Resource Constrained Offloading in Fog Computing

Ajay Kattepur[1], Harshit Dohare[2], Visali Mushunuri[1],
Hemant Kumar Rath[1] & Anantha Simha[1]
1 Embedded Systems and Robotics, TCS Research & Innovation, Bangalore, India.
2 Dept. of Electrical Engineering, IIT Madras, India.
ajay.kattepur@tcs.com, harshit13hd@gmail.com, visali.m@tcs.com,
hemant.rath@tcs.com, anantha.simha@tcs.com

ABSTRACT

When focusing on the Internet of Things (IoT), communicating and coordinating sensor–actuator data via the cloud involves inefficient overheads and reduces autonomous behavior. The *Fog Computing* paradigm essentially moves the compute nodes closer to sensing entities by exploiting peers and intermediary network devices. This reduces centralized communication with the cloud and entails increased coordination between sensing entities and (possibly available) smart network gateway devices. In this paper, we analyze the utility of offloading computation among peers when working in fog based deployments. It is important to study the trade-offs involved with such computation offloading, as we deal with resource (energy, computation capacity) limited devices. Devices computing in a distributed environment may choose to locally compute part of their data and communicate the remainder to their peers. An optimization formulation is presented that is applied to various deployment scenarios, taking the computation and communication overheads into account. Our technique is demonstrated on a network of robotic sensor–actuators developed on the ROS (Robot Operating System) platform, that coordinate over the fog to complete a task. We demonstrate 77.8% latency and 54% battery usage improvements over large computation tasks, by applying this optimal offloading.

CCS Concepts

•**Theory of computation** → *Linear programming;* •**Computer systems organization** → *Robotics;* Grid computing;

Keywords

Fog Computing; Networked Robotics; Optimization; ROS.

1. INTRODUCTION

Proliferation of the Internet of Things (IoT) has led to a flurry of research in smart homes, smart cities and energy efficient data center operations [1]. The Internet of Things typically combines the sense-actuate-compute capability of sensor networks, mobility patterns of cellular devices and large scale data volumes of Big Data applications.

Cloud computing [2] has further emerged as a central tenet in energy-efficient computing that allows on-demand scaling up/out of virtual machines. The cloud is typically used as a central storage repository for IoT devices; data stored centrally can be accessed for computation–actuation. While the elastic scaling and energy-efficiency of the cloud are appropriate in many cases, the kind of applications that do not typically fit in a central cloud environment include: a) Robotic applications needing low latency, near–real time control and actuation of devices; b) Geographically distributed sources that can have vast differences in data transfer and processing time-lines; c) Applications working in environments with intermittent network connectivity.

In order to coordinate the high volume, velocity, variety, variability of data that are produced in IoT applications, additional architectural modifications are needed. Performance bottlenecks in last mile connectivity and central cloud access are envisioned, as the number of IoT devices scales to trillions of devices [3]. *Fog Computing* [3][4] is motivated by moving data and computation close to the edge of the network by exploiting smart gateway devices. This is particularly suited for IoT applications where sensors are typically mobile, produce data at large volumes/velocity and require actuation to be completed on low latency time-scales. Fog computing builds on top of the edge computing paradigm, by reducing resource contention and scalability issues [3], while still maintaining low latency overheads. Typically any device with computing, storage, and network connectivity can be a fog node: controllers, switches, routers and embedded servers – virtual machines with relevant computational APIs may be deployed on them [4].

In this paper, we intend to evaluate the energy and latency improvements provided by fog computing. Our motivating scenario makes use of a network of robotic sensor–actuator devices in a confined room, that co-operate and communicate via the fog to complete a computation task. Such devices function in resource constrained environments (limited battery power, limited computational capacity) that need to be kept in mind while coordinating computational offloading. By making use of accurate battery usage and communication path loss, we provide an optimal deployment model, in which computation may be divided and offloaded to peer nodes. The scenario is studied with a deployment of robots in *ROS* (Robot Operating System) and the associated simulation environment *Gazebo*. It is shown to provide 77.8%

MECC '16, December 12-16, 2016, Trento, Italy

 ISBN 978-1-4503-4668-9/16/12…$15.00
DOI: http://dx.doi.org/10.1145/3017116.3022871

improvement in latency and 54% improvement in battery usage when compared to non-offloaded computations, for larger tasks. The principal contributions of our work are:

1. Emphasize the need for fog based distributed computation in mobile networked environments with resource constrained devices.
2. Accurately model the energy and communication costs with respect to computational offloading.
3. Formulate an optimal deployment strategy when dealing with distributed computation, while keeping energy and latency constraints in mind.
4. Demonstrating the advantages of optimally offloading computation over a network of mobile robotic sensor–actuators simulated in ROS/Gazebo, that coordinate to complete a task via the fog.

The paper is organized as follows: Section 2 provides a brief overview of fog computing. The description of a case study using a network of robotic sensor–actuators is presented in Section 3. In Section 4, the optimization formulation for deploying computations are presented, integrating resource models for battery, communication and computational expenditure. The analysis results for optimal deployment and computation in ROS are presented in Section 5. This is followed by related work and conclusions in Sections 6 and 7, respectively.

2. FOG COMPUTING TRADE-OFFS

Fog computing [3][4] has received some interest due to the advantages proposed over cloud centric deployments. Moving much of the storage/computation/actuation to the edge of the network can be beneficial for IoT application environments. Fog computing improves on the edge computing paradigm, by reducing resource contention and scalability issues [3]. Some of the benefits include: a) *Latency Reduction*: Reducing data transfer requirements can help shave off crucial seconds in the auto-actuation of devices b) *Network Bandwidth*: Optimizing the data that is sent over the network and necessary policies that are associated with edge-node computation can significantly lower network bandwidth requirements [5] c) *Improved Reliability*: By distributing computation and storage, reliability and redundancy of data can be further guaranteed.

A typical deployment example of an IoT application the the fog is presented in Figure 1. Based on geo-located mobile sensor data, fog compute nodes (eg. smart gateways) are deployed closer to the edge devices to handle computation and actuation. Periodic snapshots or analytics on the data are stored in the cloud data center. While the cloud may be used for long term analysis and goal setting, the geographically distributed set of mobile services make use of fog nodes for computation. Note that we make use of the terms *Fog* or *Edge* devices in a synonymous fashion: they refer to both end-devices (mobile computation nodes) and smart gateway devices that have computation capabilities.

While the advantages of offloading computation has been well studied in the grid-computing community [6], the specific tradeoffs involved in resource constrained IoT environments needs further analysis. Given a large computation that may be performed at a compute node, it may choose to perform the task locally, offload the entire task or use a hybrid approach. Constraints on this include:

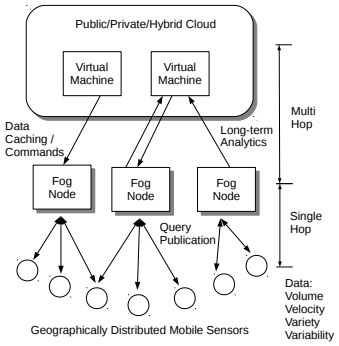

Figure 1: Mobile Devices with Fog Nodes.

1. *Energy Constraints*: Battery levels at the node that intends to offload data, target nodes and depletion associated with a set of computation units. Efficient energy management is one of the principal advantages of utilizing cloud based deployments as well.
2. *Communication Overheads*: Transmit–Receive power levels for offloading, throughput rates and additional latency incorporated into the offloading process.
3. *Computational Capacity*: Heterogeneous device CPU capacity, parallel processing and efficient task mapping to available nodes must be handled.

Incorporating all these constraints into fog/edge based processing is analyzed through a robotic case study and optimized deployments in the proceeding sections.

3. CASE STUDY: NETWORKED ROBOTICS

In order to develop a realistic scenario where mobile nodes sense, communicate and share computational load among themselves, we make use of a network of robotic sensor–actuators deployed in an indoor environment. The robots move around to collect data with on-board sensors such as location coordinates, orientation and obstacle mapping periodically. The objective is to perform a computationally intensive task over a large dataset that has been collected, in order to provide actuation in a short time-frame. Direct offloading to the cloud cannot be done due to intermittent network infrastructure and high end-to-end delays.

As tasks can be computationally expensive (e.g. indoor mapping), for limited capacity (CPU, memory, storage, battery) robots, collaborative techniques are necessary. The naive scenario is to perform the required computation locally or offload to one of the nearest peers. However, due to limited capacity and extended battery consumption of this process, this would mean reducing the lifetime of deployed robots and extended delays in computations. The alternative is to make use of the computational capacity of multiple peer nodes to offload parts of the task. In Figure 2, ① a robot intending to perform a large computation, divides a portion of the data and computation task into locally solvable units and then requests for potential resources (similar to grid computing environments [6]) via a coordinating robot/fog node. ② The coordinating node (which has information about the location, battery states and computation power of all robots using ROS `publish-subscribe`), allocates tasks to appropriate robots. ③ The completed computations from resources are returned to the coordinator, which in turn collates and returns results to the offloading device ④.

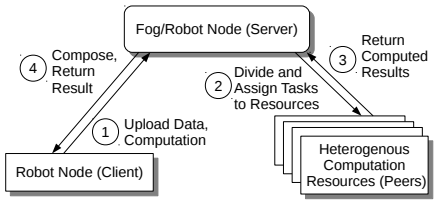

Figure 2: Distributed Computation on the Fog.

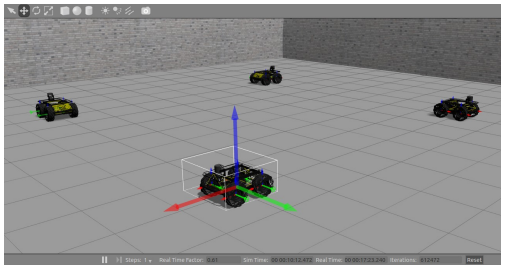

Figure 3: Husky Robots deployed in Gazebo/ROS.

As specified in [7], cloud robotic networks can follow proxy, peer or clone based topologies. Our architecture presented in Figure 2, resembles the peer based topology, with a centralized coordinating entity to allocate and collate tasks. This entity can be randomly allocated or selected based on location or computational capacity. While we have presented the fog node as the coordinating entity, another peer can also be chosen to perform this optimal deployment. The robots are deployed in an indoor environment, where they may make use of infrastructure based or infrastructure less (ad-hoc) communication (cloud connectivity unavailable).

For application development, we make use of ROS (Robot Operating System) (http://www.ros.org/), an open source programming language that can be ported to multiple robot hardware configurations. It provides services such as hardware/sensor abstractions, device control, message passing paradigms and portability across devices. ROS models may be developed in conjunction with a realistic simulation environment called Gazebo (http://gazebosim.org/), that incorporates physical constraints. The simulation in Gazebo is presented in Figure 3, with four Husky [1] robots simultaneously moving in an indoor environment.

While ROS typically uses the `publish-subscribe` messaging system to handle interactions between nodes, we implement the offloading of data using the `client-service` paradigm (Figure 2). `client-service` ensures bi-directional transfer of data combined with one-to-one communication. The client node *requests* for some data and waits for the reply; the server receives the request, performs the required computation and sends a *response* to the client. Note that while we make use of a network of robots in this case study, it can be extended to include smart gateway devices (switches, routers)[3] or additional compute-only devices.

4. OPTIMIZED DEPLOYMENT

In this section, we analyze some of the crucial deployment aspects that are associated with the fog. Models to consider include battery resources, communication path loss and computation on devices with heterogeneous hardware. An optimization formulation is presented keeping these constraints in mind.

[1]https://www.clearpathrobotics.com/husky-unmanned-ground-vehicle-robot/

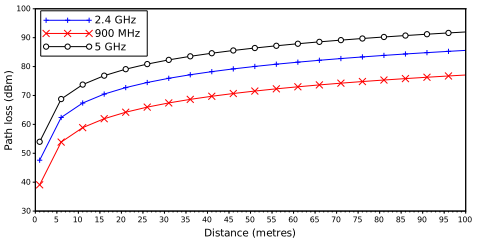

Figure 4: Path-loss vs. Transmit Freq. (eq. 3).

4.1 Resource Models

Models for battery discharge profiles and associate lifetimes have received considerable interest. As surveyed in [8], models to analyze battery discharge profiles may be circuit, electrochemical, stochastic or analytical in nature. For constant loads, the battery lifetime L can be derived from the battery capacity C and the discharge current I as $L = \frac{C}{I}$. However, modeling the discharge capacity of lithium-ion or lead acid batteries is complicated by the non-linear discharge profiles. An improvement over this is the Peukert's law [9], that relates the battery lifetime and current drawn, with parameters a and b.

$$L = \frac{a}{I^b} \tag{1}$$

Typically the value of a is close to battery capacity and b is a value in the range $[1.2, 1.7]$. When using a non-uniform discharge, the average discharging values may be employed over time periods t:

$$L = \frac{a}{\left[\frac{\sum_{k=0}^{N} I_k(t_k - t_{k-1})}{L}\right]^b} \tag{2}$$

We must note that only the average discharge current is taken into account in this case, which has been analyzed in [8] to have an error profile of 8–12%.

As we deal with a network of sensors–actuators that can communicate among themselves (infrastructure less) as well as with smart gateways/cloud (infrastructure based), appropriate communication protocols must be taken into account. The well known log-distance fading model [10] relates the received power P_r in terms of the transmitted power P_t and a path loss coefficient. For indoor environments, a more realistic path loss model has been proposed in [11], that incorporates the effect of obstacles, indoor multi-path effects as well as varying transmission frequencies. The path-loss in *dBm* is given by:

$$P_t - P_r = 20 \cdot \log_{10}(f) + K \cdot \log_{10}(d) + N_w \cdot L_w - 20 \tag{3}$$

where f is the transmit frequency in MHz, the attenuation factor K is derived based on the transmit frequency ($\approx$ 19 for 2.4 GHz) and the number N_w and loss L_w due to obstacles are also incorporated. The loss due to obstacles from measurements can range from 1 dB for glass, 7 dB for brick to 25 dB for concrete structures. When dealing with varied transmission solutions such as WiFi, Zigbee, Bluetooth and UWB [10], incorporation of the transmit frequencies can affect the measured path loss [11]. Simulated outputs using this model is presented in Figure 4. Incorporating the effect of such varying communication path loss and battery degradation into an optimization framework is covered next.

Parameter	Description
$\mathbf{C}$	Total computational task to be fulfilled
$\mathbf{L}$	Latency constraint to be fulfilled
i	ID of the edge/fog device with $i \in [1, N]$
r_i	Computation task allocated to the i^{th} node
P_i^{cap}	Total battery capacity of the i^{th} node
P_i^c	Battery consumption of the i^{th} node for a computation unit c
P_i^t	Battery consumption of the i^{th} node for a transmit unit t
f	Communication transmit frequency
d_{ij}	Distance between nodes i and j
t_{ij}	Throughput for transmit link between nodes i,j
cc_i	Number of CPU cores on node i
h_i	CPU frequency (GHz) on node i

Table 1: Parameters used for Optimization.

4.2 Optimization Formulation

In order to reduce battery consumption in relation to the computational offloading among a network of devices, we formulate an optimization problem. Each node is characterized by its location (`x,y` co-ordinates, orientation), current drawn during computation/communication, total battery capacity, number of computation CPU cores and CPU frequency. For the formulation, we assume that device with ID $i = 1$ intends to offload some of the internal computation (Figure 2). Parameters used for the optimization formulation are presented in Table 2. We consider the following:

1. *Minimize Battery Consumption*: Battery consumption for a computational task remains a key constraint that requires optimization. We formulate it as a linear programming problem with the battery consumed as a result of computation P_i^c (eq. 2) and two-way request–response communication overhead $2 \cdot P_1^t\{f, d_{1i}\}$ (eq. 3) for offloaded data (device $i \neq 1$). The values of P_i^c and P_i^t are obtained from the hardware specifications of the devices. The total computation task $\mathbf{C}$ is allocated to participating devices, with a maximal battery capacity of P_i^{cap}. The optimization (device $i = 1$ is the off-loader):

$$\begin{aligned} \min \quad & P_1^c \cdot r_1 + \sum_{i=2}^{N} (P_i^c + 2 \cdot P_1^t\{f, d_{1i}\}) \cdot r_i \\ \text{s.t.} \quad & \sum_{i=1}^{N} r_i = \mathbf{C} \\ & P_i^c \cdot r_i \leq P_i^{cap}, \quad \forall i \in N \\ & r_i \geq 0, \qquad\quad \forall i \in N \end{aligned} \tag{4}$$

   In case there is a battery re-charge between subsequent task executions, P_i^{cap} remains constant. Where there are no battery re-charges between runs, a new constraint is needed:

$$P_i'^{cap} = P_i^{cap} - P_i^c \cdot r_i \tag{5}$$

   This accounts for the power consumed due to the optimal allocation during the previous execution. For the most deteriorated battery $P_i^{cap} \to 0$ the earliest.
2. *Minimize Latency*: In fog deployments, latency is a predominant constraint. As each device has a specific number of CPU cores cc_i with corresponding operating frequency h_i, the time to complete a computation may be incorporated. For instance, for an embarrassingly parallel computation, a single core CPU running at 1.2 GHz would perform 8 times slower than a quad-core 2.4 GHz CPU. The optimization, incorporating the transmission throughput t_{ij} for data transmission (device $i = 1$ is the off-loader):

$$\begin{aligned} \min \quad & \left(\frac{1}{cc_1 \cdot h_1}\right) \cdot r_1 + \sum_{i=2}^{N} \left(\frac{1}{cc_i \cdot h_i} + \frac{1}{t_{1i}}\right) \cdot r_i \\ \text{s.t.} \quad & \sum_{i=1}^{N} r_i = \mathbf{C} \\ & r_i \geq 0, \quad \forall i \in N \end{aligned} \tag{6}$$

3. *Latency and Battery Constraints*: A combined model that takes into account both battery and latency constraints can also be formulated. While the primary cost function to be minimized captures the battery loss, the entire task must be completed within the latency constraint $\mathbf{L}$ (device $i = 1$ is the off-loader):

$$\begin{aligned} \min \quad & P_1^c \cdot r_1 + \sum_{i=2}^{N} (P_i^c + 2 \cdot P_1^t\{f, d_{1i}\}) \cdot r_i \\ \text{s.t.} \quad & \sum_{i=1}^{N} r_i = \mathbf{C} \\ & \left(\frac{1}{cc_1 \cdot h_1}\right) \cdot r_1 + \sum_{i=2}^{N} \left(\frac{1}{cc_i \cdot h_i} + \frac{1}{t_{1i}}\right) \cdot r_i \leq \mathbf{L} \\ & P_i^c \cdot r_i \leq P_i^{cap}, \quad \forall i \in N \\ & r_i \geq 0, \qquad\quad \forall i \in N \end{aligned} \tag{7}$$

5. ANALYSIS RESULTS

The first subsection concerns the various optimal deployments as a result of the formulation in Section 4. The second subsection then applies this deployment on the networked robotics case study presented in Section 3, to compare improvements in energy efficiency and latency.

5.1 Optimal Deployment

The devices are specified with coordinates, battery, computational capacities and latency/energy depletion with each compute and communication unit. The path loss model developed in eq. 3 is used in conjunction with WiFi 802.11n transmission (2.40 GHz, 1 Mb/s) between the nodes. The optimization formulations from Section 4 are solved in `Scilab` using the *Karmarkar* linear optimization solver. Three cases are studied:

1. Computation deployment assuming no battery re-charge, homogeneous device capacities – the results are presented in Figure 5 for increasing computation loads (specified in KBs), on a set of four devices. We notice approximately proportional offloading to peers, that may be dependent on the communication path loss power constraints. For low computational loads, the entire task is performed locally, without any offloading.
2. Computation deployment assuming no battery re-charge, heterogeneous device capacities: The results are presented in Figure 6, with the higher battery/computational capacity *Device 4* included. We notice that initially, most of the computation is offloaded to the higher capacity device; once battery depletion of *Device 4* becomes significant, other devices are utilized. Such a

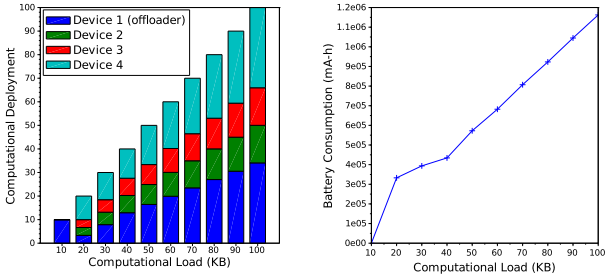

Figure 5: Case 1: No Battery Re-charge, Homogeneous Capacity.

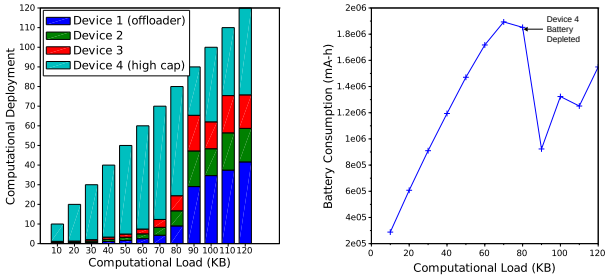

Figure 6: Case 2: No Battery Re-charge, Heterogeneous Capacity.

situation would be seen in the Fog Computing scenario, where static smart gateways/routers with higher computational capacities may be utilized for efficient offloading.

3. Computation deployment assuming no battery re-charge, latency constraint, heterogeneous device capacities: The results in Figure 7 produces a revised deployment compared to Figure 5. Devices closer to the off-loader are chosen repeatedly in order to meet the additional latency constraint for larger computation tasks.

Optimized deployments, applied to the networked robotic environments, are seen in the next section.

5.2 ROS/Gazebo Simulation

The networked robotic deployment described in Section 3 is developed in ROS-Indigo with the default Gazebo 2.2.2 on a Linux Machine with quad core i5-6200U CPU (2.30 GHz) and 4.00 GB RAM. A set of four robots collect sensor data (location and orientation coordinates) ranging from $10,000$–$200,000$ entries (sensor data of type `double` with size ranges 160 KB–3.2 MB). A computationally expensive *BubbleSort* operation is then performed over the collected data either individually or using our architecture (Figure 2). As all the robots were simulated on a single machine, we use the Linux `taskset` command to pin one CPU core/robot.

In order to generate realistic hardware capacity, the Husky datasheet in Table 2 is used. The battery model in eq. 2 is used to estimate average current drawn under standby and nominal usage (constant voltage of 24 V):

$$\begin{aligned} I_{\text{standby}} &= \left(\tfrac{20\ \text{Ah}}{8\ \text{h}}\right)^{\frac{1}{1.3}} = 2.02\ \ \text{Amperes} \\ I_{\text{usage}} &= \left(\tfrac{20\ \text{Ah}}{3\ \text{h}}\right)^{\frac{1}{1.3}} = 4.30\ \ \text{Amperes} \end{aligned} \tag{8}$$

As ROS simulates only the application level message passing, the additional delays introduced by WiFi transmission

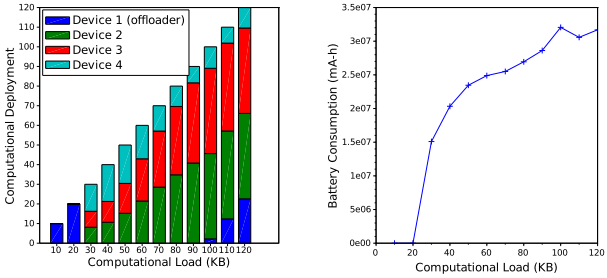

Figure 7: Case 3: No Battery Re-charge, Latency Constraint, Homogeneous Capacity.

Metric	Value
Physical Dimensions	$39 \times 26.4 \times 14.6$ inches, 50 kg
Maximum Speed	1.0 m/s
Battery Specs.	Sealed Lead Acid, 20 Ah, 24 V
Runtime (standby)	8 hours
Runtime (nominal use)	3 hours

Table 2: Husky Robot – Technical Specifications.

are incorporated as overheads. The path loss model developed in eq. 3 is used in conjunction with WiFi 802.11n transmission (2.40 GHz, 1 Mb/s) between the nodes. Note that channel collisions and interference between devices have not been considered in our model.

The results of applying our optimized deployment model is shown in Figure 8, with optimal division and battery consumption plotted against the computational load size (KBs). We compare the *individual* (computation done locally or offloaded to only one peer) opposed to *collaborative* (applying the optimized deployment from Section 4.2). In all such cases, offloading data to the cloud is not an option due to network connectivity and time delay constraints. As expected, due to the higher computational capacity available, for a *BubbleSort* computation done on $200,000$ entries (≈ 3.2 MB), the improvement in latency was close to 77.8%, inspite of additional data transmission delays over WiFi between nodes. While sorting and searching algorithms are easily parallelized, for more complex algorithms, the speedup due to parallelization may be obtained using Amdahl's law. As the battery of each node is active for smaller durations, the battery saving also improved in the collaborative model by 54%. Note that we have included the battery depleted in all the collaborating devices for battery drain.

Further analysis of battery drained with computational size is shown in Figure 9. For larger tasks, we notice that devices performing computations on its own or offloading to a single peer, would be depleted within 23 executions (it is important not to deplete a peer's battery completely as well). However, when the collaborative model with optimized deployment (battery usage minimized) was applied, the same device could be deployed for 93 executions. As most IoT environments are constrained with respect to energy, such optimizations over battery usage should prove valuable.

6. RELATED WORK

While the cloud computing paradigm [2] has received considerable industrial acceptance due to advantages such as just in time computing, elastic scaling and energy efficient data-centers, additional challenges posed by the Internet of Things (IoT) [1] require enhancements. Bonomi et al. [3] [4] have proposed architectures and toolkits around the fog

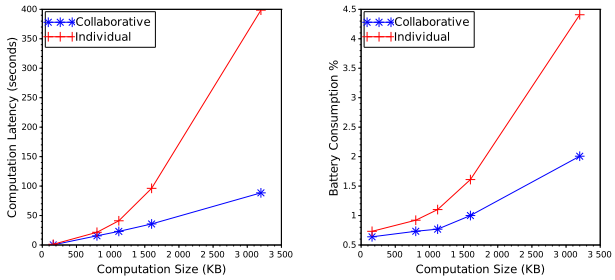

Figure 8: Computations in ROS/Gazebo.

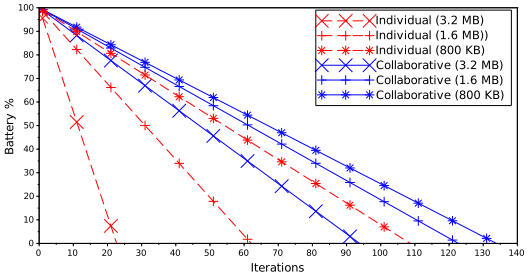

Figure 9: Device Battery Degradation per Iteration.

computing paradigm, where centralized cloud computation may be decentralized using intelligent switches, routers and gateways. Another related work in this space include the concept of cloudlets [12], where localized computing nodes are incorporated to tackle the last mile communication problem with centralized clouds. Hand-off, virtual machine migration and connectivity issues that result due to mobile nodes are an active area of cloudlets research.

In [13], the motivation for moving away from cloud centric architectures are analyzed: scalability (millions of devices), high bandwidth requirements and intermittent connectivity. It motivates the latency and security constraints of the cloud that may be overcome using fog/edge computing devices. In [14] a simulator is developed to model fog environments and measure the impact of latency, energy consumption and network bandwidth. Comparison of edge enabled vs. cloud only computations are studied through the simulator.

Studies in heterogeneous grids have proposed dividing embarrassingly parallel computations across a network of devices. Work in this area includes the Condor [6] and Boinc [15] based architectures. A large computational task may be divided based on unused computational resources available in the network. While the focus of this paper does not go into the implementation details of such offloading, idle resources on the network can be probed, collated and used for a computational task using such models.

In [7], the M2M communication amongst a network of cloud connected robots are studied using proxy, peer and clone based topologies. A gossip based mechanism is proposed for robots to communicate and offload computations. The communication overheads involved with such offloading have recently been studied with Geng et al. [16], where energy overheads in offloading over 3G and LTE networks are studied.

7. CONCLUSIONS

Proliferation of the Internet of Things (IoT) introduces new limitations in traditional sense-compute-actuate paradigms such as battery capacity, communication overheads and limited computational capacities. In this paper, we make use of the *Fog Computing* paradigm to tackle some of these challenges in an efficient manner. By formulating the resource trade-offs in terms of energy and latency, we demonstrate optimal offloading of computations across a network of devices. The results are analyzed over a network of robotic sensor–actuators developed in ROS/Gazebo, that produce 77.8% improvement in latency and 54% improvement in battery consumptions at higher loads. Such a deployment framework can prove useful in multiple scenarios involving fog based networking for IoT devices.

Future work involves using more complex algorithms such as robotic SLAM as well as distributed optimization among robotic nodes.

8. REFERENCES

[1] Samuel Greengard, "The Internet of Things", *MIT*, 2015.

[2] Rajkumar Buyya, James Broberg and Andrzej M. Goscinski, "Cloud Computing: Principles and Paradigms", *Wiley*, 2011.

[3] Flavio Bonomi, Rodolfo Milito, Preethi Natarajan and Jiang Zhu, "Fog Computing: A Platform for Internet of Things and Analytics", *Big Data and Internet of Things: A Roadmap for Smart Environments*, pp. 169–186, 2014.

[4] Flavio Bonomi, Rodolfo Milito, Jiang Zhu and Sateesh Addepalli, "Fog Computing and Its Role in the Internet of Things", *Mobile Cloud Computing Wksp.*, New York, 2012.

[5] Tom H. Luan, Longxiang Gao, Zhi Li, Yang Xiang and Limin Sun, "Fog Computing: Focusing on Mobile Users at the Edge", *Deakin University*, Technical Report, 2016.

[6] Douglas Thain, Todd Tannenbaum and Miron Livny, "Distributed Computing in Practice: The Condor Experience", *Concurrency and Computation: Practice and Experience*, vol. 17, no. 2–4, pp. 323–356, 2005.

[7] G. Hu, W. P. Tay and Y. Wen, "Cloud robotics: architecture, challenges and applications", *IEEE Network*, vol. 26, no. 3, pp. 21–28, 2012.

[8] Ravishankar Rao, Sarma Vrudhula and Daler N. Rakhmatov,"Battery Modeling for Energy-Aware System Design", *IEEE Computer*, vol. 36, no. 12, 2003.

[9] D. Linden and T. Reddy, "Handbook of Batteries", *McGraw-Hill*, 3rd ed., 2001.

[10] Theodore S. Rappaport, "Wireless Communications: Principles and Practice", *Prentice Hall*, 2nd ed., 2001.

[11] Hemant Kumar Rath, T Sumanth, Bighnaraj Panigrahi and Anantha Simha, "Realistic Indoor Pathloss Modeling for Regular WiFi Operations in India", TCS Research, 2016 (under review).

[12] Mahadev Satyanarayanan, Paramvir Bahl, Ramon Caceres and Nigel Davies, "The Case for VM-Based Cloudlets in Mobile Computing", *IEEE Pervasive Computing*, vol. 8, no. 4, 2009.

[13] Angelo Corsaro, "The Cloudy, Foggy and Misty Internet of Things: Toward Fluid IoT Architectures", Prismtech, 2016.

[14] Harshit Gupta, Amir Vahid Dastjerdi, Soumya K. Ghosh and Rajkumar Buyya, "iFogSim: A Toolkit for Simulation of IoT, Edge and Fog Computing Environments", The University of Melbourne, 2016.

[15] Attila Marosi, Jozsef Kovacs and Peter Kacsuk, "Towards a volunteer cloud system", *Future Generation Computer Systems*, vol. 29, pp. 1442–1451, 2014.

[16] Y. Geng, W. Hu, Y. Yang, W. Gao and G. Cao, "Energy-Efficient Computation Offloading in Cellular Networks", *IEEE 23rd Intl. Conf. on Network Protocols (ICNP)*, pp. 145–155, 2015.