Collaborative Mobile-To-Mobile Computation Offloading

Abderrahmen Mtibaa*, Mohammad Abu Snober†, Antonio Carelli†, Roberto Beraldi†, Hussein Alnuweiri*

*Texas A&M University Qatar
{amtibaa, alnuweiri}@tamu.edu

† Sapienza Università di Roma
abusnober@dis.uniroma1.it, carelli.1242046@studenti.uniroma1.it, beraldi@dis.uniroma1.it

***Abstract*—**
It is common practice for mobile devices to offload computationally heavy tasks off to a cloud, which has greater computational resources. In this paper, we consider an environment in which computational offloading is made among collaborative mobile devices. We call such an environment a *mobile device cloud* (MDC). We highlight the gain in computation time and energy consumption that can be achieved by offloading tasks with given characteristics to nearby devices inside a mobile device cloud. We adopt an experimental approach to measure power consumption in mobile to mobile opportunistic offloading using MDCs. Then, we adopt a data driven approach to evaluate and assess various offloading algorithms in MDCs. We believe that MDCs are not replacing the Cloud, however they present an offloading opportunity for a set of tasks with given characteristics or simply a solution when the cloud is unacceptable or costly. The promise of this approach shown by evaluating these algorithms using real datasets that include contact traces and social information of mobile devices in a conference setting.

I. Introduction

Mobile devices are increasingly being a mandatory gadget in our modern life. We rely on these tiny devices for many services that go beyond simple communication. These services require complex processing power and heavy energy consumption such in pattern recognition, reality augmentation, collaborative applications for planning and coordination. These mobile services became an indispensable part of everyday life.

Mobile users tend to access theses services via distant cloud, which has greater computational resources. However, this type of task offloading is costly due to high energy costs and high latency introduced by wireless intermittent communication between mobile devices and distant clouds. Smaller clouds, called 'Cloudlets' were proposed to make mobile task offloading less expensive [1]. Cloudlets are placed closer to users (*e.g.,* users A-D are connected to the cloudlet in Fig. 1). Bring computation resources closer to users motivates the idea of offloading tasks to nearby devices Fig. 1 shows a scenario where mobile devices, *e.g.,* nodes E-G, may not reach cloud and cloudlets resources. They are, however, able to communicate/cooperate with each others to run tasks that transcend an individual devices capabilities.

In this paper, we consider environments in which computational offloading is performed among a set of collaborative mobile devices that form what we call a *Mobile Device Cloud (MDC)* [2]. In addition to the trends mentioned earlier, MDCs are becoming a reality with the increase in average mobile devices per user or household [3] [4]. We propose leveraging these nearby computational resources, by offloading appropriate tasks to MDCs, in order to save execution time and consumed energy. Mobile applications consist of many tasks with varying data and computational resources. We call the task initiator an "*offloader*" and the task executor an "*offloadee*". We are interested in measuring time and energy tradeoffs in task offloading decisions to different offloadees, along with addressing specific MDC related challenges. Our work is comprised of two main contributions. While we believe that MDCs are not replacing the "Cloud", they present a "good" solution for mobile devices (*i.e.,* offloader nodes) to migrate a set of tasks with given characteristics. Demanding tasks requiring high resources or centralized view/oracle can not be offloaded using MDCs. In addition MDC can be considered as a solution when the cloud is unacceptable or costly.

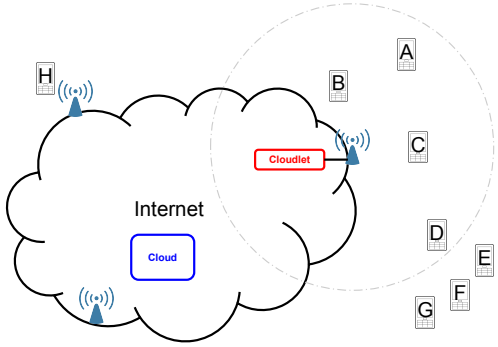


Fig. 1. Mobile-to-Mobile offloading: Motivational scenario

Our first contribution is highlighting and quantifying the benefits of collaborative mobile to mobile task offloading. We adopt an experimental approach that performs time and power measurements for MDC offloading. We also perform a data driven experiment consisting of an MDC network formed with

COLLABORATECOM 2014, October 22-25, Miami, United States
Copyright © 2014 ICST
DOI 10.4108/icst.collaboratecom.2014.257610

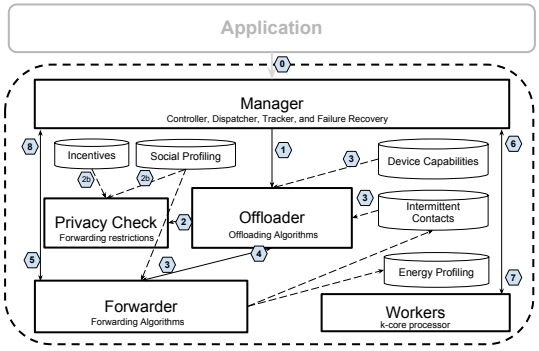


Fig. 2. Collaborative Mobile-to-Mobile opportunistic offloading architecture: a per architecture implemented on the "offloder" and the "offlodee" devices

up to 70 offlodee nodes.

Our second contribution consists of leveraging social information shared by devices' owners to select most appropriate offlodee nodes for a given task. We exploit friendship relationships and common interests as a mean to identify long and stable connection between two devices. We evaluate our social method using the Sigcomm09 dataset that contains real contact mobility traces and social information for the conference attendees. Our results exhibit the importance of choosing the suitable subset of offloadee nodes and the potential promise gained from leveraging social information.

II. Mobile-to-Mobile Opportunistic Offloading Architecture

In this section, we propose a novel peer-to-peer architecture for mobile data and computation offloading which we call a Mobile Device Cloud (MDC). Our MDC architecture abstracts all network entities into a single peer architecture as shown in Fig. 2. This architecture provides mechanisms to answer two main questions: given a task T, *(i)* when does it make sense to offload?, and *(ii)* if offloading is required, who to offload to?.

Our system architecture, illustrated in Fig. 2, is designed to accommodate data and computation offloading over network connection disruption. Each mobile opportunistic peer consists of a task manager, task offloader, task forwarder, a set of workers, a privacy and security check and a set of databases.

The task manager is responsible of receiving jobs and scheduling the tasks belonging to each job. It maintains the status of each task and job and initiates requests to the offloader to run a task or to the failure recovery engine to re-assign a delayed task. Upon receiving a task job from the application (*i.e.,* step 0) or receiving an existing task from a neighboring peer throughout the forwarding engine (*i.e.,* step 5), the task manager initiates a task status locally. It therefore assign a task id and stores its status and characteristics. Each task can therefore be assigned to one of the existing local worker (*i.e.,* 6), or sent to a remote peer workers in the network.

The offloader is key core of our collaborative Mobile-to-Mobile opportunistic offloading architecture. It decides whether and what computation to migrate. First it interacts with the privacy and security engine in order to check if the task T is eligible for offloading or not (*i.e.,* step 2). It runs an objective function that compares running the task T locally or migrating it to k neighboring devices with minimum resource capabilities

$$R_{min} = \{E_{min}, C_{min}, S_{min}\}$$

within δt, where E_{min}, C_{min}, and S_{min} represent respectively the minimum expected energy, computation and storage capabilities on the neighboring opportunistic peer. If a neighboring device u satisfies such requirement, it will then be considered as a potential offloaded, otherwise it gets ignored. If the requirements are satisfied, R_{min} will therefore be sent to the forwarder which will be checking if it can satisfy the requested conditions within the δt deadline. Otherwise, the forwarder will send back a summary of the number k' of neighboring devices and their capabilities. The offloader is then making a final decision and sends back to the forwarder whether T will be running on one of the local (*i.e.,* step 6) or the distant workers (*i.e.,* the task is ready to migrate).

The forwarder is also responsible of updating the databases with the available connections to the neighboring devices and their capabilities. Upon receiving an order to migrate the task to distant devices, the forwarder compares k and k' and select the most suitable devices to run the task T within the given time and resource constraints. It uses stored information about historical contacts summary and social information to infer the expected connection and inter-connection duration between neighboring devices. The forwarder is also receiving tasks from a neighboring forwarder engine. In this case it forwards the task to the manager and then to the workers and sends back the results to the neighboring forwarder peer who initiates the task.

In the rest of this paper, we will focus mainly on the task offloader engine. We believe that offloading task to the most appropriate devices helps reducing the total execution time as well as the total energy consumption.

III. Mobile Offloading Evaluation

We define mobile offloading as migrating data and/or computation to a more suitable computation machine (peer). It involves making a decision regarding *whether and what computation to migrate*. Our goal is mainly to: (i) identify and quantify the potential gain of migrating tasks to distant collaborative mobile devices as opposed to run them locally (*i.e.,* at the initiator device), and (ii) identify under which circumstances migrating is "advantageous".

In our evaluation, we consider two main metrics that may impact such decision (i) improving response time requirements, and (ii) saving energy. We have adopted two evaluation approaches: experimental and data-driven approaches.

A. Experimental Approach

First we run a set of preliminary experiments to quantify the time and energy required upon sending, receiving, and

(a) Mobile-to-Mobile offloading android application [5]

(b) Energy measurement circuit

Fig. 3. Our experimental setup

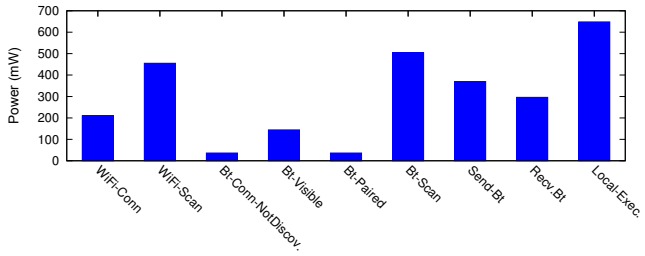


Fig. 4. Power measurement upon performing task offloading using a Samsung Galaxy S2

running tasks on a distant collaborative mobile device. Our experimental platform consists of two mobile devices running an android client application (Fig. 3(a)) and an energy measurement circuit as shown in Fig. 3(b). Details on our experimental platform can be found in [5].

Fig. 4 plots different energy measurements while performing wireless transfers using Bluetooth (Bt) between two Samsung SII devices. We measured different states of both WiFi and Bluetooth interfaces. We also perform sending and receiving messages of a fixed data using Bluetooth, we show that sending data costs 10% to 25% more energy than receiving data independently of the wireless communication used.

B. Data Driven Approach

We consider a collaborative mobile opportunistic networks where nodes are intermittently connected via a new communication environment characterized by a variety of new challenges such as mobility, disconnections, and energy constraints. We perform simulation based on real mobility traces

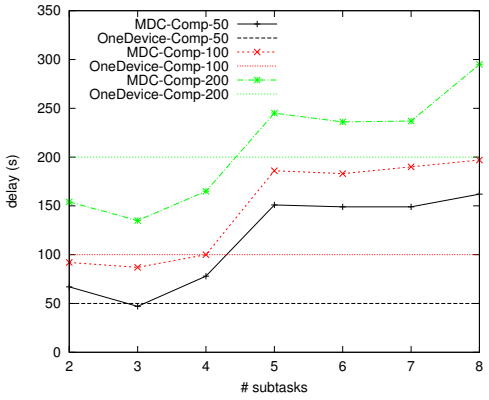


Fig. 5. Offloading tasks to other MDC devices without replication using the Cambridge trace

and the power measurements shown in the previous section as well in [6].

1) Dataset & Methodology: We perform a data driven simulation of an opportunistic network using the One simulator[1]. Node mobility is guided by two experimental mobility traces Cambridge and Sigcomm09[2]. We translate the proximity based traces into movement traces where each node gets x, y, z coordinates at each time stamp. We fix the network area to $150m \times 150m$ which represents approximatively the area of the Cambridge lab.

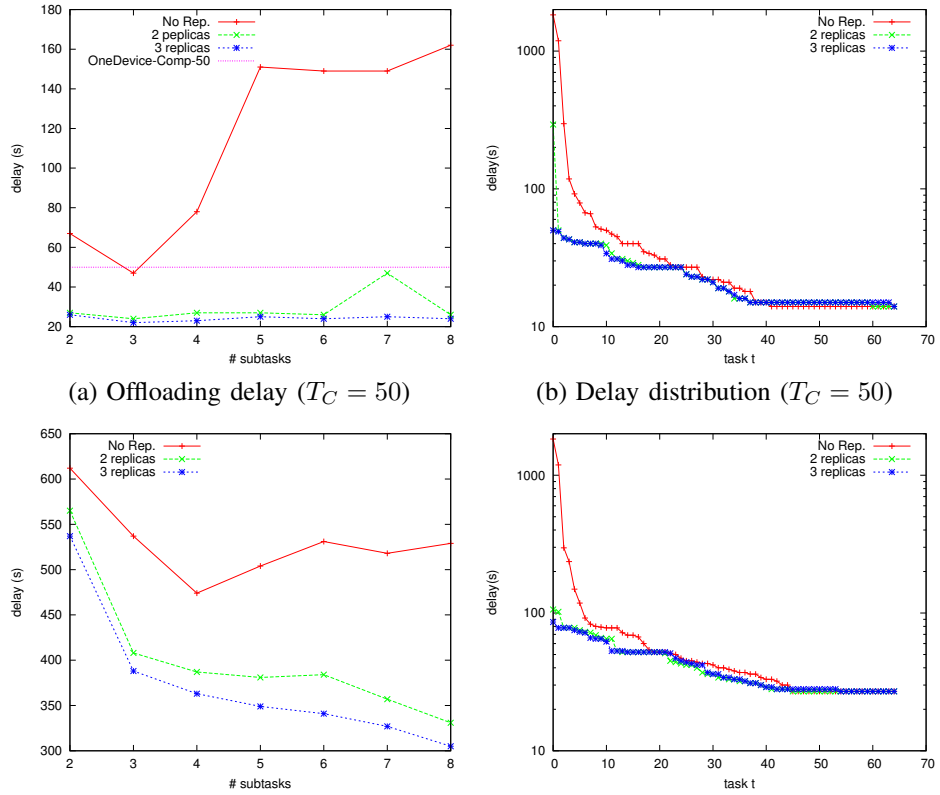
We assume that an offloader device u initiates a task T_{t_i} for offloading at every time $t_i = t_{i-1} + 500s$. We assume that the offloader device has successfully partitioned the computational task into subtasks. Our offloading scheme works as follow: Whenever a given node, u, comes in contact with another MDC node v, it decides to offload a set of subtasks to v. We consider a synthetic workload composed of a task taking a certain amount of time to complete, T_C. When the task is divided into n subtask, each subtask takes T_C/n to complete. We assume that the task has a fixed data size of $1MB$. We consider 4 random offloader nodes and we repeat each experiment 4 times. Our data results represent an average of these set of runs.

The Cambridge and the sigcomm09 datasets consists respectively of 62 and 76 nodes. Both datasets include Bluetouth contact logs between participant nodes with a granularity of $120s$. The sigcomm09 contains also social relationships between the participants including friendships and interests. While the Cambridge trace uses iMotes carried by participants to collect proximity data, sigcomm09 uses a MobiClique application that logs proximity data as well as building and maintaining a social graph connecting all participants. In our experiment, we have processed the traces by removing the period of inactivity which corresponds to nighttime (*i.e.,* from 08:00 to 20:00).

We implement a greedy forwarding algorithm that consists

[1] netlab.tkk.fi/tutkimus/dtn/theone/

[2] crawdad.cs.dartmouth.edu



(a) Offloading delay ($T_C = 50$) (b) Delay distribution ($T_C = 50$)

(c) Offloading delay ($T_C = 1000$) (d) Delay distribution ($T_C = 1000$)

Fig. 6. Impact of task replication on execution time using the Cambridge trace

of sending the first subtask to the first encountered offloadee node, then the second subtask, and so on. Upon receiving a result for a given subtask the offloader node will remove such subtask from the pool.

2) Offloading Without Task Replication: In this section, we consider mobile offloading of a given task without replications. We therefore test the ability of an offloader node to migrate a task and wait for the results from a single remote node (*i.e.,* offloadee). Without replication, the connection between the two nodes can potentially fail (*e.g.,* , intermittent connectivity, node failure, network failure). Failure cause a huge delay and therefore considerable degradation in offloading performance.

Fig. 5 shows the total execution time in function of the number of substasks of a given task T where $T_C = 50, 100$. We show that the delay increases with the number of subtasks and then stabilizes for a number of subtask greater than 5. Delay is mainly caused by opportunistic communication where a given node waits to meet an offloadee and then waits again to get the result from the selected offloadee. Migrating tasks to other MDC devices improves the performance compared to local execution. For example, for $T\_C = 200$ and number of subtasks equal to 3 the execution time passes from $200s$ to about $140s$, with a reduction of 30% with a speed up of 1,4. In addition, we show that when the task requires execution time increases the probability to loose connectivity between the two encountered nodes increases which introduces additional waiting delay.

3) Impact Of Task Replication on Execution Time: We implement a replication technique that consists of replicating the subtasks that exist in the offloader's pool. Once a subtask terminates, all replicas in the pool are removes. Fig. 6 shows the impact of the replication on the execution time for $T_C = 50, 1000s$. First we show that replication improves performance considerably with up to $4\times$ speedup. This means that 2 replicas and number of 3 subtasks works at its best and that increasing the number of subtasks or the replication factor has no additional evident effect. The distribution of task execution time in both scenarios shows that some task can take very large delay which is mainly cause by wireless distribution connection between the offloader and its offloadees. Execution times range from $2000s$ to only $15s$. However, we see that only 5% to 10% of tasks were executed in more that $100s$.

4) Impact Of Task Replication On Energy Consumption: We measure the energy consumed at the offloader node which consists mainly of data transmission, local task execution and Bluetooth scanning power consumption. Fig. 7 shows the energy consumption for the case of $T_C = 50s$. We show that offloading helps reducing the energy consumption by up to 50% compared to local execution. Note that we consider only the offloader energy consumption as we want to migrate tasks in order to save energy and increase a lifetime of a particular node. Overall energy consumption measured for both offloader

and offloadees is higher that local execution.

5) Leveraging Social Information In Opportunistic Forwarding: So far, we have been offloading task using a greedy technique based on a first come first served approach. Next, we perform an approach to the use of social interaction as a means to guide computational offloading in an opportunistic network. It has been shown that that social interaction information can be used to enable and enhance node cooperation is fundamental for the message delivery process[7], [8], [9]. We investigate identifying stable and durable connections to other devices based on social information. We do not consider, however, device capabilities and device remaining energies in our selection process. This will be considered in future work.

We propose a simple social based offloading algorithms that leverage social information such as friendship and common interests in order to identify and therefore select the most suitable neighboring offloadee nodes. Our algorithm performs as follow: whenever the offloader node u encounters another node v it decides to offload a given subtask if and only if: (i) v is a friend of u, or (ii) k friend nodes already received subtasks, where $k > 0$. Our algorithm prioritizes offlodee friends because they tend to stay closer to the offloader which may reduce connection lost.

We plot, in Fig. 8, the execution time of tasks with complexity $T_C = 100$, and 200 while replication factor is fixed to 4 replicas per task. We compare our social technique to the greedy algorithm presented earlier. We show that social tends to improve slightly the performance and reduces delay by up to $10s$. While this improvement is not significant but we believe that it is promising. Investigating more social futility function may provide even better performance by making the execution time distribution curves, shown if Fig. 6, more flat.

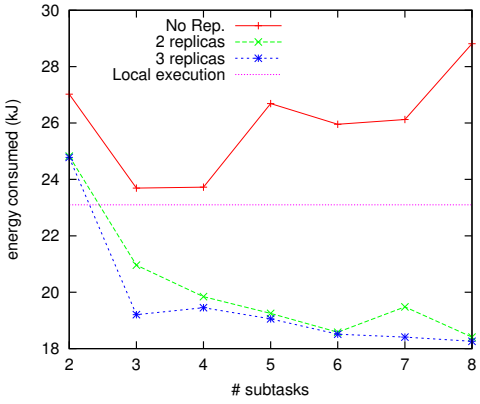


Fig. 7. Energy consumption ($T_C = 50$) using the Cambridge trace

IV. Related Work

Leveraging mobile networking and cloud computing attracts many researchers nowadays [10]. Recently, CloneCloud presents a solution which decides whether to execute a task on a remote cloud service versus executing it locally based on static analysis and dynamic profiling information of a task [11]. Similarly, MAUI investigates the energy consumption challenge when offloading computationally heavy tasks to a cloud rather than executing locally [12]. MAUI relies on developer effort to convert mobile applications to support such decision making, and secondly, it only considers the possibility of offloading to different types of infrastructures.

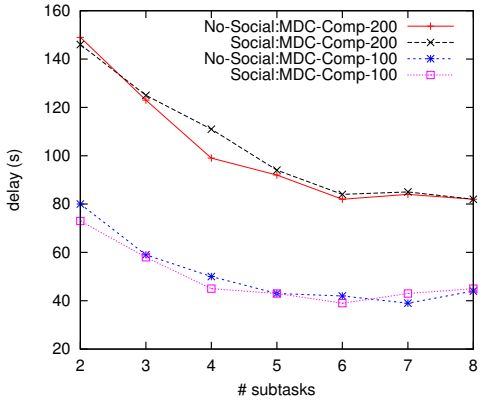


Fig. 8. Impact of social data on task offloading performance using the Sigcomm09 trace

These techniques provide task offloading mechanisms to a distant or nearby could, and does not investigate task offloading using other wireless infrastructures. Cirrus [13] proposes a first step towards defining all possible task offloading opportunities by proposing three different scenarios. These scenarios include offloading to (1) a cloud, (2) a nearby cloudlet [1] and (3) another mobile device. Serendipity focuses on the third previously described scenario [14]. It addresses the challenges of mobile to mobile task offloading when the network is intermittently connected. It presents a model for framing computational tasks, and using this model it presents algorithms that can be used to disseminate tasks to other mobile devices. However, Serendipity does not tackle the energy consumption challenges in theses networks. We believe that even though time can be saved by offloading computation, a lot of energy is wasted in offloading a task, and then waiting until the output is received. It is therefore important to consider the two metrics time and energy jointly for mobile task offloading.

Utilizing mobile devices to support computation originating from other mobile devices is a solution directly relevant to our work [5], [15]. This is made possible because of the availability of improved connectivity options for mobile devices to make use of remote processing and storage. Our work adopts an experimental approach towards testing mobile to mobile offloading in a presence of wireless intermittent connectivity. Such opportunistic communication brings a set of challenges to mobile-to-mobile offloading paradigm.

V. Conclusion and Future Work

This paper aims to motivate the use of mobile devices in creating mobile device clouds that can be used to save time

and energy when it comes to executing computationally heavy tasks. We have shown up to 4 times savings in time or energy as opposed running locally. We have shown that replication is mandatory to overcome connection lost and failure. However, we have shown also that 2 replicas can be enough to improve the performance significantly.

Finally, we have also addressed the challenge of offloading tasks to appropriate nodes. We have proposed a social-based algorithm that aim at identifying stable connections between a given offloader and its offloadees. These algorithms exploit friendship relationships or common interests between device owners or users. 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.

In future work, we will investigate more social utility function to improve the selection of the offloadees. We will combine both friendship and interests and tune dynamically the offloading percentage with the connectivity strength.

ACKNOWLEDGMENT

This publication was made possible by NPRP grant # 5-648-2-264 from the Qatar National Research Fund, a member of Qatar Foundation. The statements made herein are solely the responsibility of the authors.

REFERENCES

[1] 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.

[2] A. Fahim, A. Mtibaa, and K. A. Harras, "Making the case for computational offloading in mobile device clouds," in *Proceedings of the 19th Annual International Conference on Mobile Computing & Networking*, ser. MobiCom '13. New York, NY, USA: ACM, 2013, pp. 203–205. [Online]. Available: http://doi.acm.org/10.1145/2500423.2504576

[3] 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

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

[5] A. Mtibaa, K. A. Harras, and A. Fahim, "Towards computational offloading in mobile device clouds," in *Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on*, vol. 1. IEEE, 2013, pp. 331–338.

[6] R. Friedman, A. Kogan, and Y. Krivolapov, "On power and throughput tradeoffs of wifi and bluetooth in smartphones," in *INFOCOM, 2011 Proceedings IEEE*, 2011, pp. 900–908.

[7] 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.

[8] 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.

[9] A. Mtibaa and K. Harras, "Exploiting space syntax for deployable mobile opportunistic networking." 2013.

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

[11] 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

[12] 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.

[13] 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

[14] 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.

[15] A. Mtibaa, A. Fahim, K. A. Harras, and M. H. Ammar, "Towards resource sharing in mobile device clouds: Power balancing across mobile devices," in *Proceedings of the second ACM SIGCOMM workshop on Mobile cloud computing*. ACM, 2013, pp. 51–56.