2013 IEEE International Conference on Cloud Computing Technology and Science

Towards Computational Offloading in Mobile Device Clouds

Abderrahmen Mtibaa
ECEN Department
Texas A&M University
Doha, Qatar
Email: amtibaa@tamu.edu

Khaled A. Harras
School of Computer Science
Carnegie Mellon University
Doha, Qatar
Email: kharras@cs.cmu.edu

Afnan Fahim
School of Computer Science
Carnegie Mellon University
Doha, Qatar
Email: afahim@qatar.cmu.edu

***Abstract*—**
Many mobile applications overcome their device limitations in computational, energy, or data resources by offloading computations to the cloud. In this paper, we consider environments in which computational offloading occurs *amongst* a set of mobile devices. We call such an environment a *mobile device cloud* (MDC). In this work, we first highlight the gain in computation time and energy consumption that can be achieved by offloading tasks to nearby devices within an MDC compared to a cloud. We then propose and implement an MDC platform that enables the creation and assessment of various offloading algorithms in MDCs. This platform consists of an Android application deployable across MDC devices, and a test bed to measure power being consumed by a mobile device. We utilize this platform to carry out various offloading experiments on an MDC testbed from which we gain interesting insights into the potential for MDC offloading. Results from these experiments show up to 50% gain in time and 26% gain in energy. Finally, we address the offloadee selection problem in MDCs by proposing several social-based algorithms. The potential promise of this approach is shown by evaluating these algorithms using real data sets that include contact traces and social information of mobile devices in a conference setting.

I. Introduction

With lighter and smaller mobile devices replacing laptops and PC's [1], the advent of wearable computing devices like Pebble or Google Glass [2], and the resulting growth in mobile application market expected to reach $30 billion by the end of 2013 [3], mobile user expectations for pervasive computation and data access are unbounded. Yet, the persistent gap in hardware performance between typical hand-held devices and PCs [4] has driven many mobile applications to *not* run in isolation on mobile devices. Various application tasks, such as face recognition, body language interpretation, speech and object recognition, and natural language processing, exceed the limits of standalone mobile devices. Such applications resort to exploiting data and computational resources in the cloud via numerous wireless communication technologies such as 3G, 4G, Wifi, or even Bluetooth.

These trends have sparked researching various problems arising from data and computational offloading to the cloud [4], [5], [6], [7], [8], [9], [10], [11], [12]. Research in this area has mainly focused on profiling and offloading tasks from mobile devices to remote cloud resources [4] [8], automatically transforming mobile applications by provisioning and partitioning its execution into offloadable tasks [6] [11], and more recently, arguing the need for bringing computational resources closer to task initiators in order to save energy [7]. This idea of reducing energy costs or execution time by bringing task executors closer to mobile task initiators, has been further extended by offloading tasks to nearby mobile devices to reduce the latency for some applications [10], or balance power consumption across multiple devices [9].

In this paper, we consider environments in which computational offloading is performed among a set of mobile devices that form what we call a *Mobile Device Cloud (MDC)*. In addition to the trends mentioned earlier, MDCs are becoming a reality with the increase in average mobile devices per user or household [13] [14]. We propose leveraging these nearby computational resources, by offloading appropriate tasks to MDCs, in order to save execution time and consumed energy. We define a task as a combination of *data* required as input, and *computation* performed on this data in order to yield a result. An application is comprised of many tasks with varying data and computational resources required for their completion. We call the task initiator an "*offloader*" and the task executor an "*offloadee*". Time and energy tradeoffs in task offloading decisions to different offloadees, along with addressing specific MDC related challenges, is the focus of this paper. Our work is comprised of the following three contributions.

Our first contribution is implementing an emulation testbed for quantifying the potential gain, in execution time or energy consumed, of offloading tasks to an MDC. This testbed includes an offloading application running on a client, an offloadee server receiving various tasks, and a traffic shaper situated between the client and server emulating different communication technologies. We emulate five communication technologies, based on real measurements, that can be used to offload tasks, namely, Bluetooth 3.0, Bluetooth 4.0, WiFi Direct, WiFi and 3G. We evaluate the gain achieved using each of these technologies for different combinations of data and computation. Overall, offloading to an MDC registers up to 80% and 90% savings in time or energy respectively as opposed to offloading to the cloud. We also observe up to 20% and 35% savings in time or energy respectively by offloading to an MDC as opposed to a cloudlet [7].

978-0-7695-5095-4/13 $31.00 © 2013 IEEE
DOI 10.1109/CloudCom.2013.50

Our second contribution is providing an MDC experimental platform, needed by the research community [10] [9], to enable future evaluation and assessment of MDC-based solutions. We create a testbed that measures the energy consumed by a mobile device while performing various tasks using different communication technologies. In addition, we build a customizable and generic offloading Android-based mobile application that provides control over the amount of data and computation that need to be offloaded as well as the communication interface over which this offloading should occur. The application also measures the total time taken to offload tasks, execute them, and receive the results from other devices within an MDC. We utilize this platform to carry out different MDC offloading experiments. These experiments conducted over several mobile devices guide us to the types of tasks that should be offloaded, and in what scenarios is it better to offload tasks to an MDC versus executing them locally on the offloader device. Our experimental results show that it is possible to gain time and energy savings, up to 50% and 26% respectively, by offloading within MDC, as opposed to locally executing tasks.

Our third contribution is addressing the offloadee selection challenge in MDCs. Since devices in an MDC can naturally become mobile, selecting offloadee devices that quickly leave an MDC, would seriously compromise performance. Therefore, we propose several social-based offloadee selection algorithms that exploit contact history between devices, as well as friendship relationships or common interests between device owners or users. We evaluate these algorithms using the CoNext07 [15] that contains real contact mobility traces and social information for the conference attendees over the span of three days. Our results exhibit the importance of choosing the suitable offloadee subset and the potential promise gained from leveraging social information.

The remainder of this paper is organized as follows. Related work is presented in Section II. We make the case for MDC offloading in Section III. Section IV contains the details of our MDC experimental platform. We present our social based offloadee selection algorithms in Section V. Finally, we conclude and present our future work in Section VI.

II. Related Work

With the rise in demand for computational resources by mobile applications, various solutions for computation offloading to more powerful surrogate machines, known as cyber foraging, have been proposed [4]. Recent solutions include CloneCloud [6] and MAUI [8]. CloneCloud [6] decides, for any given task, whether to execute this task locally or to offload it to a remote cloud. It does not rely on developer effort, and by carrying out static and dynamic analysis, it partitions any given application into tasks that can be offloaded to other devices. It aims to minimizes the execution time of an application by offloading some of its constituent tasks to a cloud, while executing the rest locally. MAUI [8] relies on developer effort to convert mobile applications in a managed code environment to better support fine-grained real-time offload 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 [7] closer to mobile devices. Researchers have proposed just-in-time provisioning of these cloudlets for dynamic virtual machine creation so that mobile tasks can be easily offloaded and executed [11]. In addition, a hybrid cloud architecture, GigaSight [12], has been proposed to enable the collection and analysis of crowd-sourced videos from devices like Google Glass [2]. It achieves scalability by decentralizing the cloud infrastructure needed using cloudlets.

Energy saving solutions [9], Cirrus [5], and Serendipity [10] take some steps towards computational offloading to neighboring mobile devices. Mtibaa et al. [9] discuss offloading algorithms to neighboring mobile devices to ensure a fair consumption of energy across a group of mobile devices belonging to the same individual or household. Cirrus [5] looks into the spectrum of devices that can be used as part of a mobile device cloud, and proposes a holistic solution to cyber foraging which involves offloading not only to other mobile devices, but also to computers installed on moving vehicles or placed in different areas of a building. Serendipity [10], closest to our work, develops a system that handles task allocation in mobile device clouds, and uses emulation to explore the possible speedups gained and energy conserved using offloading in mobile device clouds. However, Serendipity does not consider the tradeoffs between communication technologies available on mobile devices, nor the different cases of data and computation loads. In addition, in this paper, we propose a generic MDC platform, and present solutions exploiting social-based information for leveraging device selection in MDCs.

III. Making the case for Offloading in MDC

In this section, we adopt an experimental approach to highlight the potential gain in energy and time that can be achieved by offloading tasks within an MDC. We investigate the most appropriate combinations of data and computation suitable for remote offloading, as compared to local execution, while considering the different communication technologies available. Since emulation provides a balance between simulation and real world experiments, we implement an emulation testbed as a first step towards evaluating the potential gain of data and computation offloading in mobile environments. This testbed evaluates the gain achieved using different communication technologies including Bluetooth 3.0, Bluetooth 4.0, WiFi Direct, WiFi and 3G.

A. Emulation Testbed Architecture

Our testbed consists of (1) an offloader client, (2) an offloadee, and (3) a traffic shaper, as shown in Fig. 1. We measure the energy consumed and time taken to complete

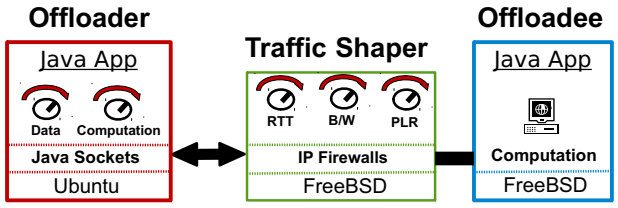


Fig. 1. A high level architecture of our Emulation Testbed

tasks being offloaded from one mobile device (*i.e.,* offloader) to another device (*i.e.,* offloadee). In this section, we consider 3 types of offloadees: a mobile device, a cloudlet, and a cloud.

The client application represents an offloader device in an MDC. We implement it as a Java Sockets based application which allows a user to define tasks as combinations of data and computation. This application sends a user-specified percentage of the pre-defined task to the offloadee application. Such percentage is translated into two main parameters: the computational load measured in total floating point operations (MFLOP), and required data to be transmitted in Megabytes. The client application keeps track of the total time it takes to send the input data to the offloadee device, execute the task at the offloadee device, and receive the results. The client application carries out each offload operation three times and calculates the average total offload time and average offloadee computation time.

The offloadee application is also a Java Sockets based application that waits for any tasks that the client application might offload to it. Upon receiving a task from the client application, it executes it and measures the time taken to carry out the computation. It then sends this time value back to the client application. We currently do not emulate different computation capabilities of the offloadee device.

The traffic shaper emulates the network conditions in order to mimic different communication technologies that we consider in our experimentation. We use Dummynet, a kernel module on top of Free BSD [16]. The traffic shaper receives all packets sent from the offloader to the offloadee. It introduces a predefined level of bandwidth and delay (RTT) constraints to emulate the network conditions provided by each technology being emulated. RTT values are measured by pinging an offloadee listening on each of the network interfaces five times.

B. Experimental Methodology

In our experiments, we vary the computational size (denoted by MFLOP) and data being sent (denoted by MB). We choose to decouple the two previous task parameter (*i.e.,* MFLOP and MB) in order to evaluate the potential gain of different application characteristics. Therefore, we abstract the task computation with a number of multiplications of different predefined matrices, and data is separately represented as String objects of given lengths. We measure the total *completion time* and the *energy consumed* by the offloader device while varying computational size (MFLOP) and data sizes (MB). In this section, we measure the energy consumption only at the offloader device since offoadee devices may not have any energy constraints (*e.g.,* cloud or cloudlet). In addition, we run 50% of the task (computational size and data) locally and offload 50% for remote execution. Measuring the overall energy consumed by an MDC as well as considering different task distributions will be investigated in the next section.

We utilize 10, 30 and 60 MFLOP computation loads[1], which correspond to computational complexities of low, medium and highly complex applications. A detailed method to estimate the MFLOP of a real representative applications can be found in [17]. We also vary the data being offloaded along with the task between 0 and 30 Megabytes. We present results for low, medium, and high computationally intensive tasks in Fig. 2. We first notice that RTT has the largest impact on the offloading performance; offloading to a cloud or offloading using 3G takes much more time and consumes more energy than offloading to a nearby cloudlet or MDC device. Moreover, we show that offloading to an MDC rather than offloading to a cloud is advantageous both in terms of response time and energy consumption. We register up to 80% savings in time by offloading to an MDC as opposed to offloading to the cloud. This considerable gain is amplified with higher computationally intensive tasks as shown in Figs. 2(c), 2(f). We also observe up to 20% and 35% savings in time or energy by offloading to an MDC as opposed to a cloudlet located nearby.

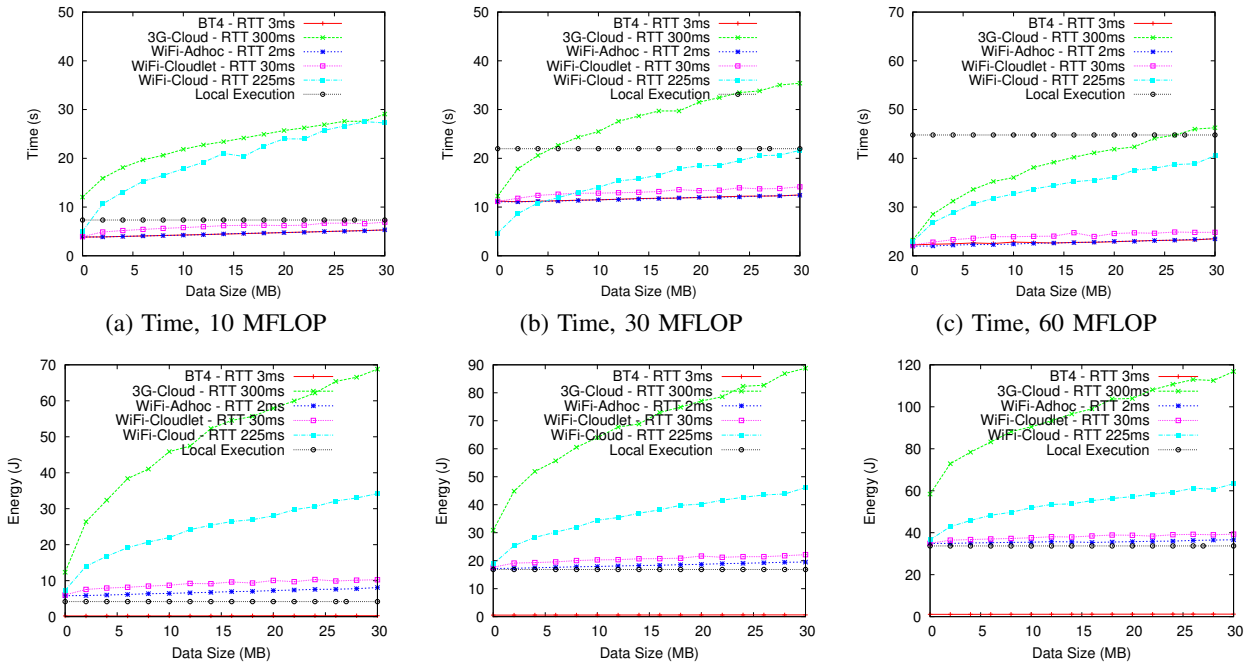
Offloading to a distant cloud consumes significantly more time and energy. However, it can be adopted only for applications that require minimal data exchange (*i.e.,* less than 5 MB in Fig. 2). Cloudlets, which are implemented in our testbed as nearby cloud resources with small RTT values, present a good offloading option for medium or high computationally intensive tasks when compared to a distant cloud.

Emulation, as a first step towards evaluating the potential gain of MDC offloading, shows that MDCs can be an advantageous platform for computational offloading. In the following section, we propose a real implementation for a generic MDC platform in order to validate our emulation results.

IV. MDC Experimental Platform

Researchers in mobile cloud computing typically resort to implementing or migrating representative resource heavy applications on mobile devices in order to evaluate new architectures, task scheduling algorithms, or different offloading 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 characteristics of potential future applications that may not exist yet.

[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



(a) Time, 10 MFLOP (b) Time, 30 MFLOP (c) Time, 60 MFLOP

(d) Energy, 10 MFLOP (e) Energy, 30 MFLOP (f) Energy, 60 MFLOP

Fig. 2. Completion time and energy consumption for low (10 MFLOP), Medium (30 MFLOP), and High (60MFLOP) computationally intensive tasks.

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 device cloud (MDC) experimental platform for mobile cloud computing research. Our platform consists of (1) an energy testbed, and (2) an Android application to carry out experimentation on real world MDCs.

A. Energy Testbed

With the focus now shifting to only MDC offloading, our goal is to learn the energy cost for each potential operation that would take place in various MDC offloading scenarios. Our approach is as follows. We carry out experiments that measure the energy consumed when an MDC device is idle, and when it performs offloading operations, via each wireless technology available. As shown in Fig. 4, we take out the battery of the device being tested and solder wires coming from a power supply into the battery contacts of the device. The power supply that we use comes with a built in ammeter and voltmeter. We then provide a constant voltage according to

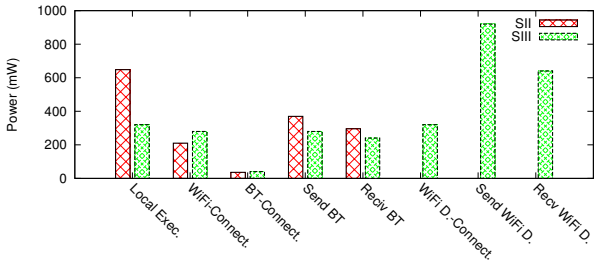


Fig. 3. Results from Energy Testbed Experimentation

the manufacturer specifications and power the device on. Using the current and voltage readings, we are able to determine the power being consumed by the phone at any instance during our experiments.

We measure the power consumption of different tasks running on two different phones: Samsung Galaxy SII and SIII. Tasks are carried out for a minute each to account for system load fluctuations, and for each of the tasks, a base reading is taken before performing the task itself. Thus the power being consumed by the specific task can be calculated by subtracting the base value from the total power being consumed while performing the task. We carry out such experiments for most of the communication technologies we have identified above. We note that Bluetooth 4.0 was not considered in our experiment because drivers for it were not available at the time of experimentation.

Fig. 3 compares different energy measurements while per-

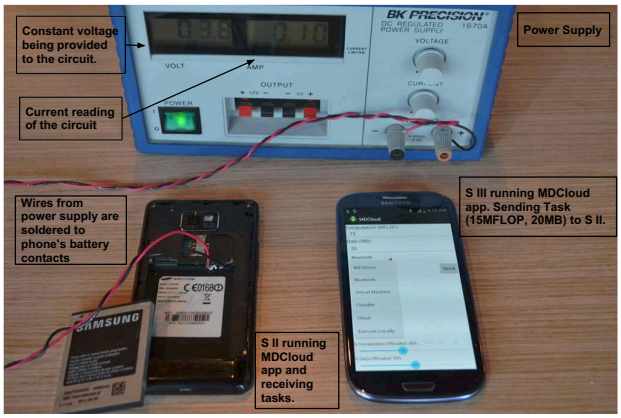


Fig. 4. MDC Testbed - A Scenario

forming wireless transfers using Bluetooth (BT) and WiFi Direct (WiFi D.) between two Samsung SII devices and two Samsung SIII devices. We send the same data size using both Bluetooth and WiFi Direct, and show that Bluetooth is 80% to 120% more energy efficient than WiFi Direct. Moreover, we notice that sending data costs 10% to 25% more energy than receiving data independently of the wireless communication used. This plot confirms the fact that WiFi Direct is an energy expensive technology. In fact, SIII with WiFi Direct connected to another SIII and idle, consumes almost the same energy compared to an SIII sending via Bluetooth to another SIII device. We note that the Samsung SII does not implement WiFi Direct, so we plot only the SIII measurements in Figure 3.

B. MDCloud Application

We implement an application called MDCloud that uses the API we developed which is described in [17], and runs on Android 4.1. The application allows a user to artificially create a task that consists of computation (measured in MFLOP) and input data (measured in MB). Users can also select the wireless communication technology used to offload the defined task among a list of offloadee choices given in a drop down menu. Offloadee choices include 'MDC', 'Cloud' and 'Cloudlet'. In addition, the interface provides two seek-bars that allow the user to specify what percentage of computation and data should be offloaded, and the remaining percentage is executed locally. Once the task has been offloaded, the application logs 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), and the computational time for every device and data transfers separately.

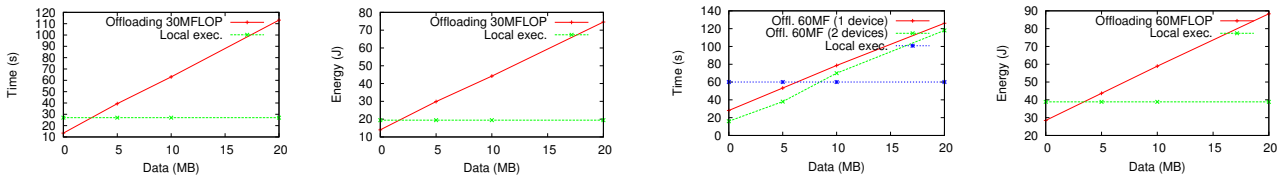
Upon defining a task and the communication technology used, the original task will be fragmented and the selected percentages are offloaded to remote devices as specified by the user. The remaining sub-tasks are executed locally. We have tested the application on multiple phones as well as offloading to one or multiple offloadees.

C. MDC Testbed & Results

We create an experimental testbed consisting of two Samsung Galaxy SII and two Samsung Galaxy SIII phones, all running our MDCloud application. We aim to obtain practical insights into making offloading decisions in mobile device clouds in order to make conclusions about appropriate strategies to be adopted for developing offloading algorithms. In all of these scenarios, we chose Bluetooth as the standard communication technology, because of its energy efficiency and widespread availability in most modern smart devices. Fig. 4 pictures a particular scenario of an MDC consisting of two devices, where one of the devices a Galaxy SIII is offloading a task to a Galaxy SII, and the power being consumed by SII is being measured.

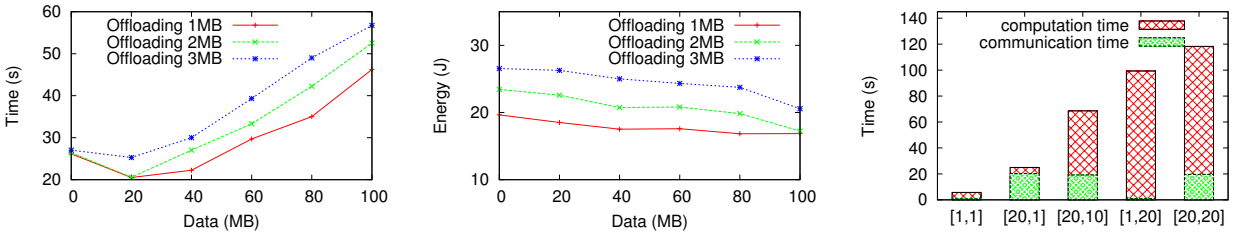
Fig. 5 plots the performance results for medium and high computationally intensive task offloading in MDCs. We measure the *overall energy consumption of both offloader and offloadee device(s)*, as well as the total task response time. We show that experimental results confirm the emulation results shown earlier. For a higher MFLOP value, we show larger gains in both energy and time conservation while offloading the task to another device using MDCs. We register up to 50% gain in time and 26% gain in energy by offloading half the task to one other device (Figs. 5(c) and 5(d)). Fig. 5(c) also shows that offloading to more than one device further improves the gain with respect to time. This is done by running 34% of the task locally and running $2 \times 33\%$ remotely on two offloadee devices. We achieve up to 40% gain in time by offloading to multiple devices compared to a single device. This gain is reduced, for larger data sizes, because the overhead introduced by exchanging such sizes mitigates the gain achieved by distributing the computation among multiple devices.

We then vary the percentage of computation running on the offloadee device, between 0% and 100%, in order to investigate the tradeoff between computation and communication to maximize gains in time and energy. Determining optimal task splitting during run-time will be investigated in future work. Fig. 6(a) shows that the best performance is registered when only 20% of the task is offloaded to the offloadee device and the rest is executed locally; such distribution achieves up to 51% gain in time. In terms of energy (Fig. 6(b)), the best gain is registered while offloading 100% of the task to another device, yielding up to 16% conservation in energy [17]. This leads us to an important question: what is the relationship between the energy consumption for data transmission versus computation during offload operations? Fig. 6(c) shows that communication can take up to 100 times more time than computation does. This is also shown when 25% increase in data costs 4x more time than a 25% increase of task complexity. This analysis demonstrates that having more computation provides avenues for more gain to be achieved in terms of time and energy conservation, while having more data means a reduction in the energy and time that can be achieved by offloading tasks into an MDC.



(a) Resp. Time, 30MFLOP (b) Energy, 30MFLOP (c) Resp. Time, 60MFLOP (d) Energy, 60MFLOP

Fig. 5. Response time and energy consumption for Medium (30MFLOP), and High (60MFLOP) computationally intensive tasks



(a) Time vs offloading perc. (b) Energy vs. offloading perc. (c) Computation vs. Communication

Fig. 6. Understanding offloading tradeoffs in MDC

V. Offloadee Selection in MDC

As opposed to traditional cloud computing architectures where the offloadee is always a distant cloud, offloder devices in an MDC face a very challenging problem which is selecting the most suitable set of offloadee devices in order to improve performance.

A. Problem Statement

We consider a mobile wireless network where devices may be intermittently connected to each other which may cause long periods of disconnection and large end-to-end communication delays. In such network data is transferred using opportunistic communication over intermittent links. We considered a scenario where an offloader device u initiates a task T_t consisting of k subtasks at a given time t. We assume that the offloader device has successfully partitioned the computational task into subtasks T_t, our goal is mainly how to allocate such tasks for remote processing by the entities it encounters. *The goal of the mobile device u is to use the available, potentially intermittently connected, computation resources in a manner that improves its computational experience,* e.g., *minimizing local power consumption and/or decreasing computation completion time and/or monetary cost*.

In this section, we aim to identify *stable and durable connections opportunities* to other devices based on social or contact history information. We do not consider, however, device capabilities and device remaining energies in our selection process. This will be considered in future work.

B. Offloadee Selection Algorithm

We propose 4 social based offloadee selection algorithms that leverage social information in order to identify and therefore select the most suitable neighboring offloadee nodes. Our proposed algorithms are very similar and differ only with the social graph used to connect two device owners. We consider 4 *Types* of social connections: friendship-based (F-based), interest-based (I-based), a combination of friendship and interest-based ((FandI)-based), and contact history based (C-based). Alg. 1 presents a high level pseudo-code of each algorithm according to its $Type$ ($Type = \{F, I, FandI, C\}$). We assume that nodes store and update one or more social databases such as user friendship graph (*e.g.,* Facebook). Alg. 1 uses such databases to select the best offloadee candidate among the set of neighboring nodes at time t. Selection is simply correlated with the distance in the social graph; *i.e.,* friends which are at distance 1 in the friendship graph will be prioritized compared to users that have only common friends (*i.e.,* distance 2 in the friendship graph).

Algorithm 1 forward($u, T_t, subtasks, Type$)

Require: $subtasks > 0$
Require: $Type \leftarrow \{F, I, FandI, C\}$ ▷ friend, interest, both, or contact-based forwarding
Require: $Neighbors \neq \emptyset$ ▷ node u is surrounded with neighboring devices
1: **function** SELECTBESTCANDIDATE($Type, Neighbors$) ▷ select the best Type-based neighbor candidate
2: **end function**
3: **for** $subtasks \geq 0$ **do**
4: $v \leftarrow$ SELECTBESTCANDIDATE($Type, Neighbors$)
5: $Send(v, T_t/subtasks)$
6: $Neighbors \leftarrow Neighbors - \{v_k\}$
7: $subtasks = subtasks - 1$
8: **end for**

C. Evaluation

We first run a set of preliminary experiments using the MDC platform previously described. We perform many experiments

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 I
EXPERIMENTAL RESULTS FOR BLUETOOTH OFFLOADING TASKS FROM S2 TO S3: [SEND ; RECV ; COMP] ENERGIES IN JOULES

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 the response time and consumed energy for each task (*i.e.,* energy consumed upon sending, energy consumed upon receiving, and energy consumed upon computing a task) as shown in Tab. I. We vary computation from 0 MFLOP to 60 MFLOP. We also vary the input data sizes from 0 to 20MB (*e.g.,* compressed video file). Note that Tab. I 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 47 mobile devices. Nodes move according to the CoNext07 real world mobility trace [15]. Nodes are also socially connected according to the same CoNext07 trace which has Facebook and real life social connections between nodes. We assume that all nodes have the same battery level and enough energy to run the whole experiment. We emulate a traffic trace where nodes generate tasks independently. Tasks are generated following a Poisson arrival process of rate λ. Task characteristics (*i.e.,* complexity and data size) are selected randomly from Table. I. We run the two main experiments in order to evaluate the task completion time and the overall energy consumption of social-based offloadee selection algorithms.

Our first experiment aims to evaluate the ability of different social-based offloadee selection algorithms to identify and select stable connections. We, therefore, select time-slots within the data set where at least 5 nodes are within direct proximity; *i.e.,* one node is in contact with at least 4 other nodes at a given time t as shown in Eq. 1.

$$\exists t \geq 0, \exists n \in N, degree(n) \geq 4 \quad (1)$$

where, n is a node belonging to a set of nodes N is the data set, $degree(u)$ is a function that computes the proximity degree of a given node u.

We assume that such a node is an offloader and the neighbors are potential offloadees. The goal of any algorithm is to select stable connection with neighboring nodes in order to ensure that the connection duration between the two encountered nodes remains during (i) the migration of the task (*i.e.,* exchange of data), (ii) task execution in the remote node, and (iii) sending back the results. We evaluate the ability

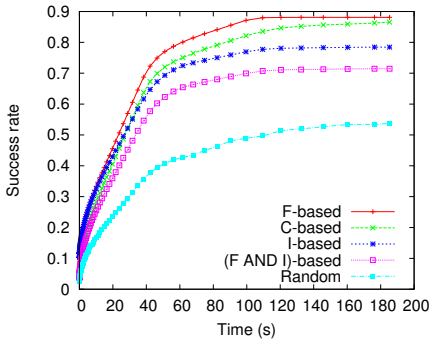


Fig. 7. Performance evaluation of our forwarding scheme

of each of our social-based offloadee algorithm to identify stable connections using all selected times-lots. We therefore measure the success rate of each task defined as the fraction of completed tasks within any time t. A 100% success implies that all tasks and subtasks have been migrated, executed, and results were sent back successfully.

Fig. 7 plots the success rate of all considered social-based selection algorithms compared to a random selection where the offloder randomly picks offloadees. Results represent an average of all success rate values within selected times-lots in the CoNext07 data set according to Eq. 1. Fig. 7 shows that the friend-based selection algorithm outperforms all other considered schemes. It achieves up to 40% increase compared to random selection and 15% more success rate than the interest-based algorithm. We also show that the contact based selection algorithm takes more time to reach a near optimal success rate; this is mainly explained by the fact that the contact-based algorithm uses contact history to estimate social relationship, so the more time it takes the better is its approximation. Unexpectedly, the combination of friendship and interests is ranked 4^{th} with regards of success rate. This is mainly due to the very conservative social definition that results in selecting only a few nodes satisfying this definition (*i.e.,* nodes that are both friends and share at least three common interests).

Fig. 7 also shows a very important results which is that none of the considered algorithm reach a 100% success rate within more than 3 minutes. This implies that there is a need to implement mechanisms such as task offloading redundancy or aggregation in order to improve the performance of these algorithm.

Our second experiment consists of running the experiment

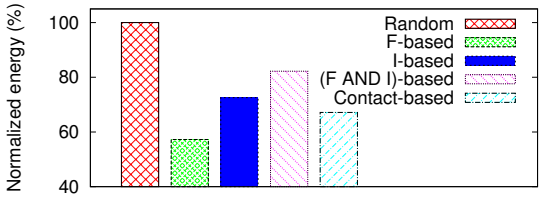


Fig. 8. Performance evaluation of our forwarding scheme

until all the algorithms reach 100% success rates. We therefore compute the energy consumed by each algorithm. The difference in energy is mainly due to the interface idle tim on the offloader until it receives the results since the energy consumed while sending data is identical for all considered algorithms.

Fig. 8 plots the energy consumption of the offloader node while running each selection algorithm normalized by the energy consumption while running the Random based selection algorithm. As expected, the friendship based solution achieves the best energy consumption. It consumed 40% less energy than Random selection and up to 20% less energy than other considered algorithms. This gain in energy is mainly due to a reduction on the waiting time of the offloader.

VI. Conclusion and Future Work

This paper aims to motivate the use of mobile devices in computational offloading in order to ave time and energy. We have introduce mobile device clouds (MDCs) representing environments in which computational offloading is performed among a set of mobile devices. We have shown up to 80% and 90% savings in time or energy respectively as opposed to offloading to a distant cloud. We have also observed up to 20% and 35% savings in time or energy by offloading to an MDC as opposed to a nearby cloudlet.

We have provided an MDC platform that allows offloading algorithms to be tested on actual mobile device clouds consisting of multiple devices. We have used this platform to carry out experiments that have provided insights into designing automated offloading decisions. We have shown up to 50% gain in time and 26% gain in energy by offloading within MDC, as opposed to locally executing tasks.

Finally, we have also addressed the offloadee selection challenge in opportunistically connected MDCs. Opportunistic communication causes long periods of disconnection and large end-to-end communication delays which introduces large task termination delays. We have proposed several social-based offloadee selection algorithms. These algorithms exploit (i) contact history between devices, as well as (ii) friendship relationships or common interests between device owners or users in order to estimate the most suitable offloadee devices that can carry out the computation with a lower risk of disruption. We have adopted a real trace based experiment to highlight the importance of choosing suitable offloadee subsets and the potential promise gained from leveraging social information.

While we have looked into making offloading decisions based on the contents of the task at hand, in the future, we plan to implement an automated decision maker engine that leverages users profiles and device capabilities and make real time offloading decisions in order to minimize response time and overall energy. Such engine will consider and compare all offloading options and compute an utility function in order to select the best option.

References

[1] (2013) Microsoft tag survey. [Online]. Available: http://goo.gl/VH1tp

[2] (2013) Techcrunch google glass api article. [Online]. Available: http://goo.gl/XYJz6

[3] (2010, January) Gartner says consumers will spend 6.2 billion in mobile application store in 2010. [Online]. Available: http://www.gartner.com/newsroom/id/1282413

[4] J. Flinn, "Cyber foraging: Bridging mobile and cloud computing," *Synthesis Lectures on Mobile and Pervasive Computing*, vol. 7, no. 2, pp. 1–103, 2012.

[5] C. Shi, M. H. Ammar, E. W. Zegura, and M. Naik, "Computing in cirrus clouds: the challenge of intermittent connectivity," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, ser. MCC '12. New York, NY, USA: ACM, 2012, pp. 23–28. [Online]. Available: http://doi.acm.org/10.1145/2342509.2342515

[6] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "Clonecloud: elastic execution between mobile device and cloud," in *Proceedings of the sixth conference on Computer systems*, ser. EuroSys '11. New York, NY, USA: ACM, 2011, pp. 301–314. [Online]. Available: http://doi.acm.org/10.1145/1966445.1966473

[7] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *Pervasive Computing, IEEE*, vol. 8, no. 4, pp. 14–23, 2009.

[8] 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*, 2010, pp. 49–62.

[9] A. Mtibaa, A. Fahim, K. Harras, and M. Ammar, "Towards resource sharing in mobile device clouds: Power balancing across mobile devices," in *Proceedings of the second edition of the MCC workshop on Mobile cloud computing*, ser. MCC '13. New York, NY, USA: ACM, 2013.

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

[11] K. Ha, P. Pillai, W. Richter, Y. Abe, and M. Satyanarayanan, "Just-in-time provisioning for cyber foraging," in *Proceeding of the 11th annual international conference on Mobile systems, applications, and services*, ser. MobiSys '13. New York, NY, USA: ACM, 2013, pp. 153–166.

[12] P. Simoens, Y. Xiao, P. Pillai, Z. Chen, K. Ha, and M. Satyanarayanan, "Scalable crowd-sourcing of video from mobile devices," in *Proceeding of the 11th annual international conference on Mobile systems, applications, and services*, ser. MobiSys '13. New York, NY, USA: ACM, 2013, pp. 139–152.

[13] CDB. (2011, January) Household penetration rates for technology across the digital divide. [Online]. Available: http://communities-dominate.blogs.com/brands/2011/01/household-penetration-rates-for-technology-across-the-digital-divide.html

[14] Bloomberg. (2012, August) Average household has 5 connected devices, while some have 15-plus. [Online]. Available: http://goo.gl/ob6Tkh

[15] A. Mtibaa, A. Chaintreau, J. LeBrun, E. Oliver, A.-K. Pietilainen, and C. Diot, "Are you moved by your social network application?" in *WOSN'08: Proceedings of the first workshop on Online social networks*. New York, NY, USA: ACM, 2008, pp. 67–72.

[16] L. Rizzo, "Dummynet: a simple approach to the evaluation of network protocols," *SIGCOMM Comput. Commun. Rev.*, vol. 27, no. 1, pp. 31–41, Jan. 1997.

[17] A. Fahim, A. Mtibaa, and K. Harras, "Making the case for computational offloading in mobile device clouds," Carnegie Mellon University, Tech. Rep. CMU-CS-13-119, June 2013.