Towards Resource Sharing in Mobile Device Clouds: Power Balancing Across Mobile Devices

Abderrahmen Mtibaa
Texas A&M
amtibaa@tamu.edu

Afnan Fahim, Khaled A. Harras
Carnegie Mellon University
{afahim, kharras}@qatar.cmu.edu

Mostafa H. Ammar
Georgia Institute of Technology
ammar@cc.gatech.edu

ABSTRACT

Despite the increased capabilities of mobile devices, mobile application resource requirements can often transcend what can be accomplished on a single device. This has been addressed through several proposals for efficient computation offloading from mobile devices to remote cloud resources or closely located computing resources known as cloudlets. In this paper we consider an environment in which computational offloading is performed *among a set of mobile devices*. We call this environment a *Mobile Device Cloud (MDC)*. We are interested in MDCs where nodes are *highly collaborative*. We develop computational offloading schemes that *maximize the lifetime* of the ensemble of mobile devices where we consider the network to be *alive* as long as no device has depleted its battery. As a secondary contribution in this work, we develop and use an experimentation platform that allows us to evaluate a range of computational models and profiles derived from a realistic testbed. We use this platform as a first step in an evaluation exercise that demonstrates the effectiveness of our computation offloading algorithms in extending the lifetime of an MDC.

Categories and Subject Descriptors

C.2.4 [**Computer**]: Communication Networks

General Terms

Algorithms, Design, Experimentation, Performance

Keywords

Resource sharing, Mobile computing, Opportunistic computing, Energy saving

1. INTRODUCTION

Mobile devices (e.g., smartphones, tablets) are increasingly capable devices with significant computational power. The demand for sophisticated mobile applications has been fed by this increased capability. As a result, users take for granted the ability of their devices to perform complicated tasks such as image or audio or video processing, reality augmentation, collaborative processing and decision making.

There continue to be two main roadblocks to unleashing the full computational power of mobile devices. The first is the continued power constraints in modern devices where battery technology advances have not kept pace with the advances in processing capability. Second, is that users demand applications whose computing requirements often transcend what can be accomplished on a single device. To address these concerns there has been a body of work dealing with techniques for efficient computation offloading from mobile devices to remote cloud resources [7, 6] or closely located computing resources known as cloudlets [13].

In our work, we consider environments in which computational offloading is performed *among a set of mobile devices* forming what we call a *Mobile Device Cloud (MDC)*. Such an offloading context was considered in the Serendipity system [14]. Our work is distinguished from [14] in that we consider a *highly collaborative* context where the goal of computational offloading is to *maximize the lifetime* of the MDC. This context arises in several scenarios. The first is where the mobile devices all belong to the same user, for example, a user's smartphone, tablet and laptop, or where the devices belong to the same household. The second context is where the devices are carried by a group of people on a single collaborative mission. This can happen, for example, in military or disaster relief scenarios. In such scenarios, the incentive to collaborate on computational tasks is not an issue and the communal goal of prolonging the lifetime of the collection of devices makes sense. This incentive is further amplified if a connection to a cloud is costly (in terms of money or power due to large RTT [7]), unreliable, or simply unavailable.

The main question we address in this paper is given 1) the initial state of power availability in a collection of collaborative mobile devices, and 2) a set of computational tasks with known or estimated power consumption profiles on the mobile devices, what is the best approach to schedule the computation among the set of devices so as to maximize their lifetime. Our work will consider a spectrum of connectivity environments ranging from always connected (single or multiple hops) to intermittently connected. We will focus on the always connected context for this paper and relegate consideration of other connectivity models to future work.

Our work can be viewed as providing mechanisms for the balancing of power consumption due to computational load

MCC'13, August 12, 2013, Hong Kong, China.

across nodes of a mobile device cloud (MDC). We envision this to be part of a general effort for balancing consumption of all resources across an MDC, such as storage capacity, communication bandwidth, and computational power. Ultimately, we believe that effective resource management across the elements of a highly-collaborative MDC can significantly increase their usability as powerful computing platforms.

Our efforts and the previous effort in [14] have highlighted the need for a general experimentation platform to provide meaningful evaluation of mobile device cloud (MDC) systems. Such experimentation platforms need to enable general evaluation with a range of computational models and profiles and need to be informed with realistic parameters derived from real systems. As a secondary contribution in this work, we develop and use such an experimentation platform in our evaluation.

The rest of this paper is organized as follows. Section 2 briefly discusses work related to our research. In Section 3 we describe the MDC context and present our algorithms for computational offloading that aim to maximize the network lifetime. Section 4 describes our MDC experimental platform. Section 5 presents the results from an experimental evaluation of our algorithms. Section 6 concludes the paper and discusses our future work agenda in this area.

2. RELATED WORK

With the rise in mobile application demand on computational resources, various solutions for computation offloading to more powerful surrogate machines, known as cyber foraging, have been proposed [8]. Recent solutions include CloneCloud [6] and MAUI [7]. CloneCloud presents an algorithm, based on static analysis and dynamic profiling information of any given task, for deciding whether to execute this task locally or on a remote cloud. MAUI relies on developer effort to convert mobile applications in a managed code environment to better support fine-grained real-time offloading decision making; it also considers the possibility of offloading to different types of high-end infrastructures depending on their RTT in order to conserve energy. The impact of large RTT's on power consumption when offloading computation is further examined and utilized as an incentive for bringing resource-rich computational infrastructure, known as Cloudlets [13] closer to mobile devices.

In our work, we consider computational offloading *among a set of mobile devices*. Such a system has been considered in [14] where the main concern was how to schedule and allocate computational subtasks from an initiator to other intermittently-connected mobile devices. The main goal was to speed up computation through parallelism and to, if possible, reduce the initiator's power consumption. In this work, we consider the same context in which mobile devices can offload computational subtasks to other mobile devices in a *Mobile Device Cloud (MDC)* which may or may not be intermittently connected. Our work here is distinguished from that in [14] in that we consider a *highly collaborative* context where the goal of computational offloading is to *maximize the lifetime* of the ensemble of devices.

Numerous energy saving solutions have been proposed by the sensor networks community based on topology control, route choice, sleep/wakeup protocols, low MAC protocol duty cycles, data reduction, different data sampling techniques, and sensor clustering solutions [2, 11, 15, 12, 3]. Similar to this body of work, we consider a network to be

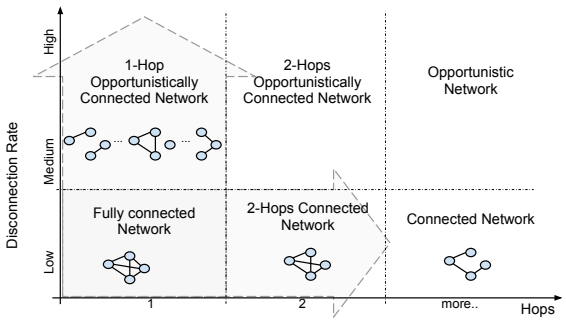


Figure 1: MDC Communication Contexts

alive as long as no device has depleted its battery. However, the characteristics and typical conditions in sensor networks differ significantly from the contexts considered in this paper since they typically assume low mobility, high node homogeneity, constrained nodes that are not often recharged, and topologies that overload nodes closer to data collecting sinks.

3. MAXIMIZING THE LIFETIME OF MOBILE DEVICE CLOUDS (MDC)

3.1 MDC Communication Context

Based on the scenarios presented in the introduction, we believe that various mobile devices nowadays, and more so in the future, can be easily grouped in a single ensemble of devices that can act as a mobile device cloud (MDC). This ensemble can be formed in various ways including manual assignment (similar to adding all iOS devices to one's iTunes, or android devices with google), intelligent device contact profiling, or via social profiles present on these devices (e.g. devices of members in the same household might be assigned to a single MDC). This discovery and MDC clustering process is addressed in various ways in the research community [1] and therefore, we assume that such mechanisms exist.

Regardless of the resource being shared between members of an MDC, the network topology of that MDC has great impact on the performance of offloading or resource sharing algorithms. We believe that the two topological factors that mostly impact any proposed solution are the number of hops and the disconnection rate due to node mobility between any pair of MDC nodes. The communication contexts created as a result of these two factors are shown in Fig. 1. Our work will consider the spectrum of connectivity environments starting from the most basic, a fully connected MDC with low disconnections (e.g. multiple devices carried by a single owner), to more complex scenarios induced by gradually considering two hop connected MDC's with low disconnections as well as one hop opportunistically connected MDCs. We relegate consideration of other portions of the connectivity contexts to future work.

3.2 Network Model & Assumptions

We consider a network of n mobile devices. Each device u has limited energy E_t^u, and available computation C_t^u capabilities at any given time t. These nodes form a connected or intermittently connected graph G.

Each device node u performs a set of tasks $T_k^u, k = 1..j$. We assign to each task a time-to-live deadline, denoted by

TTL_{T_k}. T_k also consists of two main components; computation C_{T_k} and data transfer D_{T_k}.

We assume in this paper that all tasks can run simultaneously and there are no dependencies between any two tasks.

$E^{u,v}_{T_k}$ denotes the expected energy requirement for a task T^u_k to run on a device v. Such energy is a function of the required time to compute the task at node v denoted by $C^{u,v}_{T_k}$ and the required time to transfer the task data input and output between the source node u and node v denoted by $D^{u,v}_{T_k}$.

$$E^{i,u}_{T_k} = f(C^{i,u}_{T_k}, D^{i,u}_{T_k})$$

Where f is a function that determines the expected energy consumed based on (i) computation and (ii) data transfer requirements. In this paper we adopt an experimental approach that updates a table of estimated energy and response times for each data and computation pair (*i.e.,* each task). Different energy functions f can also be considered in order to estimate the energy consumption in real time.

3.3 Power Balancing Algorithm

In wireless opportunistic networks, nodes have limited capacity batteries. A node fails when its battery drains, which causes considerable delay degradation in the network. Balancing available power is then particularly important to prolong the *lifetime* of the network. We believe that in a mobile device cloud all nodes in the network are important and a node failure is equivalent to network failure. We therefore define the *lifetime* of the network as the time for the first node to die as in [5, 9].

Our goal in this paper is to prolong the lifetime of the network by balancing energy consumption among MDC nodes. We propose power availability balancing solutions for each of the communication contexts previously defined in order to highlight the potential gain in network lifetime.

Alg. 1 describes the approach to schedule the computation among the set of devices so as to maximize their lifetime given the initial state of power availability in a collection of collaborative mobile devices, and a given task $T_k(C_{T_k}, D_{T_k}, TTL_{T_k})$ for a node u characterized by its available remaining energy E^u_t ,and available computation capabilities C^u_t at a given time t. It provides a function that returns G all members belonging to the same family of node u ($GETMEMBERS(u)$), and the communication context used to connect such devices. In this paper we simply assume that nodes manually select their family members (*e.g.,* all MAC devices registered to iTune). Alg. 1 considers a spectrum of connectivity environments ranging from always connected; (i) single, or (ii) multiple hops to (iii) intermittently connected. For each connectivity environment, it estimates data communication ($Offl(v)$ represents the estimated time to offload the data to node v, and $Resp(v,u)$ the time to send back the results from node u to v.) and task computation time ($Comp()$) as well as the power consumption of the given task on all mobile devices including itself and offloads to the nodes v such that it balances the remaining power among all nodes in G.

3.3.1 Single Hop MDC

A node u, belonging to a single hop MDC cluster G, receives a list of neighbors that form a fully connected network of mobile devices. The $GetMembers(u)$ is responsible of selecting theses nodes forming the MDC based on social or contextual profiling. It also estimates the lifetime of each edge in the graph as an expected contact duration between two neighboring devices.

Node u, upon generating a given task T_k, it offloads it to a neighbor node v if and only if the expected response time of migrating the task to v is within the task deadline (TTL_{T_k}) and the estimated remaining energy is maximized as shown in Eq. 1.

$$\max_{v \in G}\{(E^u_t - E_{comm}(u,v)) + (E^v_t - E_{comp}(v) - E_{comm}(v,u))\} \quad (1)$$

where $E_{comm}(u,v)$ is the total amount of energy required by node u to send D_{T_k} to another node v, and receive the result data back from the selected node v. $E_{comm}(v,u)$ represents node v's estimated communication energy for receiving D_{T_k} from node u and the energy required to send back the results. $E_{comp}(v)$ is the estimated amount of energy required by node v to compute T_k.

After selecting node v that gives the best energy balance, if $v = u$ the node executes the task locally and updates its remaining energy. Otherwise, it offloads the task and waits for its completion to receive the results back from node v.

3.3.2 Two-Hop MDC

In the case of a two hop MDC cluster G, node u receives the topology of the network and computes all shortest paths to all nodes in the graph. For each of its tasks T_k, node u computes all expected energy consumed if the task has to be offloaded to a every device v in the graph G. This expected energy is estimated differently if the task is offloaded throughout a single hop or two hop paths. Knowing the shortest distance to each node v, Node u verifies (i) Eq. 1 if v is directly connected to u or (ii) Eq. 2 if there exists another node w that connects u to v.

$$\max_{v \in G}\{(E^u_t - E_{comm}(u,w)) + (E^w_t - E_{comm}(u,w,v)) + (E^v_t - E_{comp}(v) - E_{comm}(v,w))\} \quad (2)$$

where w is an intermediate node connecting nodes u and v, and $E_{comm}(u,w,v)$ is an estimated communication energy consumption of node w to receive data from u, forward it to v, receive back results from v, and forward it to u.

3.3.3 Opportunistic MDC

In the case of opportunistic (intermittently connected MDC), node u verifies whether or not contacts with its neighbors are long enough to carry the task data, as well as the existence of a communication channel to receive back the results. All this should happen within the task deadline. We assume in this paper, that node u can predict the contact duration and frequency connecting him to its MDC nodes. This also can be estimated via social or contextual profiling or historical contacts with other node.

Once the availability of a communication channel to a subset of nodes in node u's MDC graph G is ascertained, it offloads the task to the node that maximizes Eq. 1.

4. THE MDC EXPERIMENTATION PLATFORM

Researchers in mobile cloud computing resort to implementing or migrating representative resource heavy applications on mobile devices over which they evaluate new architectures, task scheduling algorithms, or different offloading

Algorithm 1 Power balancing Alg. (node $u(E_t^u, C_t^u)$)

Require: $Retrieve(DeviceCapabilities)$
1: **function** GETMEMBERS(u) ▷ returns MDC of u
2: **while** $\exists(T_k(C_{T_k}, D_{T_k}, TTL_{T_k}) \in buffer(u))$ **do**
3: $G \leftarrow$GETMEMBERS(u)
4: $\forall v \in G, E^v \leftarrow CurrentEnergy(v)$
5: **switch** G **do**
6: **case** $1hopMDC$
7: $\forall v \in G$,
8: **if** $Offl(v) + Comp(v) + Resp(v,u) \leq TTL_{T_k}$ **then**
9: $offoadTo(v|Eq.1)$
10: $Update(E^u, E^v)$
11: **case** $2hopMDC$
12: $\forall v \in G$,
13: **if** $\pi(u,v) \subset G$, **then** ▷ $\pi()$ a shortest path in G
14: **if** $Offl(v) + Comp(v) + Resp(v,u) \leq TTL_{T_k}$ **then**
15: $offoadTo(v|Eq.2)$
16: $Update(E^u, E^v)$
17: **else**
18: $\exists w, \pi(u,w,v) \subset G$,
19: **if** $Offl(w) + Offl(v) + Comp(v) + Resp(v,w) + Resp(w,u) \leq TTL_{T_k}$ **then**
20: $offoadTo(v|Eq.2)$
21: $Update(E^u, E^w, E^v)$
22: **case** $OppMDC$
23: **for** $(v \in G)$ AND $(Wait(u,v) + Offl(v) + Comp(v) + Wait(v,u) + Resp(v,u) \leq TTL_{T_k})$ **do**
24: $offoadTo(v|Eq.1)$
25: $Update(E^u, E^v)$

techniques. Since appropriate, flexible, and open source mobile applications are not easily accessible, this approach is time consuming and takes the focus away from the main research contributions. Even with the effort exerted in integrating research contributions with representative applications, results are coarse grained, potentially application dependent, and take away the ability of evaluating future applications that might not exist yet.

Based on this observation, we believe there is a need for a generic flexible platform that can be utilized by researchers to freely test mobile cloud computing resource sharing and offloading solutions. This tool should decouple two main components that characterize any mobile application: the amount of data as well as the computational load that any task or job will require. These two components of the application should also be easily broken down into distributable sub-tasks that researchers can control in real-time. Similar to simulations, this flexibility in the generic platform allows researchers to test their solutions over a fine-grained range of parameters that can represent a wider spectrum of current and future applications.

We introduce our mobile data cloud (MDC) experimental platform for mobile cloud computing research as shown in Fig. 2. We implement the MDC platform as an android application with a set of APIs for mobile-to-mobile task offloading. This platform allows users to generate tasks with different computational loads (measured in total floating point operations and denoted in MFLOP) and relevant data input (measured and denoted in MB). It also provides APIs that enable building more specialized applications that can offload sub-tasks, set by the user, using various wireless technologies, such as WiFi, Bluetooth, or WiFi Direct. The user can select the number of connected devices from a pool of devices within its proximity, as well as the amount of data and computational load to be offloaded to each connected device. When the user executes a task generation and offloading scenario by pressing the send button, the original

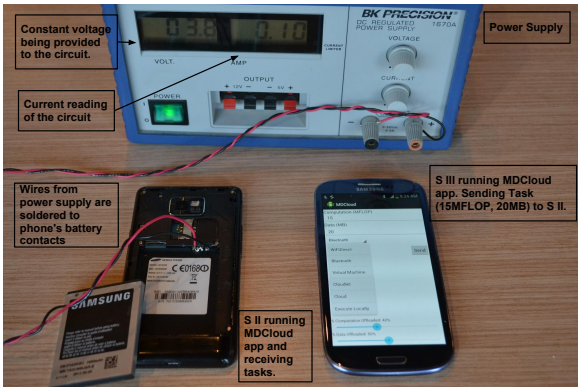


Figure 2: MDC Experimentation Platform: measuring energy consumption on an S2 upon receiving a task (15MFLOP, 20MB) from S3 device

task is, therefore, fragmented and the selected percentages will be forwarded to remote devices while the remaining subtasks will run locally. The MDC platform enables the application to log the total response time for each task (*i.e.,* the time when the task was initiated to the time when the results are sent back to the initiating user). It also logs the task computational completion time for every device as well as the data transfer time separately.

In order for the MDC experimental platform to be complete, real-time power measurements are needed for various states of an application. We therefore set-up an energy measurement circuit as shown in Fig. 2 in order to measure the computational and communication energy consumption. We remove the battery from the phone, solder a DC power supply to phone's power input as shown in the figure, and supply a constant voltage that matches that of the battery. We measure the electric current drawn as a result of each event, such as sending or receiving data (using Bluetooth, WiFi, or WiFi direct) as well as a given computational load, and calculate the power consumed for each event. We run these events independently for the duration of one minute to smooth out any fluctuations that occur as a result of other OS-related tasks. Once we have the power for each event computed, we finally measure the duration taken for any event in our real experiments and calculate the resulting total energy consumed in Joules.

We use this platform to generate power usage profiles for various computational and data transmission demands. These profiles are in turn used as input to system simulations as described in the next section.

5. EXPERIMENTAL EVALUATION

5.1 Evaluation Methodology

We first run a set of preliminary experiments using the MDC platform previously described. We perform many experiments while making Bluetooth transfers between a Samsung S2 and a Samsung S3 devices. We also run many experiments independently consisting of sending, receiving and computing tasks with different data sizes and computation complexities. We construct a table that logs response time and consumed energy for each task (*i.e.,* energy consumed upon sending $Es_{D_{T_k}^{S2,S3}}$, energy consumed upon re-

MB [$Comm\_time$]	MFLOP [$Comp\_time$]				
	0 [0s]	15 [13.43s]	30 [28.441s]	45 [44.19s]	60 [57.59s]
0 [0s]	[0 ;0 ; 0]	[0 ; 4.39 ; 4.29]	[0 ; 9.06 ; 8.98]	[0 ; 13.392 ; 12.85]	[0 ; 18.38 ; 17.76]
5 [24.85s]	[9.19 ; 5.96 ; 0]	[9.62 ; 6.24 ; 4.29]	[9.18 ; 5.95 ; 9.1]	[9.19 ; 5.95 ; 14.14]	[9.17 ; 5.95 ; 18.43]
10 [49.354s]	[18.26 ; 11.84 ; 0]	[18.24 ; 11.83 ; 4.38]	[18.26 ; 11.84 ; 9.36]	[18.44 ; 11.96 ; 13.56]	[18.28 ; 11.85 ; 18.77]
20 [98.583s]	[36.59 ; 23.73 ; 0]	[36.59 ; 23.73 ; 4.39]	[36.56 ; 23.71 ; 8.82]	[36.43 ; 23.63 ; 13]	[36.4 ; 23.61 ; 18.56]

Table 1: Experimental results for Bluetooth offloading from S2 to S3: [send ; recv ; comp] energies in J

ceiving $Er_{D_{T_k}^{S2,S3}}$, and energy consumed upon computing a task $E_{T_k}^{S2,S3}$) as shown in Tab. 1. We vary computation from 0 MFLOP to 60MFLOP[1]. We also vary the input data sizes from 0 to 20MB (*e.g.,* compressed video file). Note that Tab. 1 consists of results of a Bluetooth transfer from S2 to S3. We also run experiments for all possible transfer combinations. Results for other combinations are not shown in the paper to avoid redundancy.

Next, we consider a network consisting of 5 mobile devices; 2 Samsung S2 and 3 Samsung S3 running the MDC application. We emulate an MDC cluster where nodes generate tasks independently. Tasks are generated following a Poisson arrival process of rate λ. The goal is to fairly distribute the computation and data load in order to achieve a fair energy distribution within a deadline δt. We assign initial battery levels randomly between the following two initial levels 25% and 100%.

We, then, measure three main metrics; (i) the average remaining energy in the networks, (ii) the fairness index, and (iii) the lifetime of the network. We use Jain's fairness metric [10], denoted by F, to evaluate the available power balancing among all nodes in the network. The best case is reached when F is close to 1 (fair allocation), and the worst case is given by $1/n$.

$$F = \frac{(\sum_1^n E_{\delta t}^i)^2}{n \cdot \sum_1^n (E_{\delta t}^i)^2}$$

5.2 A Single Hop MDC Cluster

Fig. 3-(a) plots both the Jain fairness index and the average remaining energy at each node over time. We show that, while the average remaining energy in the network is dropping almost linearly, the Jain fairness index increases quickly. The linear dropping of energy is mainly explained by the Poisson task arrival over time; it drops from 63% at the beginning of the experiment to 0% at 153 minutes. At the beginning of the experiment the remaining energy distribution was not fair with a Jain fairness index of 47%. It, however, reaches 80% within 20 minutes only. The network then reaches a steady state and F oscillates around 85% and 93% before dropping to 25% when only one node remains alive in the network at 147 minutes.

Fig. 3-(d) compares the network lifetime of three offloading algorithms; our energy balancing algorithm which we will call Fair, a random offloading algorithm (Random), and optimal time offloading algorithm (Greedy) which consists of always offloading to the nodes with the best computation capabilities (*e.g.,* send always to tablets instead of mobile phones). We show that Fair prolongs the lifetime of the network by 60 to 75% compared to the two other considered offloading algorithms; Fair, indeed, successes on keeping all nodes alive for about 2 hours while Random and Greedy fail within only 40 minutes. We also show that Fair maintains a fairly distributed remaining energy among the nodes as shown in Fig. 3-(a). With the Fair algorithm, the power depletion of a node is an indicator that the other nodes are about to also have their power depleted. This is verified in Fig. 3-(d)where a first battery depletion was directly followed (no more than 25 minutes) by other nodes failures.

[1]20MFLOP is the approximate complexity of a face recognition app., 60 MFLOP is the approximate complexity of a virus scan against a library of 1000 virus sig. when the size of the file system is 1MB

5.3 Two Hop MDC Cluster

We consider all possible combinations of 5 node networks with a diameter of 2 hops. Within such networks, the shortest distance between any two nodes is less than or equal to 2 hops. Data points in the following figures represent the average of all data points given by each network combination. Similar to the previous scenario, we assign random initial battery levels to the 5 nodes and generate tasks following a Poisson arrival process.

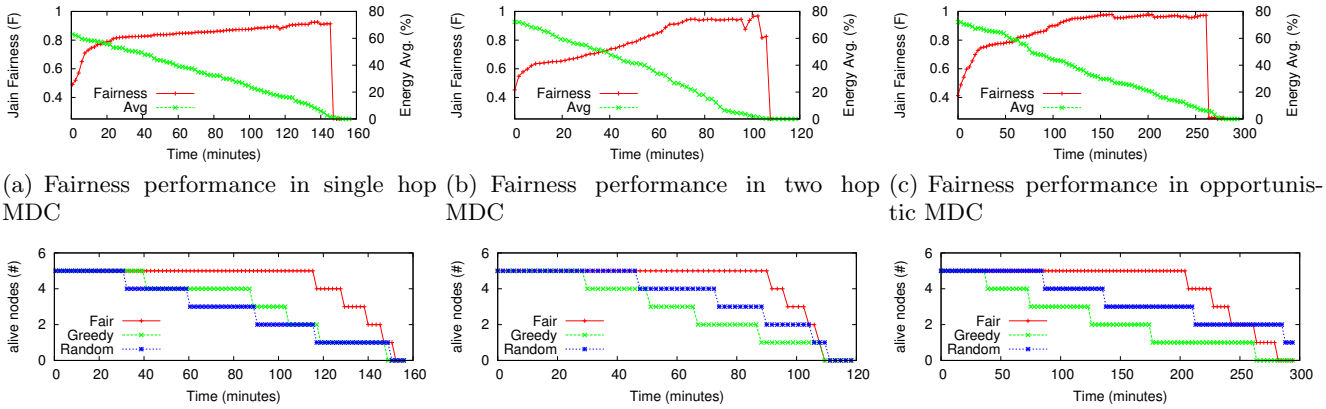
We plot in Fig. 3-(b) the average remaining energy as well as the Jain fairness index over time. Similar to the 1-hop scenario, we show that our Fair algorithm helps to balance energy among all the nodes in the network. The fairness index increases to reach a steady state within 65 minutes. It took more time than the 1 hop scenario to reach a steady state, this can be explained by the fact that in 2 hop paths, a task depletes two nodes' batteries instead of one and makes available energy balancing harder. We also show that the fairness index fluctuates after the first battery drop in the network reaches 0.25% when only one node remains in the network.

Fig. 3-(e) compares the network lifetime using the three previously described algorithms; Fair, Greedy and Random offloading algorithms. We show that our Fair algorithm prolongs the lifetime of the network by almost 50% compared to the random offloading algorithm and 65% compared to the Greedy algorithm. Fair, indeed, helps extend the depletion time of the first node in the network to more than 90 minutes. However, with Greedy offloading one node's battery is depleted within 30 minutes. This is because the algorithm directs all tasks to the most powerful node in the network which impacts its battery and also the batteries of the nodes around it (nodes responsible of forwarding the task to the most powerful one).

5.4 Opportunistic MDC Cluster

We now consider the case of intermittently connected networks where nodes' mobility patterns and/or future contacts between nodes are predictable. Considering a more general scenario with unpredictable contacts will be investigated as a part of our future work.

Our analysis relies on a subset of nodes in the Infocom06 data set [4] which consists of a real user mobility trace of 78 participants attending the IEEE Infocom 2006 conference.



(a) Fairness performance in single hop MDC (b) Fairness performance in two hop MDC (c) Fairness performance in opportunistic MDC

(d) Network lifetime of Fair vs. Greedy vs and Random in a single hop MDC (e) Network lifetime of Fair vs. Greedy vs and Random in a two hop MDC (f) Network lifetime of Fair vs. Greedy vs and Random in opportunistic MDC

Figure 3: Available power balancing for single hop, two hop and opportunistic MDC clusters

We select a subset of 5 nodes forming the highest contact rates between them (Nodes No. 40, 32, 19, 15, and 22 in the data set).

We perform a store-carry-forward algorithm to offload tasks from a node u to another node v. We model the evolution of contacts between the 5 selected nodes by a time varying graph $G(t)$. We assume that each node predicts future contacts by accessing $G(t), \forall t \geq 0$.

A given node u, upon generating a task, estimates the remaining energies throughout each opportunistic path and verifies that the expected response time stays within the deadline assigned to the task. Fig. 3-(c),(f) plot energy balancing performance over time. While opportunistic networks add a longer waiting delay to the task completion time, our Fair algorithm manages to reach 80% Jain fairness index within less that one hour. Then it reaches the steady state within less than two hours. It drops to 25% at 265 minutes when only one node was alive. Other algorithms did not manage to keep all their nodes alive more than 90 minutes.

6. CONCLUSION & FUTURE WORK

In this paper, we have considered the question of how to maximize the lifetime of a highly collaborative mobile device cloud. We accomplish this by developing techniques for balancing the available power across devices under various connectivity assumptions. We developed an experimentation platform that allowed us to obtain realistic power and computation profiles which we then used to evaluate the performance of our power balancing techniques.

This work represented a preliminary study into the question of resource management in highly collaborative mobile device clouds. We believe that effective management of resources across the nodes in such an MDC can significantly improve their utility as powerful computing platforms.

Our work will continue in several directions: 1) We will continue to consider the full spectrum of communication contexts as described in Sec 3. 2) We also plan to consider management considerations for other MDC resources such as device storage. 3) We will also generalize our resource management goals to include some criteria for weighted resource usage to accommodate various prioritization policies among devices.

7. REFERENCES

[1] M. Abdellatif, A. Mtibaa, K. A. Harras, and M. Youssef. GreenLoc: an energy efficient architecture for WiFi-based indoor localization on mobile devices. In *IEEE ICC 2013*.

[2] G. Anastasi, M. Conti, M. Di Francesco, and A. Passarella. Energy conservation in wireless sensor networks: A survey. *Ad Hoc Networks*, 7(3):537–568, 2009.

[3] L. Bai, L. Zhao, and Z. Liao. Energy balance in cooperative wireless sensor network. In *Wireless Conference, 2008. EW 2008. 14th European*, pages 1–5. IEEE, 2008.

[4] A. Chaintreau, A. Mtibaa, L. Massoulié, and C. Diot. The diameter of opportunistic mobile networks. In *Proceedings of ACM CoNext*, 2007.

[5] J.-H. Chang and L. Tassiulas. Energy conserving routing in wireless ad-hoc networks. In *INFOCOM 2000*, volume 1, pages 22–31. IEEE, 2000.

[6] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti. Clonecloud: elastic execution between mobile device and cloud. EuroSys '11, pages 301–314, New York, NY, USA, 2011. ACM.

[7] E. Cuervo, A. Balasubramanian, D. ki Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl. Maui: making smartphones last longer with code offload. In *MobiSys'10*, pages 49–62, 2010.

[8] J. Flinn. Cyber foraging: Bridging mobile and cloud computing. *Synthesis Lectures on Mobile and Pervasive Computing*, 7(2):1–103, 2012.

[9] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *System Sciences, 2000. Proceedings of the 33rd Annual Hawaii International Conference on*, pages 10–pp. IEEE, 2000.

[10] R. Jain, D.-M. Chiu, and W. R. Hawe. *A quantitative measure of fairness and discrimination for resource allocation in shared computer system*. Eastern Research Laboratory, Digital Equipment Corporation, 1984.

[11] A. Kwok and S. Martinez. Energy-balancing cooperative strategies for sensor deployment. In *Decision and Control, 2007 46th IEEE Conference on*, pages 6136–6141. IEEE, 2007.

[12] C. Li, M. Ye, G. Chen, and J. Wu. An energy-efficient unequal clustering mechanism for wireless sensor networks. In *Mobile Adhoc and Sensor Systems Conference, 2005*, pages 8–pp. IEEE, 2005.

[13] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies. The case for vm-based cloudlets in mobile computing. *Pervasive Computing, IEEE*, 8(4):14–23, 2009.

[14] C. Shi, V. Lakafosis, M. H. Ammar, and E. W. Zegura. Serendipity: enabling remote computing among intermittently connected mobile devices. In *MobiHoc*, pages 145–154, 2012.

[15] M. Ye, C. Li, G. Chen, and J. Wu. Eecs: an energy efficient clustering scheme in wireless sensor networks. In *Performance, Computing, and Communications Conference, 2005. IPCCC 2005. 24th IEEE International*, pages 535–540. IEEE, 2005.