U.P.B. Sci. Bull., Series C, Vol. 76, Iss. 1, 2014 ISSN 2286 – 3540

OFFLOADING FOR MOBILE DEVICES: A SURVEY

Alexandru-Corneliu OLTEANU[1], Nicolae ȚĂPUŞ[2]

We survey existing research efforts regarding offloading for mobile devices. We propose a General Offloading Model and a Taxonomy for Offloading Concerns. We also identify research directions: the balance between offloading and adaptation, and the opportunity for finding novel offloading mechanisms or conducting design space exploration.

Keywords: offloading, smartphones, cloud computing, taxonomy

1. Introduction

Modern handheld devices, such as smartphones and tablets, offer portability, increased computational power, and communication capabilities. Among their characteristics there are: connectivity, processing capabilities, sensing abilities, pervasiveness, heterogeneity, and limited battery supply. The limited battery supply and processing capabilities, at least in respect to the user demand, are the characteristics that trigger most the interest in offloading research. The connectivity supports the offloading process, while the heterogeneity provides several challenges.

Given the characteristics mobile devices possess, they are an attractive option for users to interact with each other, through social applications, and with their environment, through home automation. The popularity of mobile devices can be seen in many ways. Facebook, who has announced recently their increase to over 1 billion monthly active users, reports that more than a half of their users reach their social network using a mobile device [6].

People use mobile devices daily in activities ranging from entertainment to solving professional tasks. Mobile applications span a vast application-domain, being developed for various purposes, such as gaming, multimedia streaming, travel, communication, etc. Many of these types of applications rely on connectivity and on data stored remotely. Also, many of them make a lot of use of the high computation power of mobile devices. Among this generous application space, there are several types of applications that would benefit from offloading:

- applications that are computational intensive (e.g. Chess)

[1] Teaching Teaching Assistant, Faculty of Automatic Control and Computers, University POLITEHNICA of Bucharest, Romania, e-mail: alexandru.olteanu@cs.pub.ro

[2] Professor, Faculty of Automatic Control and Computers, University POLITEHNICA of Bucharest, Romania, e-mail: nicolae.tapus@cs.pub.ro

- applications that rely on data from server side (e.g. object recognition)
- applications with pipeline-based processing (e.g. image processing)
- applications that already interact with the cloud (e.g. m-commerce)

Although mobile devices are growing in functionality and computing power, we believe the role of more powerful infrastructure, to augment the capabilities of mobiles, will increase, due to limitations in battery performance and power dissipation, due to ever increasing user demands, and due to social interactions among users. As shown in [36], the convergence of mobile computing and cloud-based services is one of the leading ways in which cloud computing will evolve in the near future.

Software that uses the interaction of mobile devices with the cloud is already on the market. However, recent research efforts have also identified the cloudlet as an offloading target [31], emphasizing the trade-off between communication and computation costs. Inspired by this research, we see mobile devices as part of a hierarchy of computing systems people are using today, comprised of wearable devices, handheld devices, cloudlets, and clouds.

We survey existing research efforts regarding offloading for mobile devices. We propose a General Offloading Model and a Taxonomy for Offloading Concerns. We also identify research directions. We emphasize the balance between offloading and adaptation, as an alternative or as a complement to offloading. Moreover, we outline the opportunity for novel offloading mechanisms, like partial and parallel offloading, and the importance of design space exploration.

2. General Offloading Model

In this section, we present a general offloading model that describes a generic offload system. In this chapter we offer an overview of each of the key components involved in offloading and we will detail their functionality through a taxonomy in Section 3.

We divide the components of an offloading system in two planes: components on the client – the mobile device – and components in the environment – either a cloud, a cloudlet, a peer device, or a hybrid environment, as discussed in Section 1. The components are depicted in Fig. 1 and are detailed in the remainder of this section.

Many of the current research efforts focus on thoroughly understanding the application to be offloaded. Therefore, the Application Monitoring part of the offloading system is key in obtaining a beneficial offloading process. The Profiling component is able to assess the way in which the application is functioning through various mechanisms, such as static or dynamic analysis. The information obtained from the Profiling component may be used by the

Partitioning component, which aims to split the application in components of predefined granularities and identify which of them can be offloaded.

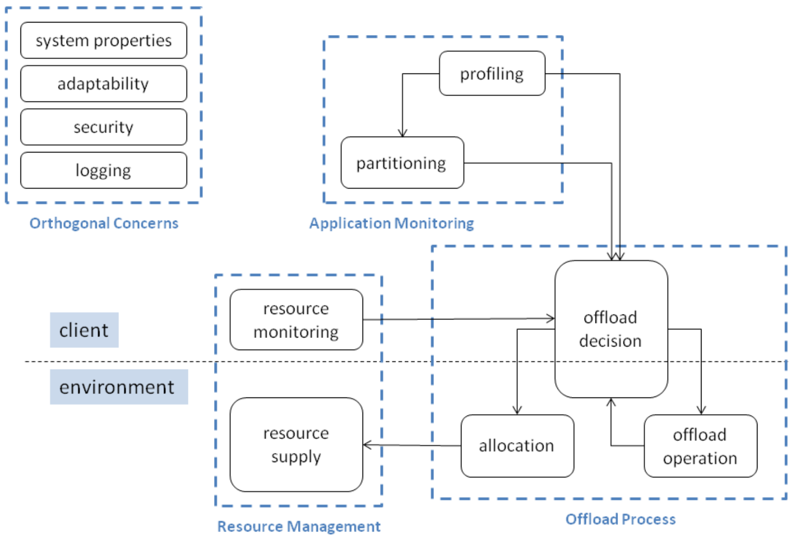

Fig. 1. General offloading model.

There is also a need to assess the resources, both local and remote, that are available for running the application. Thus, the Resource Management component spans both the client and the environment plane. In the client plane, the Resource Monitoring component assesses parameters such as battery level, CPU load, wireless connection quality, and so on. In the environment plane, the Resource Supply component manages the external resources that may be used in offloading, through mechanisms such as discovery and provisioning. Resource discovery is useful in opportunistic approaches, such as cyber foraging, in which the mobile device tries to find available offloading targets in its environment. Resource provisioning is a proactive approach, highly utilized in cloud environments, in which resources are dynamically created to adjust to computing needs.
The Offload Process itself needs to be an iterative process, due to mobility and the changing nature of the execution conditions. For example, the mobile client may switch from WiFi to 3G, or may reach a critical battery level, with consequences on the offloading process. The Offload Decision component receives information from Application Monitoring and Resource Management, to assess the current offloading needs and conditions, and from previous iterations of the Offload Operation, to assess their benefits and defects. Offload Decision can choose:

- *what* to offload, among the sets of partitions offered by the Partitioning component

- *when* to offload, as sometimes it may be not worth offloading at all
- *where* to offload, and instruct the Allocation component to use the appropriate offloading target.

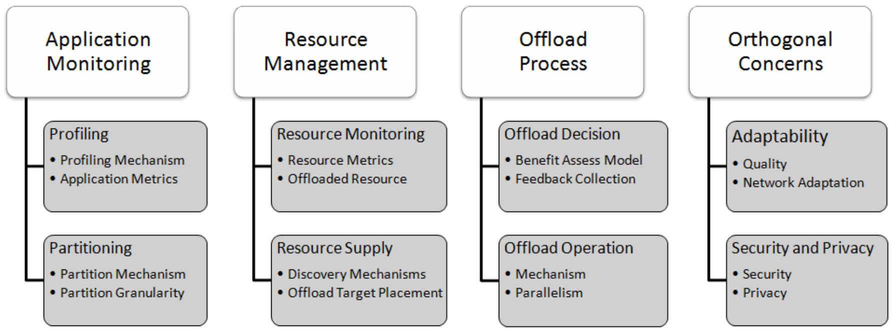

Fig. 2. Our taxonomy for offloading concerns.

Besides the basic offloading process, research efforts also address a number of orthogonal concerns. Some approaches focus on adaptability, e.g. a game may turn o animations if the offloading system is not able to sustain a reasonable frame rate, or the device may switch communication networks through handover. Security concerns derive from offloading to remote resources that usually belong to third-parties. Logging may be used for accountability, billing and monitoring the whole process.

3. Taxonomy for Offloading Concerns

In this Section, we present a novel taxonomy for offloading concerns, structured on the model presented in Section 2. The model identifies four major areas of offloading-related topics: Application Monitoring, Resource Management, Offload Process and Orthogonal Concerns. Each of these areas has a number of key topics of interest. Fig. 2 shows our three-level taxonomy.

A. Application Monitoring

Many research efforts on offloading for mobile devices use only on a couple of applications, thus simplifying the Application Monitoring area. However, there are some that strive for an automated solution, which would work on many types of applications, and thus need to employ complex mechanisms (see Table 1).

A.1. Profiling. Profiling can be done through various mechanisms and using various application metrics.

A.1.1. Profiling Mechanism. Many researchers, like [5][33][2], employ a mix of static and dynamic analysis. Notably, there are a few efforts [17][4][13] that employ online pro ling, which monitors the application behavior while offloading.

Table 1

Mapping Application Monitoring approaches with our taxonomy

	Profiling Mechanism	Application Metrics	Partition Mechanism	Partition Granularity
Eom, 2012 (Snarf) [5]	S,D,P	C,P	C	J
Hong, 2009 [16]	S,D	E,T	M	T
Kemp, 2012 (Cuckoo) [18]	S,P	M,C,T	M	M
Huerta, 2010 [17]	S,O	T,L	M,C	M
Lagerspetz, 2011 [22]	S	E	M	C
Cuervo, 2010 (MAUI) [4]	D,O,P	P	C	M
Ou, 2007 [28]	S	M,C,P,F	M	C
Gu, 2004 [13]	O	M,P,L,F,D	G	C
Zhang, 2010 [37]	S,P	M,C,E	M,C	C
Satyanarayanan, 2009 [31]	P	-	M	V
Verbelen, 2012 (AIOLOS) [32]	S,D,O	C,E,D	G	V,T

Legend: Profling Mechanism: S=static analysis, D=dynamic ofline profling, O=dynamic online pro ling, P=programmer input (a priori); Application Metrics: M=memory usage, C=CPU time, T=operational time, E=energy, P=portability, F=access frequency, L=location, D=input/output data; Partition Mechanism: M=manual partition, C=code annotation, S=chunk splitting, G=graph processing techniques; Partition Granularity: V=vm, J=job, P=process, T=thread, C=component/object, M=method.

A.1.2. Application Metrics. Most of the research efforts on this topic measure, per component, performance related metrics, such as CPU usage, memory usage, energy consumption and operational time. Some works [28][13][32] also employ statistical metrics such as the number of invocations a component has and the amount of data it communicates, which have a role in graph specific algorithms used in partitioning. Qualitative metrics such as portability and location of the component are also collected for the offloading decision [33][11].

A.2. Partitioning serves as starting point in the offload process. The application is split in components of various granularities and an offload decision is made about each of them. The decision is taken by modeling the components as a graph, quantifying each node and each edge with specific metrics and applying graph specific algorithms to decide which components are going to be offloaded and which are going to be computed locally.

A.2.1. Partition Mechanism. A number of algorithms can be used when partitioning, such as clustering. Some solutions rely on code annotation from the developer, while others strive for an automatic approach.

A.2.2. Partition Granularity. Partitioning the application can be performed at various levels. Components can differ in size from an approach to another, ranging from a method or piece of code, to full threads or processes, and even to entire virtual machines.

B. Resource Management

In offloading works that focus on resource management (see Table 2), it can be noted that opportunistic or cyber foraging approaches are often based on collocation, as mobile devices will use computational power available in their vicinity. Only a few deal with resource supply, as many researchers prefer to statically de ne the remote resources in their experiments.

Table 2

Mapping Resource Management approaches with our taxonomy

	Resource Metrics	Offloaded Resource	Discovery Mechanisms	Offload Target Placement
Ferber, 2012 [7]	T,X	C	S	CC
Kemp, 2011 [19]	E,Bat	N,C	S	CC, HC
Hassan, 2011 (map-reduce) [14]	C,E,X	C,D	S	CC,RC
Huerta, 2010 [17]	$,C,M	C,M	C	P
Balan, 2007 [1]	C,M,B,Bat	C	S	RC
Yan, 2010 [35]	X,$	C	B	CC
Ou, 2007 [28]	M, C, B	C	S	RC, HC, HS
Zhang, 2010 [37]	C, M, Bat, N	C, S, N	C	CC, CS
Satyanarayanan, 2009 [31]	C,M,X	C	B	CL
Flores, 2013 [8]	C,B	C,D	S	CC,CS
Marin, 2013 [23]	C,M,B,Bat	C	B	CC

Legend: Resource Metrics: C=CPU load, M=memory load, N=network latency, B=bandwidth usage, T=total execution time, X=interaction delay, E=energy consumption, Bat=battery level, $=cost; Offloaded Resource: C=computation, N=communication, D=data/content, M=memory, S=storage; Discovery Mechanisms: S=static, C=collocation, B=broker; Offload Target Placement: CC=cloud computing, CS=cloud storage, CD=CDN, CL=cloudlet, RC=residential computers, HC=home computer, HS=home server, HM=home mobile devices, P=peers.

B.1. Resource Monitoring is present in most of the works about offloading for mobile devices. All of them select one or a combination of resources to be offloaded and many also employ a mechanism to monitor existing resources on the device and in the environment.

B.1.1. Offloaded Resource. Most of the solutions we studied focused on methods to offload computation. However, there are some notable variations. Huerta et al. [17] offload processing that has high memory requirements and thus is not suitable for mobile devices. Hassan [14] and Flores cite ores2013adaptive perform offloading to use the data already present in the cloud rather than bringing it to the mobile device. Communication offloading in the form of push notifications, shown in research works like [19], has already been implemented as a commercial solution by many mobile technology producers, like Google [12].

B.1.2. Resource Metrics mirror the application metrics presented in A.1.2. and are used when deciding whether the mobile device has enough resources to perform the operations locally or not. Most of the approaches monitor some form of computational load [14][17]and battery level on the device [1][37][23]. Many also measure the impact of network transfers as network latency [37], bandwidth usage [1][28][23] or interaction delay [7][14][35][30].Huerta [17] and Yan [35] also take into account the monetary cost of using remote resources.

B.2. Resource Supply is a more niche topic, as many offloading approaches are only tested on a statically defined resource pool, on various levels of target placement, such as laptops [1] or virtual machines in commercial infrastructures like Amazon Web Services [7][18][14]. However, some approaches deal with resource supply in the form of discovery or provisioning, where resources can as well be located on various levels of target placement.

B.2.1. Offload Target Placement. Kemp et al [19] investigate offloading communication to a single push server. They note that one may want to use elastic cloud resources when load gets too high, but they admit that Cuckoo, their framework, does not support migration of code between servers. Kumar et al [21] highlight the benefits of using the cloud storage, that has beneficial effects on the amount of transferred data and thus on the performance of the offloading process

B.2.2. Discovery Mechanisms. Zhang et al [37] focus on two scenarios that require massive resources in constrained locations that encourage the use of devices in the vicinity, by leveraging collocation. In [31], the authors describe the Kimberley Control Manager, that supports browsing and publishing services using the Avahi mechanism in Linux. This module acts as a broker in identifying and maintaining connections with the remote resources. In the work presented by Marin et al [23], a component named Cloud Receiver also acts as a broker in deciding which cloud resource to use.

C. Offload Process

Table 3 summarizes efforts regarding the Offload Process in existing scientific literature.

C.1. Offload Decision gathers some of the most diverse ideas in offloading for mobile devices, depending on the benefit assessment. The approaches differ in the way they assess benefits, how they collect feedback from previous iterations of the offloading process and how they take into account context.

C.1.1. Benefit Assessment. Most researchers take into account the performance of the mobile device with and without offloading, expressed as running times. Some also optimize the energy consumption, like [21][22][37], or the monetary costs, like [7][35][37]. Satyanarayanan[31] and Verbelen[33] also refer to the quality of the result when offloading, like better image resolution.

C.1.2. Feedback Collection. Most of the solutions based on remote code execution discuss methods to integrate the results obtained on the remote resources back into the mobile device. Some solutions, based on virtual machines, like [31] and [32] only discuss state integration. Notably, some approaches, like [14][35][28] get into the complexities of handling failures in remote processing.

Table 3

Mapping Offload Process approaches with our taxonomy

	Benefit Assessment	Feedback Collection	Mechanism	Parallelism
Ferber, 2012 [7]	P,C	N	R,S	S
Kumar, 2010 [21]	P,E	S	M,R	S,P
Hassan, 2011 (map-reduce) [14]	P	R,F	MR	P
Lagerspetz, 2011 [22]	P,E	R	R,S	P
Cuervo, 2010 (MAUI) [4]	P	S	R	S
Yan, 2010 [35]	P,C	R,F	S	P
Ou, 2007 [28]	P	R,F	R	S
Zhang, 2010 [37]	P,C,E	S,R	M	P
Satyanarayanan, 2009 [31]	P,Q	S	C	P
Verbelen, 2012 [33]	Q	R	R	S
Verbelen, 2012 (AIOLOS) [32]	P	S	M	S
Flores, 2013 [8]	P	R	MR	P

Legend: Benefit Assessment: P=performance, C=cost, E=energy consumption, Q=quality; Feedback Collection: S=state migration, R=result integration, F=handling failure, N=none; Mechanism: J=job partition, C=cloning / replication, M=migration (class/object/process/thread), R=remote execution (opportunistic cyber foraging, data staging), MR=map-reduce, S=Service; Parallelism: S=sequential, C=concurrent, P=parallel.

C.2. Offload Operation. The offload mechanism is usually one of the focal points in most of the papers on mobile offloading, as there are many variations and some correlations with the granularity of the application. Parallel offloading is considered by some researchers, as a method to increase performance, but its benefits are application specific and bounded by the additional complexity that it brings. Processing division is at the core of the offloading process, which usually splits the processing spatially, and, less often, temporally. Data division is rarely considered.

C.2.1. Mechanism Lagerspetz et al [22] focus on le indexing among devices and between devices and the cloud. The offloading process is described as remote execution among devices and as a service from the cloud. Zhang et al [37] propose an application model based on migrating application components named weblets. In [31], the authors use entire VM migration and dynamic VM synthesis to replicate the work.

C.2.2. Parallelism. In [35], the authors propose a solution that uses the Amazon Mechanical Turk, a service where tens of thousands of people are actively working on simple tasks for monetary rewards, to perform image search. The sollution is parallel, in a sense that each person matches the query to a set of photos. In [37], the authors conduct offloading on an image processing application. A weblet pool is created on the cloud, and images are processed in parallel by pool members.

D. Orthogonal Concerns

Studying papers that deal with Orthogonal Concerns while offloading (see Tablet 4), it can be noted that very few research efforts address all types of orthogonal concerns, but many address at least one.

Table 4

Mapping Orthogonal Concerns with our taxonomy

Paper	Adaptability	Security & Privacy
Kumar, 2010 [21]	-	S,P
Kemp, 2011 [19]	-	S,P
Eom, 2012 (Snarf) [5]	-	S
Kemp, 2012 (Cuckoo) [18]	Q	S,P
Klein, 2010 [20]	H	-
Wang, 2010 [34]	Q	-
Hoang, 2012 [15]	A	-

Legend: Adaptability: H=Handover /Network Adaptation, Q=Quality

Adaptation, Security; Privacy and Logging: S=security, P=privacy.

D.2. Adaptability. In [20], the authors describe Heterogeneous Access Management schemes in the context of Mobile Cloud Computing that is performing network handover, based on location awareness, network load, user movement predictions, and so on. In [34] the authors propose a rendering adaptation technique with a focus on user experience.

D.3. Security & Privacy. Kumar et al [21] discuss several privacy concerns related to using cloud resources, giving examples such as bugs, third-party vendors and location tracking. Eom et al [5] propose to use SocialVPN, a tunneling technique through Virtual Private Networks that has a double benefit: better security and virtualization.

4. Research Directions

Improvements in mobile applications derived from offloading are increasingly reaching the users. For example, the communication offloading technique presented by Kemp [19] is also implemented commercially by major mobile service providers, like in [12]. Also on the market, applications such as Shazam use re-mote processing as a basic way of functioning. However, many research efforts still face a number of challenges until they can be implemented for the general public.

Application Monitoring: most of the research focuses on increasingly automatic ways of partitioning applications at an operations level. We believe that application monitoring can also be understood from a workload perspective, with results such as load predictions, that in turn can lead to better resource provisioning and smarter offloading systems. For details we refer to [26] and [27].

Resource Management: resource discovery and provisioning is hardly considered. Many approaches refer to opportunistic offloading [3], resource scavenging, cyber foraging [1], and so on. However, cloud providers, with expertise in discovery and provisioning, can efficiently offer their resources to the mobile software market, a market with hundreds of millions of users. [25]

Offload Process: research can be made on partial and parallel offloading, as well as exploiting the region of interest. Moreover, experimental setup can be better tuned for real-life applications, as most of the current efforts propose experiments with few clients and sometimes with laptops instead of mobiles. We refer to [25] for details on a novel Exploratory Space of offloading concerns.

As an example of our work on offloading, we experiment with several offloading techniques on OpenTTD, a popular open-source simulation game. We modify it for instrumentation, repeatability and offloading capabilities. Fig. 3 shows two of the metrics we collect, CPU load and in-game time, throughout 10 minute gaming sessions, using a mobile device as client and a laptop as cloudlet resource provider.

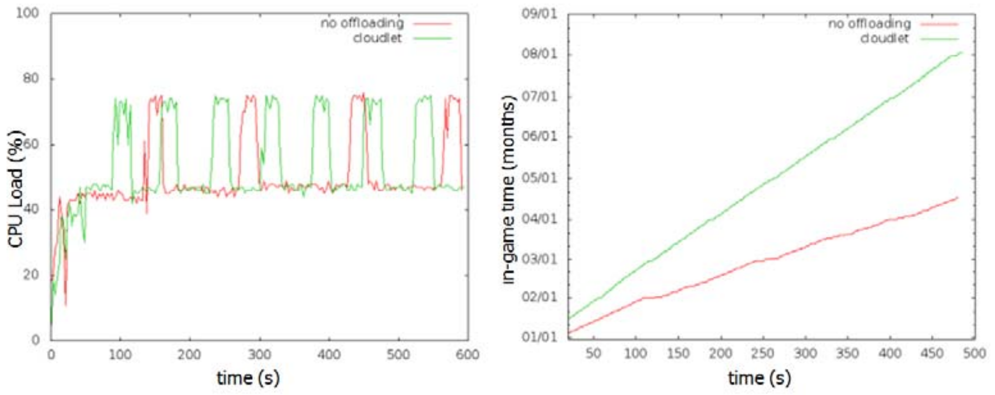

Fig. 3. Comparison of local running of AIs (red) with offloaded running of AIs (green)

Both charts indicate that, without offloading, the game slows down, to compen-sate for the lack of processing power of the client device. As indicated by the values in the right-hand chart, as well as by the number of spikes in the left-hand chart, in a 10 minute gaming session, the offloaded version covers almost 8 in-game months, while the local version covers only 4 in-game months.

Orthogonal Concerns: our taxonomy identifies at item D.2. the adaptability of the system as a key orthogonal concern. Adaptation can be performed in terms of quality of the result, network access or admission control. It is often a good companion to offloading and sometimes even a better alternative. Finding a good balance between offloading and adaptation can be a novel approach for performance optimization.

For details, we refer to our application domain exploration, published in several papers. In [24] and [29] we investigate Communication Adaptation and Offloading for distributed applications for the mobile device and custom hardware extensions, such as sensor devices and home automation networks. In [9] and [10] we investigate Computation Adaptation and Offloading for loop-based applications, with a focus on video processing – e.g. augmented reality application – and video rendering applications – e.g. popular simulation game.

5. Conclusion

In this paper, we survey existing research efforts regarding offloading for mobile devices and, to structure this vast material, we propose a primer on offloading, a General Offloading Model and a Taxonomy for Offloading Concerns. We also identify research directions: we emphasize the balance between offloading and adaptation, and outline the opportunity for novel offloading mechanisms, such as parallel offloading and partial offloading. We believe focusing on a specific appli-cation domain can o er better insights on application monitoring and offloading techniques, while maintaining a high level of applicability. We refer to our studies on online social applications in terms of statistically modeling the workloads [26] and conducting offloading design space exploration [25].

Acknowledgments

The authors would like to thank Alexandru Iosup. This work is funded by the Sectoral Operational Programme Human Resources Development 2007-2013 of the Romanian Ministry of Labour, Family, and Social Protection through POSDRU/107/1.5/S/76909.

REFERENCES

[1] R. K. Balan, D. Gergle, M. Satyanarayanan, and J. Herbsleb. Simplifying cyber foraging for mobile devices. MobiSys '07, pages 272-285. ACM, 2007.

[2] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti. Clonecloud: elastic execution between mobile device and cloud. CCS '07, pages 301-314. ACM, 2011.

[3] R. I. Ciobanu, C. Dobre, and V. Cristea. Sprint: Social prediction-based opportunistic routing. 2013.

[4] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl. Maui: making smartphones last longer with code offload. MobiSys '10, p 49-62. ACM, 2010.

[5] H. Eom, P. St Juste, R. Figueiredo, O. Tickoo, R. Illikkal, and R. Iyer. Snarf: a social networking-inspired accelerator remoting framework. MCC '12, pages 29-34. ACM, 2012.

[6] Facebook reaches one billion users, 2012. online: http://bit.ly/QSHk1P, access May 2013.

[7] M. Ferber, T. Rauber, M. Torres, and T. Holvoet. Resource allocation for cloud-assisted mobile applications. IEEE Cloud '12, pages 400 -407, june 2012.

[8] H. R. Flores and S. Srirama. Adaptive code offloading for mobile cloud applications: exploiting fuzzy sets and evidence-based learning. ACM MCS'13, pages 9-16. ACM, 2013.

[9] A. Fasui, A. C. Olteanu, and N. Tapus. Fault tolerant surveillance system based on a network of mobile devices. In RoEduNet, 2013. Available: http://bit.ly/WCOr0K.

[10] A. Gherghina, A. C. Olteanu, and N. Tapus. A marker-based augmented reality system for mobile devices. In RoEduNet, 2013. Available: http://bit.ly/XXPLYV.

[11] I. Giurgiu, O. Riva, D. Juric, I. Krivulev, and G. Alonso. Calling the cloud: enabling mobile phones as interfaces to cloud applications. In Middleware 2009. Springer, 2009. p.83-102.

[12] Google, Inc. Android Cloud to Device Messaging Framework, 2012.

[13] X. Gu, K. Nahrstedt, A. Messer, I. Greenberg, and D. Milojicic. Adaptive offloading for pervasive computing. Pervasive Computing, IEEE, 3(3):66-73, 2004.

[14] M. A. Hassan and S. Chen. Mobile mapreduce: Minimizing response time of computing intensive mobile applications. In MobiCASE, pages 41-59, 2011.

[15] D. T. Hoang, D. Niyato, and P. Wang. Optimal admission control policy for mobile cloud computing hotspot with cloudlet. WCNC '12, pages 3145-3149. IEEE, 2012.

[16] Y.-J. Hong, K. Kumar, and Y.-H. Lu. volume ISCAS '09, pages 1673 -1676, may 2009.

[17] G. Huerta-Canepa and D. Lee. A virtual cloud computing provider for mobile devices. MCS '10, pages 6:1-6:5, New York, NY, USA, 2010. ACM.

[18] R. Kemp, N. Palmer, T. Kielmann, and H. Bal. Cuckoo: A computation offloading framework for smartphones. volume 76 of MCAS'12, pages 59-79. Springer, 2012.

[19] R. Kemp, N. Palmer, T. Kielmann, and H. Bal. Energy e cient information monitoring applications on smartphones through communication offloading. Mobicase, 2012.

[20] A. Klein, C. Mannweiler, J. Schneider, and H. D. Schotten. Access schemes for mobile cloud computing. MDM'10, pages 387-392. IEEE, 2010.

[21] K. Kumar and Y.-H. Lu. Cloud computing for mobile users: Can offloading computation save energy? IEEE Computer, 43(4):51-56, 2010.

[22] E. Lagerspetz and S. Tarkoma. Mobile search and the cloud: The benefits of offloading. PERCOM'11, pages 117 -122, march 2011.

[23] R.-C. Marin and C. Dobre. Reaching for the clouds: contextually enhancing smartphones for energy efficiency. 2013.

[24] A. Olteanu, G. D. Oprina, N. T~apus, and S. Zeisberg. Enabling mobile devices for home automation using zigbee. In CSCS, pages 189-195. IEEE, 2013.

[25] A. C. Olteanu, N. Tapus, and A. Iosup. Extending the capabilities of mobile devices for online social applications through cloud offloading. In CCGRID, pages 160-163, 2013.

[26] A. C. Olteanu, A. Iosup, and N. T~apus. Towards a workload model for online social applications: Icpe 2013 work-in-progress paper. In ICPE, pages 319-322. ACM, 2013.

[27] A. C. Olteanu, A. Iosup, N. Tapus, and F. Kuipers. A workload evolution model for online social games. Internet Computing. submitted.
[28] S. Ou, K. Yang, and J. Zhang. An e ective offloading middleware for pervasive services on mobile devices. Pervasive and Mobile Computing, 3(4):362-385, 2007.
[29] D. O. Rizea, D. S. Tudose, A. C. Olteanu, and N. Tapus. Adaptive query algorithm for location oriented applications. In RoEduNet, 2013. Available: http://bit.ly/WGXRbg.
[30] M. Satyanarayanan. Pervasive computing: Vision and challenges. Personal Communications, IEEE, 8(4):10-17, 2001.
[31] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies. The case for vm-based cloudlets in mobile computing. IEEE Perv. Comp., 2009.
[32] T. Verbelen, et al. Aiolos: Middleware for improving mobile application performance through cyber foraging. Journal of Systems and Software, 85(11):2629-2639, 2012.
[33] T. Verbelen, P. Simoens, F. De Turck, and B. Dhoedt. Cloudlets: Bringing the cloud to the mobile user. In MCCS, pages 29-36. ACM, 2012.
[34] S. Wang and S. Dey. Rendering adaptation to address communication and computation constraints in cloud mobile gaming. In GLOBECOM '10, IEEE, pages 1-6. IEEE, 2010.
[35] T. Yan, V. Kumar, and D. Ganesan. Crowdsearch: exploiting crowds for accurate real-time image search on mobile phones. In MSAS, pages 77-90. ACM, 2010.
[36] ZDNet. 10 ways cloud computing will change in 2013. online: http://zd.net/SA6d1F, last access Jan 2013.
[37] X. Zhang, S. Jeong, A. Kunjithapatham, and S. Gibbs. Towards an elastic application model for augmenting computing capabilities of mobile platforms. In Mobile wireless middleware, operating systems, and applications, pages 161-174. Springer, 2010.