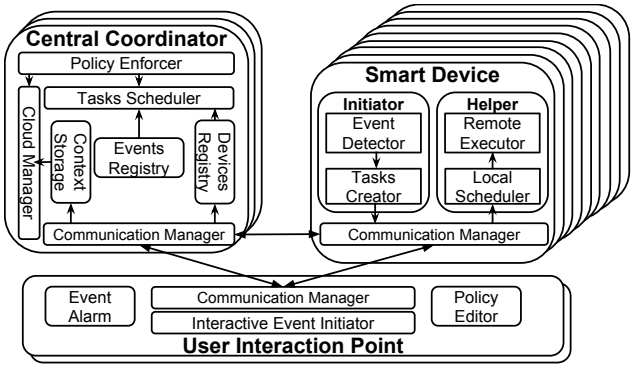


Figure 3: SymbIoT Architecture. Different roles are satisfied by devices available in the environment depending on its capabilities and user defined policy.

Satisfying SumbIoT's goals can also produce significant byproducts. These include: 1) protecting users privacy by reducing the granularity of the information that travels outside the home network and increasing the control over selecting which data to travel while giving the option for processing everything locally, 2) reducing the load on the cloud and, therefore, saving its resources for cost efficient tasks, 3) preserving energy of battery operated devices by allowing for Bluetooth Low Energy (BLE) to be the main means of communication between the local network devices instead of relying on a centralized points of communication that are typically used to connect to the Internet.

4. SYMBIOT ARCHITECTURE

SymbIoT is a system that can be used to allow heterogeneous IoT devices to cooperate to achieve common goals. These goals are prioritized based on their urgency from a user's perspective. Tasks to achieve these goals are then distributed on devices such that deadlines, which are set based on the nature of the tasks, are met. Devices assigned a task are responsible for executing it without affecting the expected performance of their primary function for which they are deployed. It is important to note that those devices will be typically purchased for very specific, and diverse, purposes while SymbIoT aims for using these devices for any generic purpose.

4.1 Overview

SymbIoT's processes have three modes of operation: 1) sensors-triggered interactive processes (i.e., processes with tight deadline requirements), 2) user-triggered interactive processes (i.e., processes with realtime interactive requirements), and 3) background processes. This gives SymbIoT the flexibility to allocate processing, storage, and communication resources as required for each process. It also allows for prioritizing allocation based on task requirements and sensitivity. Figure 2 shows different scenarios of SymbIoT with each scenario having a process that runs in a different mode.

The diverse nature of devices also mandates different modes of operation for each device. Figure 3 shows the architecture of the proposed system. The system has three main roles that can be satisfied by one or more devices from the pool of devices available in the environment. The role that each device can play is based on its capabilities (e.g., de-

vices without a user interface cannot be used as User Interaction Points), their current status (e.g., devices currently used by the user might not be a good choice to be coordinators), and user and vendor defined policy and preferences. A Central Coordinator keeps track of devices and events, assigns tasks, distributes resources, and enforces user policy. A Central Controller is a plugged powerful device with extra storage (e.g., PC or gaming console). User Interaction Points are devices with touch screens or other I/O peripherals and are assigned their tasks based on the user's location and the user's preferred mode of interaction. Finally, Smart Devices are other IoT devices that are not playing either of the other two roles. They are responsible for helping each other by providing extra processing or storage capabilities. We note, however, due to the opportunistic nature of the system, SymbIoT tasks can only preempt each other and not the main task of a certain device. This process is analogous to dynamic spectrum access scenarios where secondary users must yield usage of resources to primary users.

While all the components presented in Figure 3 are essential, we investigate specific components of the system with a focus on demonstrating SymbIoT's feasibility, value, and its ability to satisfy its design goals. In particular, we describe our initial take on *task configuration and generation*, *nodes configuration advertisement*, and *scheduling procedures*. Finally, we highlight open problems and challenges that should be resolved to realize a full scale deployment of SymbIoT.

4.2 Tasks Configuration

SymbIoT is designed to accommodate heterogeneous tasks that vary in priority, possible peripherals, and required computational resources. Hence, an important goal is to allow a flexible task abstraction to accommodate various types of tasks. There are two sources of labeling information for each task, the user which identifies priorities and possible peripherals, and the system which estimates required computational resources.

User's Perspective Tasks classified as *urgent* or *critical* require that all resources at hand (i.e., devices with idle cycles) should be marshaled to solve this task. *Critical* tasks include health monitoring and intervention applications (e.g., fall detection, seizure detection, and heart monitoring) and security applications (e.g., intrusion detection, localization, and identification). It is clear that for such applications, periodic, light weight, processes will be continuously monitoring the state of the system. Tasks classified as *real-time* require that low latency and responsiveness are imperative. This means that more network bandwidth, graphics processing, and memory should be allocated to such tasks. The third class of tasks is *background* which means that these tasks can be preempted or delayed by higher priority tasks. The user identifies a sorted list of devices, from the pool of devices available on the network, as preferred methods of interaction with each task. Depending on the task priority, SymbIoT tries to satisfy user preference.

Developer's Perspective A developer of SymbIoT applications should aim at parallelizing the tasks developed whenever possible. This allows SymbIoT to represent each task as subtasks which inherit all the task's user defined features. However, each subtask might require different amount of computational resources. The task configuration should specify explicitly its subtasks and the subtask configurations.

4.3 Device Configuration

An IoT device can play several roles as shown in Figure 3. This requires a careful abstraction of a device's capabilities to ensure the maximization of the utilization of that device. Based on a device's configuration file, the coordinator node can assign different roles to that device (i.e., a backup central node, a smart device, or a user interface). The central node registers events that can be generated by that device and the nature of the tasks triggered by each event. This allows smooth scheduling of tasks once an event is triggered.

A device's configuration should specify its capabilities. Information on the device's processing, memory, and storage capabilities and installed software packages allow for accurate task distribution that realistically meets the user's demand. Furthermore, specifying the device's UI peripherals and possible interaction mechanisms (e.g., screen, voice, touch screen, gestures, keyboard, and/or mouse) allows for proper displaying of alarms for events and other interactive features that the system can support (e.g., seamless switch of screens as the user moves).

4.4 Task Scheduling

We propose following a simple, yet effective, approach for scheduling tasks discussed in [14]. This approach is suitable for systems with heterogeneous tasks that are clearly ordered with respect to priority. While this approach is used to deal with massive systems (i.e., Google's infrastructure), we believe that the same concept is suitable for this domain due to its flexibility and suitability of heterogeneous environments. We describe the approach briefly.

When a task is initiated, it is added to the pending tasks queues, which are divided based on priority. The task is added to the queue that corresponds to its priority and the scheduling algorithm scans queues in a round robin manner to make sure that tasks with lower priority are not starved by higher priority tasks. The scheduler performs two tasks: 1) *Feasibility checking* to decide if there are enough resources in the network to allow for an execution of the task given the user defined constraints; this also includes preemption of lower priority tasks to allow for the execution of higher priority tasks. 2) *Scoring* allows the algorithm to rank each of the available machines based on its suitability for the task which is decided based available resources on that machine, and user's interaction preferences.

5. SYMBIOT'S FEASIBILITY

In this section, we demonstrate the potential power of SymbIoT using a video processing application. The purpose of this implementation is to show that even lower-end smart devices, previously perceived as devices with extremely limited resource, can now be used for useful computations. We choose an application where tags are detected and labels are imposed on pictures. Such tagging systems can be used, for example, to help the elderly find tagged objects. Our implementation is based on Acruo [9] augmented reality library.

(a) Cloud setup. (b) SymbIoT setup.

Figure 4: Diagrams of the experiments layout.

Video Processor	Network Type	RTT	Processing Per Frame	Latency Per Frame	Internet Bandwidth Used	Cost
Cloud (Figure 4(a))	Emulated minimal latency Emulated maximal latency	30 ms 220 ms	17 ms	46 ms 198 ms	69.1 MBps 13.9 MBps	\$9.95-\$29.95 per month [1]
SymbIoT (Figure 4(b))	LAN with 1 Raspberry Pi LAN with 2 Raspberry Pies	< 1 ms	553 ms	1131 ms 542 ms	0 Mbps	Free

Table 1: A comparison between cloud and SymbIoT implementations of a video analysis service.

Our experiment setup consisted of the following pieces. A Dell laptop with a webcam to stream pictures at a rate of 10 fps with each frame of 2.7 MBytes, a Raspberry Pi model B, and a Raspberry Pi model B+ both with a 700 MHz processor and 512 MB memory. All devices are connected over Ethernet assuming that all devices in this setup are plugged in. We compare two scenarios: 1) The video stream is sent to the cloud for processing where our cloud is the Dell laptop with the communication between the client and server emulated based on the statistics mentioned in [11]. 2) The video stream is processed locally either using one local device or distributed between the two Raspberry Pis. Figure 4 illustrates the two setups we used.

Table 1 summarizes the results of the experiments. The table shows the scenarios where either the cloud is used by the IP camera for processing while the rest of the smart devices nearby are left idle, and the scenarios where smart devices cooperate over the home LAN. The implementation allows the client to continuously send frames to the server for processing and frames are buffered. This enables the client to exhibit an inter-frame delay that is slightly lower than the overall trip time of each frame.

The results show that for high RTT values the cloud sustains 5 frames per second. The results also show that the number of Raspberry Pis available is roughly equal to the number of frames per seconds processed because frames can be distributed and collected from these processors in a round robin fashion. This means that only 5 and 20 Raspberry Pis can match the performance of a cloud for high and low RTT values respectively. Note that only cloud latency was emulated which means that bandwidth constraints can also significantly hinder the frame rate in the cloud's case. We note also that a typical home includes several devices that are much more powerful than a Raspberry Pi (e.g., gaming consoles, smart TVs, and even modern smart thermostats which are equipped with 800 MHz processors). The table also shows the amount of Internet bandwidth and the cost paid for a typical IP security camera cloud service. We don't report the local network usage as it will be identical in both cases. However, SymbIoT allows for eliminating the extra cost of this type of applications by performing processing, storage, and network interactions locally.

6. CONCLUSION AND FUTURE WORK

In this paper, we presented a vision of IoT devices operating in symbiosis to improve the user's experience and to reduce core network resource consumption and cloud services dependence. We presented our view of the design goals and architecture of a system that enables this vision. We demonstrated that even the low-end IoT nodes are capable of performing useful computations. Several challenges are yet to be addressed to fully implement the SymbIoT vision. These include 1) resource discovery and virtualization to facilitate the usage of resources of one device on the network by other devices, 2) local network configuration to select and enable a useful subset of links from the potential mesh of links among local IoT devices and to select the suitable technology for each link (e.g., BLE, WiFi, and Ethernet), 3) policy definition and interpretation to ensure proper usage of devices and smooth execution of tasks.

Acknowledgment

This work was supported in part by the US National Science Foundation through grants NETS 1409589 and NETS 1161879.

7. REFERENCES

[1] dropcam by Nest. `https://www.dropcam.com/cloud-recording`, 2015.

[2] Google's Project Brillo. `http://developers.google.com/brillo/`, 2015.

[3] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. Fog computing and its role in the internet of things. In *Workshop on Mobile Cloud Computing (MCC)*, 2012.

[4] P Brody and V Pureswaran. Device democracy: Saving the future of the internet of things. *IBM*, 2014.

[5] Karim Habak, Mostafa Ammar, Khaled A Harras, and Ellen Zegura. Femtoclouds: Leveraging mobile devices to provide cloud service at the edge. In *IEEE CLOUD 2015*.

[6] Minsung Jang, Karsten Schwan, Ketan Bhardwaj, Ada Gavrilovska, and Adhyas Avasthi. Personal clouds: Sharing and integrating networked resources to enhance end user experiences. In *INFOCOM*, 2014.

[7] Geoffrey Mainland, David C Parkes, and Matt Welsh. Decentralized, adaptive resource allocation for sensor networks. In *NSDI*, 2005.

[8] Stephan Monterde. Cisco Technology Radar. *White Paper, December*, 2014.

[9] R Munoz-Salinas. Aruco: a minimal library for augmented reality applications based on opencv, 2012.

[10] Charith Perera, Arkady Zaslavsky, Peter Christen, and Dimitrios Georgakopoulos. Context aware computing for the internet of things: A survey. *Communications Surveys & Tutorials, IEEE*, 16(1):414–454, 2014.

[11] Mahadev Satyanarayanan, Paramvir Bahl, Ramón Caceres, and Nigel Davies. The case for vm-based cloudlets in mobile computing. *Pervasive Computing, IEEE*, 8(4):14–23, 2009.

[12] Cong Shi, Vasileios Lakafosis, Mostafa H Ammar, and Ellen W Zegura. Serendipity: enabling remote computing among intermittently connected mobile devices. In *MobiHoc*, 2012.

[13] Salvatore J Stolfo, Malek Ben Salem, and Angelos D Keromytis. Fog computing: Mitigating insider data theft attacks in the cloud. In *Security and Privacy Workshops (SPW)*, 2012.

[14] Abhishek Verma, Luis Pedrosa, Madhukar Korupolu, David Oppenheimer Eric Tune, and John Wilkes. Large-scale cluster management at google with borg. In *EuroSys*, 2015.

[15] Jonathan Gaw Ruthbea Yesner Clarke Mario Morales Bob Kraus Vernon Turner, Carrie MacGillivray. IDC FutureScape: Worldwide Internet of Things 2015 Predictions, 2014.