Aura: An IoT based Cloud Infrastructure for Localized Mobile Computation Outsourcing

Ragib Hasan, Md. Mahmud Hossain, and Rasib Khan
{ragib, mahmud, rasib}@cis.uab.edu
Department of Computer and Information Sciences
University of Alabama at Birmingham, AL 35294, USA

***Abstract*—Many applications require localized computation in order to ensure better performance, security, and lower costs. In recent years, the emergence of Internet-of-Things (IoT) devices has caused a paradigm shift in computing and communication. IoT devices are making our physical environment and infrastructures smarter, bringing pervasive computing to the mainstream. With billions of such devices slated to be deployed in the next five years, we have the opportunity to utilize these devices in converting our physical environment into interactive, smart, and intelligent computing infrastructures. In this paper, we present Aura – a highly localized IoT based cloud computing model. Aura allows clients to create ad hoc clouds using the IoT and other computing devices in the nearby physical environment, while providing the flexibility of cloud computing. Aura provides localized computation capability from untapped computing resources. Computations done on Aura are highly flexible, giving clients full control to start, stop, migrate, and restart computations in nearby devices as the clients move between different physical locations. To demonstrate the feasibility of Aura, we have ported a lightweight version of MapReduce to run on IoT devices, and evaluated its performance.**

***Keywords*—Mobile cloud, Internet of things, MapReduce**

I. Introduction

The emergence of cloud computing has created a major shift in computing. Since 2005, the technology behind clouds has advanced greatly to provide low-cost but highly scalable distributed computing services. However, cloud service providers use large data centers that are geographically distant from their clients. Amazon Web Services, for example, has its massive cloud data centers located in 11 regions throughout the world [1]. A client therefore must send and receive its data over long distances through the public Internet when using such clouds. For highly interactive and time critical services, especially for mobile clients, such longer latencies can cause slow performance, security issues, and application availability problems. To provide optimal performance and minimal data movement between the client and the cloud, it would be better if the cloud is physically close to the client and moves as the client changes her location over time.

As an example, let us consider a cloud-enabled mobile application running on a smart phone. Due to the lower computation capability of mobile phone processors, such an application would ideally offload all the computation to the cloud, and provide only the visual display of the application on the phone. However, traditional clouds located hundreds or thousands of miles away from the mobile phone client may have large latencies, resulting in sluggish performance. Also, the application might deal with sensitive information which the client may not want to send over the public Internet to the cloud. An ideal solution can be achieved if the cloud is physically near the client's mobile phone so that all communications occur over one or two network hops from the client, and the data never has to traverse over the public Internet. On the other hand, building cloud infrastructures is very expensive, requiring hundreds of millions of dollars to set up and operate cloud data centers, making it impossible and economically infeasible to have data centers located near clients. For such localized computing facilities, we need a lightweight system for outsourced computation that can be incorporated into each building or physical infrastructure, making it available at a close distance from clients. To achieve this, in this paper, we present *Aura* – a system for outsourced computation on lightweight ad hoc clouds built using Internet of Things (IoT) devices.

As we enter the age of the Internet of Things (IoT), many devices in our everyday environment have gained significant computing capabilities as well as network connectivity. Smart thermostats, intelligent cars, programmable and Internet-connected appliances and LED light bulbs – all of these devices contain powerful processors and the ability to perform computations [2]. According to Business Intelligence, there were 1.9 Billion smart IoT devices in deployment as of December 2013, a number that will rise to 9 Billion by 2018 [3]. Another estimate from Gartner predicts that by 2020, the number of smart IoT devices will rise to 26 Billion with the market exceeding $300 billion [4]. IoT devices will outnumber smartphones, laptops, tablets, and traditional computers (7.3 Billion) by four times [3]. The rise in the number and usage of IoT devices will mean that in any building, there will likely be hundreds, if not thousands, of such devices with Internet connectivity and computing capability.

While most IoT devices have a dedicated purpose (such as environmental control with a smart thermostat), they have sophisticated processors and available memory that can be utilized for general-purpose computations. At present, these processors are not utilized fully, and are lying idle most of the time. The rise in the number of IoT devices capable of computing gives us an opportunity to make our buildings smarter and capable of running a computing in the infrastructure. That is, by leveraging the IoT devices in creating a computing fabric, we can make our buildings and infrastructures smart and make them "think" in the practical sense. We argue that, by combining the computational capabilities of such IoT devices, we can build a small-scale localized cloud.

Researchers have looked at distributed computing from various angles, yet the focus has remained on using desktops or unused servers in a given institution (as in Condor [5]) or all over the world (as done in SETI@Home, Folding@Home [6]). However, in all such projects, the computation and data have to traverse over multiple hops between the client and the

servers, which causes the latency and data security issues we mentioned earlier. An IoT based cloud can solve the issue by providing computation very close to the client, while providing the flexibility of the cloud computing model.

In this paper, we present Aura – a system for truly localized computation outsourcing using an IoT based lightweight cloud. Aura can be used to offload computations from a user's mobile device to the large number of IoT devices present in a building environment, allowing fast processing through the redundancy and scalability. The data and computations always remain close to the user. As the mobile client moves between physical locations, the data and computations are transparently migrated to the new computing environment where the computation can be resumed on the intermediate results, thus keeping the computation in proximity of the user. Aura also provides a tiered incentive model using which users can negotiate with the IoT devices to reach an optimal contract via micro payments to the devices for the work done. This incentive model ensures that the device owners have an interest to devote the unused cycles on their IoT devices for outsourced computations.

Contributions. The contributions of this paper are as follows:

1) We present Aura – a highly localized and mobile ad hoc cloud computing infrastructure using low-power IoT devices present in the building environment.
2) We provide an incentive and contract mechanism which allows building and operating such a system in an economically feasible manner.
3) We implemented Aura on the popular Contiki operating system for IoT devices [7], and ported a lightweight MapReduce prototype [8] to the IoT devices.
4) We demonstrate the feasibility and performance analysis of running such MapReduce operations on IoT devices.

Organization. The rest of the paper is organized as follows: Section II provides background information on IoT and cloud computing and the motivation behind IoT based localized clouds. Section III looks into the challenges in building such a system. Section IV provides a discussion of related research work. We present Aura and discuss its architecture in detail in Section V. We then discuss the implementation details and evaluation results from Aura in Section VI. We discuss the overall system and comment on the security and privacy aspects in Section VII and conclude in Section VIII.

II. Background and Motivation

A. Background

Internet-of-Things. Internet-of-Things refer to uniquely identifiable and Internet connected devices with computing and sensing and/or actuation capabilities [2]. IoT devices include smart thermostats, light bulbs, biochips, automobiles, sensors, automation systems, etc. The IoT devices can communicate over Wi-Fi (802.11.x), 6LoWPan (802.14.15) or through wired connections. Current application of IoT devices includes environmental monitoring, infrastructure management, industrial actuation applications, building and home automation, and medical/healthcare systems. The small scale and embedded nature of IoT devices mean that in any urban environment, there will be thousands of such devices, with current estimates of 1000 to 5000 trackable devices around a given user [2]. Such devices are and will be characterized respectively by their small sizes, low-power usage, limited range, and sophisticated sensing and computing capabilities. Together, the IoT devices and traditional computing systems will form massive, hierarchical distributed systems with complex interactions and information exchanges. An example of a popular IoT device is Nest, a smart and Internet-capable thermostat developed by Google, which contains an ARM Cortex A8 CPU (1 GHz) [9]. This example IoT device shows the sophisticated computing capability available to current IoT devices, which we propose to leverage in building our Pico clouds. A Nest device could support a significant value chain above and beyond its current use cases given the extra capabilities inherent in the system.

Cloud Computing. It refers to a computing model with infinite and elastic resource scalability, on-demand provisioning, and pay-as-you-go payment model [10]. Traditional cloud systems require the creation of large data centers with thousands of servers. Clouds and utility computing models have been extremely successful in recent years, enabling inexpensive yet large scale computing to solve complex problems, and allowing businesses to enjoy the economies of scale. Researchers have also developed highly efficient distributed data processing programming models for cloud computing. A very popular data processing model is MapReduce [8], introduced by Google and widely used in its Hadoop implementation in clouds.

B. Motivation

An IoT based localized cloud infrastructure has several advantages over traditional clouds. As discussed in Section I, such clouds allow highly localized computing environments very near to the clients, while providing the flexibility and scalability of clouds. Next, building such a computing infrastructure allows the utilization of the unused computing capability of the IoT devices which is otherwise wasted. A localized cloud also provides computing services with access to rich contextual information. For example, the cloud can serve as a backbone to the location aware services providing access to information relevant to that specific location. It augments our buildings and infrastructures with a smart and intelligent computing fabric. On the other hand, a localized IoT cloud allows users to seamlessly integrate with new environments and have their information experiences follow them efficiently. Finally, such a cloud can be deployed easily in existing IoT devices with minimal additional expenses.

In our work, we want to create new cloud architectures using the low-powered IoT devices as the backend servers. In particular, IoT clouds may be the clouds of interaction, work offload, user experience, and collaboration, whereas data center clouds will continue to do the large scale big data analytics, HPC, and multi-tier business applications.

III. Issues and Challenges

In this section, we discuss various issues and challenges in building an IoT based cloud infrastructures.

IoT device operation. Each IoT device has its own core functionality. The formation of and IoT cloud should not severely impact the operation of the device. On the other hand, the core function of the IoT device in turn should not debilitate or jeopardize the computations outsourced to that IoT device.

Power. The extra computations and data transfers due to participation in an IoT cloud will cause additional power consumption. Therefore, we need smart algorithms and schedulers

that can intelligently run IoT cloud computations on the devices to minimize power usage. The cloud backend operations such as MapReduce operations should be modified in order to run efficiently on the low-powered IoT devices (by minimizing and aggregating data transfer operations).

Security. The security and privacy of users, computations, and data are important issues. As the users are using the computation facilities in various buildings they visit, the devices used in the IoT cloud may not be fully trustworthy. Therefore, the system should provide facilities for trustworthy computations and guarantees for confidentiality, integrity, and availability of the data. On the other hand, allowing a cloud service process to be run on the IoT device should not endanger the operation of the IoT device or the data generated on or processed by the device. Finally, the data and computation migration between the IoT based computing facilities present in various locations should be secure and trustworthy.

Incentive and Economics. Allowing IoT devices to be used as part of a cloud costs the device owner money. The owners will need a strong incentive to participate in such a cloud. On the other hand, mobile clients also need to determine whether running the computation in the nearby IoT based cloud will be feasible in terms of costs, as opposed to running it in the device itself or sending it to a conventional cloud.

Robustness. There can be significant churn in the number and performance of IoT devices participating in the cloud, which can drop off, or preempt the cloud compute task in favor of the device's own operations. The high churn rate will require a robust and failure-resilient redesign of cloud computing.

IoT Heterogeneity and Interoperability. The IoT devices use a variety of operating systems. Also, the capability and features available to each IoT device is different, ranging from devices with powerful processors to those with very low capacity processors. Therefore, providing sustainable performance from an IoT based cloud will be a challenge.

IV. Related Work

Researchers have explored distributed computation outsourcing in various directions. Beberg et al. discussed Folding@Home which uses highly distributed computation to determine protein structures [6]. The Condor system is a large scale distributed computing platform which runs over a heterogeneous set of servers [5]. mClouds is a mobile device based ad hoc cloud where mobile phones can form a cloud computing platform [11]. In [12], Hoang et al. described a mobile cloud architecture which uses sensors and mobile cloud nodes to collect and manage data from the environment. However, all of these proposed systems do not necessarily provide localized computation, do not include any incentive or bidding, and do not provide any task completion estimates or guarantees through contractual basis. In [13], Noor et al. presented CellCloud, a mobile cloud built using mobile phones where the mobile base station acts as the controlling node. They discuss a bidding scheme for negotiating micro payments made by task owners to the participating mobile phones. We can build on this model in order to create a feasible model of incentives for the IoT based cloud.

Researchers have also explored the problem of mobile computation offloading [14], [15], [16], [17]. Mobile cloud architecture, application model, time-constrained or real time task offloading scheduler and algorithm are proposed in [18], [19], [20], [21], [22]. Our work is complementary to the research on mobile computation offloading and can leverage the offloading or scheduling scheme on the mobile devices to determine optimal strategies for offloading tasks to IoT clouds.

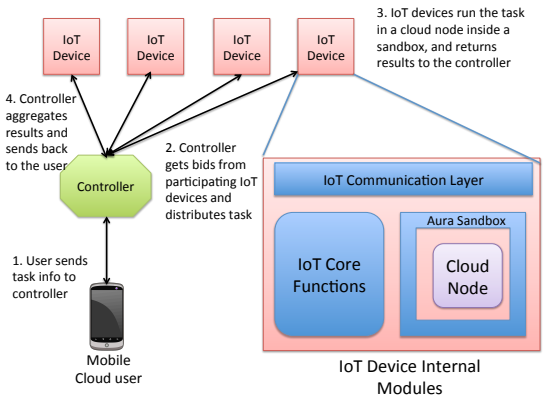

Fig. 1: Conceptual overview of the Aura model

V. Aura Architecture

Aura is built on top of a large set of IoT device workers. We assume that the IoT devices are located at fixed locations in a building or other physical infrastructure and have local network or Internet connectivity. The IoT devices run a special sandboxed process inside which the cloud functions are run in isolation from the other activities. The conceptual operation of Aura is shown in Fig. 1 and later explained further in Fig. 2.

A. Components

The Aura architecture has three major components. *a*) Mobile Agent(s), *b*) Controller(s), and *c*) IoT devices. Fig. 2 presents an overview of Aura's system model. Next, we describe the properties and functions of these components:

Mobile agents. The Mobile agents (M-Agent) are personal mobile devices: smart phone, laptop, etc. M-Agents are running applications that require offloading to the cloud. When a user enters a building, the M-Agent advertises the job along with a job description: time-to-finish job, outsourcing price, etc.

IoT devices. The IoT devices (integral part of Aura) perform the actual outsourced computation. Interested devices advertise their own device specifications, requirements and capabilities: computation speed, storage/ memory status, energy level, network and security protocols, etc.

Controller(s). A Controller (mobile-computation-broker) provides communicational and computational abstraction between IoT devices and the M-Agent, such that M-Agent does not have to deal with worker IoT devices directly. A controller receives job announcements, initiates discussion about job's pricing with interested IoT devices. If the proposed payment seem economically attractive, only then it accepts the job. A controller's roles and responsibilities consist of task break down (sub-tasks) and distribution , control command issuance (start, pause, resume, stop), and sub-tasks' progress monitoring.

B. Operational Model

Upon activation of an IoT device in a building, the device owner sets the configuration based on which the device

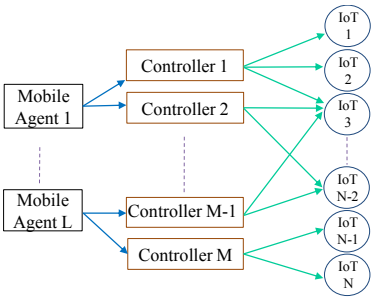

Fig. 2: Architecture of an Aura based system

advertises its availability for IoT cloud participation. Similarly, the controller node owner also sets up the configuration.

When a user enters a building, the M-Agent running on the user's mobile device advertises the jobs and initiates a negotiation round with the controller. Upon reaching an agreement, the computation code and the data are sent to the controller for distribution to the IoT worker nodes. When the computation is done, the resulting data is returned to the M-Agent via the controller. If at any point, the M-Agent wants to suspend operation, it can send a message to the controller. At that time, the computation is suspended and intermediate results are sent back to the M-Agent, which can take this computation state to a new location and resume the job at the Aura infrastructure at that location. Alternatively, the computation can be resumed at the mobile device natively, or sent to a conventional cloud. Next, we describe the advertisement and discovery protocol in detail.

Advertisement and Discovery Protocol. Devices (controller, M-Agents and IoT devices) join and leave Aura network at any time. After joining a network, it announces its presence which we denote as *entrance-advertisement*. Advertisement message contains device or job specific information such as job or device status , requirements, configuration, etc. When a device leaves the network, it should announce its departure through exit-advertisement message. Upon receiving the exit-advertisement, other components will be aware that this device is no more connected with the network. A device might also send discovery messages in search for other devices. For example, an IoT device discovers a controller by sending discovery messages in the Aura network. We use UPnP message standard for advertisement and device discovery, where messages will be sent to the multicast address 239.255.255.250 on port 1900 via the UDP protocol [23].

An advertisement message must starts with NOTIFY field. The NTS field in the advertisement is used to distinguish between entry (NTS:ssdp:alive) and exit (NTS:ssdp:byebye) messages. Advertisement message ownership is determined by the NT field. if an advertisement is made by an IoT device, then NT will be set to 'IoT Device' – the same is true for controller and M-Agent. There might have additional field in the advertisement for device specification (cpu speed, memory, per hour charge, etc.) and job description (job title, completion time, interrupted computation, etc.). A discovery message must start with M-Search. The search-target is determined by ST field. For instance, if ST is set to 'Controller', then someone (M-Agent/ IoT device) is looking for controllers. We illustrate the message format and its attributes in Fig.s 3, 4, and 5.

Let us consider a scenario for job advertisement. When

Entrance Advertisement	Exit Advertisement	Controller Discovery
NOTIFY * HTTP/1.1 HOST: 239.255.255.250:1900 NT: IoT Device NTS: ssdp:alive Pricing: $x per Hour Time to Live: 12 Hour CPU Speed: 700 MHz RAM: 400 MB Flash Drive: 1GB	NOTIFY * HTTP/1.1 HOST: 239.255.255.250:1900 NT: IoT Device NTS: ssdp:byebye	M-SEARCH * HTTP/1.1 HOST: 239.255.255.250:1900 MAN: ssdp:discover ST: Controller

Fig. 3: Advertisement and discovery message for IoT Device

Entrance Advertisement	Exit Advertisement	IoT Device Discovery
NOTIFY * HTTP/1.1 HOST: 239.255.255.250:1900 NT: Controller NTS: ssdp:alive	NOTIFY * HTTP/1.1 HOST: 239.255.255.250:1900 NT: Controller NTS: ssdp:byebye	M-SEARCH * HTTP/1.1 HOST: 239.255.255.250:1900 MAN: ssdp:discover ST: IoT Device

Fig. 4: Advertisement and discovery message for Controller

Entrance Advertisement	Exit Advertisement	Controller Discovery
NOTIFY * HTTP/1.1 HOST: 239.255.255.250:1900 NT: Mobile Agent NTS: ssdp:alive Job Title: Task-1 Completion Time: 20 Min Interrupted Computation: Y	NOTIFY * HTTP/1.1 HOST: 239.255.255.250:1900 NT: Mobile Agent NTS: ssdp:byebye	M-SEARCH * HTTP/1.1 HOST: 239.255.255.250:1900 MAN: ssdp:discover ST: Controller

Fig. 5: Advertisement and discovery message for M-Agent

arriving at a location with Aura, an M-Agent sends a discovery message in the network to look for a suitable controller. It cannot outsource its job until it finds a controller. It also advertises its identity and potential job description which includes additional fields such as 'Job Title', 'Job Completion Time', etc. 'Interrupted Computation' field is set to Y, denoting that agent might request the controller to issue stop command in the middle of computation, and also the controller needs to send back the intermediate result and computation states to the M-Agent. The M-Agent might resume computation from the last saved state of computation in some later time, in some different IoT network with an Aura service (see Fig. 5).

Offloading and State/Result Preservation. We now present the information flow among the M-Agent, controller, and IoT Devices in a building (say building-1). Let us consider a scenario, where an Aura network is comprised of one M-Agent, one controller, and several IoT devices under that controller. Here, the sequence of events for computation outsourcing with state preservation happens in the following steps:

1) A new IoT device joins in the network, and multicasts its advertisement. *2*) The Controller receives and parses the advertisement message to learn about the new IoT device (computational capability, device-live-time, pricing, etc.). *3*) If the controller feels interested, then it sends an acknowledgement to the IoT device, otherwise it simply discards the message. *4*) Upon receiving the acknowledgement, the IoT device is connected with the controller. *5*) After sometime, a new M-Agent, which has a job to be outsourced, enters building-1. It joins the network and makes a job advertisement. *6*) The controller receives and parses the advertisement to get job details (job description, job completion duration, etc.). *7*) The controller then consults with its IoT devices to decide whether to take the job or not. *8*) If it decides to accept, then it states a price for that job. *9*) If the M-Agent agrees with the price quotation, then it submits the job to the controller. *10*) The controller splits the job in several sub-tasks, and sends those to different IoT nodes according to their capabilities and price for computation.

An exception might arise while IoT devices are performing the sub-tasks. The M-Agent might wish to stop the job before

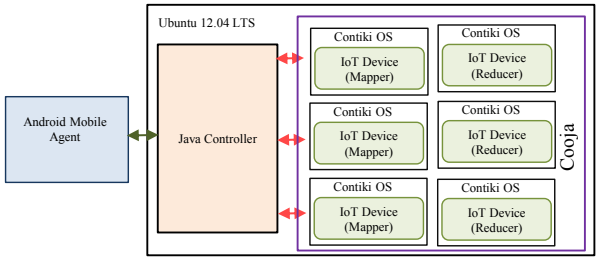

Fig. 6: Experimental environment

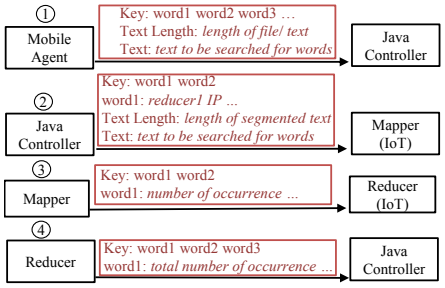

Fig. 7: Inter component message flow

it is finished, as the user moves from building-1 to building-2. In that case, it requests the controller to stop computations, and to send the intermediate result and computation state. The controller propagates the stop command to the participating IoT node, and asks them to send the intermediate results and computation states immediately. The IoT nodes do so and send the intermediate results and computation states to the controller, and ultimately to the M-Agent. Now, the M-Agent enters into building-2 and connects to its network. When it resumes the task, it is starting from where it left off in building-1.

Task Completion Failure Penalty. When an IoT device fails to complete a task within the given time frame, the M-Agent might make under payment, and/or assign a bad reputation to the controller for its unsatisfactory performance. The controller might do the same to the failing IoT device. As a result, the chance of getting further tasks, for both the controller and the IoT device, will be diminished due to bad reputation.

VI. Implementation and Evaluation

For the experimental purpose, we have created a proof of concept implementation of Aura. Our conceptual Aura system includes an Android application as the M-Agent, a desktop based Java application as the controller, and several virtual IoT devices running Contiki OS – a popular operating system for IoT devices [7]. To demonstrate the feasibility of running cloud based data flow computations on Aura, we ported MapReduce [8] to the Contiki platform and implemented a lightweight Contiki compliant MapReduce framework for IoT nodes. We deployed the mapper and reducer binaries to Tmote-SKY [24] IoT devices, which were simulated on Cooja [25] (see Fig. 6). Finally, we evaluated our implementation with the canonical MapReduce example problem: Word count.

In our experiments, the Android M-Agent posts a job (word-count) to the controller. The M-Agent running on the phone transfers a file to the controller application along with the set of words it wishes to be counted from the file's content. Upon receiving the file and set of words, the controller application splits the file and the words into several segments and distributes

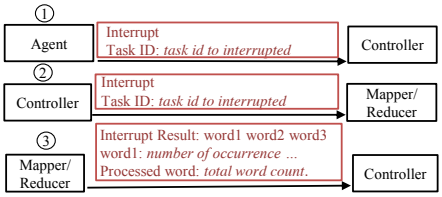

Fig. 8: Message format for interrupted task

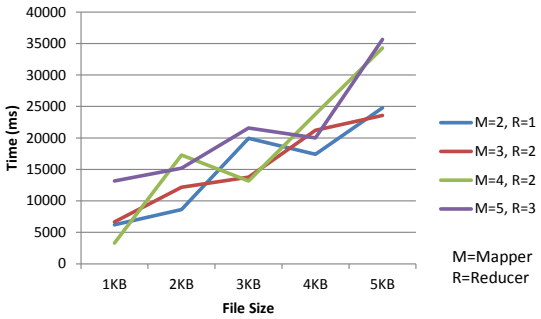

Fig. 9: Task completion time vs task size with variable workers

those to some mappers (IoT nodes). Mappers count the words, and send the total number of individual occurrences of the words to the reducer IoT nodes. The reducers do the final summation and send the result back to the Controller, which then relays the result to the android M-Agent. We use our own message format for the communication between android M-Agent and controller, and between controller and IoT devices: mappers and reducers (see Fig. 7). Our implementation also covers the features for computational-state and intermediate-result preservation for computation-stop request in the middle of an ongoing computation (see Fig. 8).

We evaluated the task completion time for various mapper and reducer configurations and file sizes, the results of which are presented in Fig. 9. Our experiment shows that it is quite challenging to find an optimal number of mappers and reducers to finish a job within the given time frame by the M-Agent. Usually, it might be thought that the involvement of more mappers and reducers will help to finish the job before the deadline, as execution of sub tasks will go in parallel. But in practice, things might be getting worse. We see from Fig. 9 that when the file size is 1 KB then only two mapper and one reducer did the job in the quickest time, while five mappers and three reducers gave the worst result. From the figure, we can infer that three mapper and two reducers give the optimal solution. Therefore, when a controller breaks down a task in to sub-tasks, it needs to consider the participant number for the whole task, so that the overall task completion is not affected negatively.

VII. Discussion

Security perspectives. IoT devices are still dependent on manufacturer specific black-box security [26]. Unfortunately, there are still many unresolved issues regarding the adoption of cloud-based models for services and computations using IoT devices [27], [26]. The distinct multi-device environment of Aura complicates the situation more than traditional service-oriented frameworks and are vulnerable to both insider and outside threats [28], active and passive attacks [29].

Auditing is important for security critical jobs [30]. Given the volatile nature of the framework, we realize that the devices

will have limited monitoring and auditing facilities. We may adopt external process documentation and provenance recording solutions proposed in other similar distributed architectures [31]. A solution to privacy of computation may be obtained using homomorphic security schemes [32], [33], [34]. Moreover, Device-to-device security in dynamic environments to protect individual communication channels can also be leveraged using host identity protocol [35], [36]. However, IoT devices are still limited in hardware resources, which may restrict the possibility of integration of strong security technologies.

Applicability of Host Identity Protocol. The proposed framework is based on IoT devices, which come along with their inherent properties of unstable connectivity and high mobility. Additionally, the clients are spacio-temporally co-located mobile devices with preferentially chosen controllers based on the proximity. The evolving nature of the network communication suits the dynamic behavioural support for HIP – a meta-layer protocol working in between the IP namespace and the application layer [35], [36]. HIP introduces a separation of the port-IP binding for the application layer using a Host Identity (HI) namespace. Therefore, Roaming mobile devices can maintain the connection with the controllers till when the process states are saved and transferred safely in case of incomplete tasks. Additionally, devices can perform mutual verification prior to beginning a service session [35], [36].

VIII. Conclusion

Internet of Things devices are becoming ubiquitous, and in the coming years, there will be thousands of such devices in our physical infrastructure. While the devices are typically equipped with low-end processors, the sheer number of such devices predicted to be present in any building allows us to successfully run a computation in a loosely formed cloud built using the IoT devices. In this paper, we presented Aura – a platform that achieves the goals of localized and highly scalable computation. Our proof of concept implementation of Aura on the Contiki platform as well as the simplified MapReduce port shows the feasibility of such a model. In future work, we want to extend the model and explore techniques for ensuring the security and privacy of the computation as well as migration. In addition, we will develop secure sandboxing techniques and scheduling algorithms which ensure that the core functionality of the IoT devices will not be affected when the devices participate in an IoT cloud. We will also test Aura on a large scale and heterogeneous IoT testbed.

Acknowledgment. This research was supported by a Google Faculty Research Award, the Department of Homeland Security Grant FA8750-12-2-0254, and by the National Science Foundation CAREER Award CNS-1351038.

References

[1] Amazon Inc., "Global infrastructure," Online at http://aws.amazon.com/about-aws/global-infrastructure/.

[2] A. Iera, C. Floerkemeier *et al.*, "The internet of things," *IEEE Wireless Communications*, December 2010.

[3] Business Intelligence, "Here comes the internet of things," Online at https://intelligence.businessinsider.com/the-internet-of-things-2013-10.

[4] Gartner Inc., "Forecast: The internet of things, worldwide," Online at http://www.gartner.com/newsroom/id/2636073, 2013.

[5] D. Thain, T. Tannenbaum *et al.*, "Distributed computing in practice: The condor experience," *Concurrency and Computation: Practice and Experience*, 2005.

[6] A. Beberg, D. Ensign *et al.*, "Folding@home: Lessons from eight years of volunteer distributed computing," in *Proc. of IPDPS*. IEEE, 2009.

[7] "Contiki OS," http://www.contiki-os.org/start.html.

[8] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Proc. of CACM*, 2008.

[9] S. Sheffer, "Nest thermostat teardown reveals ARM Cortex A8 CPU, ZigBee support," The Verge, http://j.mp/1zIZefj, Dec 2011.

[10] NIST, "The nist definition of cloud computing," Online at http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf, September '11.

[11] E. Miluzzo, R. Cáceres *et al.*, "Vision: mclouds-computing on clouds of mobile devices," in *Proc. of MobiSys*. ACM, 2012.

[12] D. Hoang and L. Chen, "Mobile cloud for assistive healthcare (mocash)," in *Proc of APSCC*, Dec 2010.

[13] S. Noor, M. Haque *et al.*, "Cellcloud: A novel cost effective formation of mobile cloud based on bidding incentives," in *Proc. of IEEE Cloud*, June 2014.

[14] M. Shiraz, A. Gani *et al.*, "A review on distributed application processing frameworks in smart mobile devices for mobile cloud computing," *Proc. of ICUFN*, 2013.

[15] X. Ma, Y. Cui *et al.*, "Energy optimizations for mobile terminals via computation offloading," in *Proc. of PDGC*. IEEE, 2012.

[16] E. Cuervo, A. Balasubramanian *et al.*, "Maui: making smartphones last longer with code offload," in *Proc. of MobiSys*. ACM, 2010.

[17] B.-G. Chun, S. Ihm *et al.*, "Clonecloud: elastic execution between mobile device and cloud," in *Proc. of EuroSys*. ACM, 2011.

[18] X. Zhang, A. Kunjithapatham *et al.*, "Towards an elastic application model for augmenting the computing capabilities of mobile devices with cloud computing," *Mobile Networks and Applications*, 2011.

[19] T. Justino and R. Buyya, "Outsourcing resource-intensive tasks from mobile apps to clouds: Android and aneka integration," in *Proc. of CCEM*, 2014.

[20] T. Shi, M. Yang *et al.*, "An adaptive probabilistic scheduler for offloading time-constrained tasks in local mobile clouds," in *Proc. of ICUFN*, 2014.

[21] A. Mtibaa, K. A. Harras *et al.*, "Towards computational offloading in mobile device clouds," in *Proc. of CloudCom*. IEEE, 2013.

[22] H. Eom, P. S. Juste *et al.*, "Machine learning-based runtime scheduler for mobile offloading framework," in *Proc. of CloudCom*, 2013.

[23] "Universal Plug and Play (UPnP)," http://www.upnp.org/.

[24] "Tmote sky," http://j.mp/1DiMGsd.

[25] "Cooja Simulator," https://github.com/contiki-os/contiki/wiki/An-Introduction-to-Cooja.

[26] C. M. Medaglia and A. Serbanati, "An overview of privacy and security issues in the internet of things," in *The Internet of Things*. Springer, 2010.

[27] H. Takabi, J. Joshi *et al.*, "Security and Privacy Challenges in Cloud Computing Environments," *Security Privacy, IEEE*, Nov-Dec 2010.

[28] W. Jansen and T. Grance, "Guidelines on Security and Privacy in Public Cloud Computing," NIST, Tech. Rep., 2011.

[29] T. Ristenpart, E. Tromer *et al.*, "Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds," in *Proc. of CCS*. ACM, 2009.

[30] Y. Chen, V. Paxson *et al.*, "What's new about cloud computing security?" EECS Department, University of California, Berkeley, Tech. Rep., 2010.

[31] P. Groth, M. Luck *et al.*, "A protocol for recording provenance in service-oriented grids," in *Proc. of OPODIS*, 2005.

[32] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proc. of STOC*. ACM, 2009.

[33] M. van Dijk and A. Juels, "On the impossibility of cryptography alone for privacy-preserving cloud computing," Cryptology ePrint Archive, Report 2010/305, 2010.

[34] R. A. Popa, C. M. S. Redfield *et al.*, "Cryptdb: Protecting confidentiality with encrypted query processing," in *Proc. of SOSP*, October 2011.

[35] R. Moskowitz, P. Nikander *et al.*, "Host identity protocol," *RFC5201, April*, 2008.

[36] L. Bokor, Z. Faigl *et al.*, "Survey and evaluation of advanced mobility management schemes in the host identity layer," in *Proc. of IJWNBT*. IGI Global, 2014.