Architecture Strategies for Cyber-Foraging: Preliminary Results from a Systematic Literature Review

Grace A. Lewis[1,2], Patricia Lago[2], and Giuseppe Procaccianti[2]

[1] Carnegie Mellon Software Engineering Institute, USA
[2] VU University Amsterdam, The Netherlands
glewis@sei.cmu.edu, p.lago@vu.nl, g.procaccianti@vu.nl

Abstract. Mobile devices have become for many the preferred way of interacting with the Internet, social media and the enterprise. However, mobile devices still do not have the computing power and battery life that will allow them to perform effectively over long periods of time or for executing applications that require extensive communication or computation, or low latency. Cyber-foraging is a technique to enable mobile devices to extend their computing power and storage by offloading computation or data to more powerful servers located in the cloud or in single-hop proximity. This paper presents the preliminary results of a systematic literature review (SLR) on architectures that support cyber-foraging. The preliminary results show that this is an area with many opportunities for research that will enable cyber-foraging solutions to become widely adopted as a way to support the mobile applications of the present and the future.

1 Introduction

Mobile Cloud Computing (MCC) refers to the combination of mobile devices and cloud computing in which cloud resources perform computing-intensive tasks and store massive amounts of data. Increased mobile device capabilities, combined with better network coverage and speeds, have enabled MCC such that mobile devices have become for many the preferred form for interacting with the Internet, social media, and the enterprise. However, mobile devices still offer less computational power than conventional desktop or server computers, and limited battery life remains a problem especially for computation- and communication-intensive applications.

Cyber-foraging is an area of work within MCC that leverages external resources (i.e., cloud servers or local servers called surrogates) to augment the computation and storage capabilities of resource-limited mobile devices while extending their battery life. There are two main forms of cyber-foraging. One is computation offload, which is the offload of expensive computation in order to extend battery life and increase computational capability. The second is data staging to improve data transfers between mobile devices and the cloud by temporarily staging data in transit.

The goal of this paper is to present the preliminary results of a Systematic Literature Review (SLR) to discover software architecture solutions that support cyber-foraging and set the stage for future and necessary research in this area. Section 2 presents a very brief summary of the SLR elements. Section 3 presents the analysis of the identified primary studies using a categorization of architecture decisions that are relevant for cyber-foraging systems. A summary of the observations and findings from the primary studies is presented in Section 4. Section 5 presents related work. Finally, Section 6 presents conclusions and the next steps in our research.

2 Research Method

To identify work related to architectures for cyber-foraging an SLR was conducted following the guidelines proposed in [1] and [2]. The research question was stated as "What software architecture and design strategies for cyber-foraging from mobile devices can be identified in the literature?" The main data source was Google Scholar and snowballing was used to complement the set of primary studies. Due to page limitations, the details related to inclusion and exclusion criteria, search string used, search string validation, results of the multiple search rounds, and threats to validity can be found at `http://www.cs.vu.nl/~patricia/Patricia_Lago/Shared_files/SLR-ArchCyberForaging.pdf`. A set of 57 primary studies was identified [3] Table 1 shows the computation offload systems found in the primary studies and Table 2 shows the data staging systems.

3 Analysis of Primary Studies

Defining an architecture for a system that uses cyber-foraging to enhance the computing power of mobile devices requires making decisions on where, when and what to offload, from the perspective of the mobile device. The systems from the primary studies were analyzed to obtain the answers to these questions.

3.1 Where to Offload

In cyber-foraging, computation or data is offloaded to resources with greater computing power. These resources are located in either single-hop or multi-hop proximity of mobile devices.

Most of the systems in the studies (16/60 or 27%) offload to only *Proximate Disconnected* resources, which are surrogates located in single-hop proximity of the mobile device that can operate without being connected to a cloud resource.

[3] The total of primary studies is 57 but the total of systems analyzed is 52 for computation offload and 8 for data staging for a total of 60 systems because two of the computation offload studies present two different systems and one study presents systems for both computation offload and data staging.

This is expected because of the advantages of lower latency and battery consumption that come from using WiFi or short-range radio instead of broadband wireless (e.g., 3G/4G) [3]. These systems therefore assume that the surrogate can function stand-alone and offload computation is pre-provisioned (i.e., at system deployment time) or provisioned at runtime from the mobile devices themselves. However, many of these systems could be adapted to work with remote cloud servers or any addressable offload target but would lose the advantage of lower latency due to proximity.

Table 1: Computation Offload Systems in Primary Studies

System	Where			When		What													
						Granularity					Payload								
	Prox .Disconnected	Prox. Connected	Remote	Runtime Decision	Always Offload	Process	Function	Component	Service	Application	Computation	Partitioning Algo.	Parameters	Application State	Device Context	Source Location	Setup Instructions	Continuous Data
mHealthMon [4]			X	X					X				X					
Mobile Agents [5]			X	X				X					X					
Clone-to-Clone (C2C) [6]	X			X			X						X					
Chroma [7]	X			X			X						X					
Collaborative Applications [8]	X			X			X						X	X				
Computation and Compilation Offload [9]	X			X			X						X					
Cloud Media Services [10]			X		X			X							X			
Roam [11]		X	X	X				X					X	X		X		
CloneCloud [12]	X				X	X								X				
MAUI [13]	X		X	X			X						X					
Kahawai [13]	X				X					X								X
HPC-as-a-Service [14]	X		X		X				X				X					
OpenCL-Enabled Kernels [15]	X		X	X				X			X		X					
Real Options Analysis [16]	X		X	X			X						X					
3DMA [17]			X	X				X					X					
Spectra [18]	X			X			X						X					
AlfredO [19]			X	X				X				X	X					
Collective Surrogates [20]		X	X		X					X							X	
Grid-Enhanced Mobile Devices [21]		X	X		X			X					X					
Cloudlets [22]	X				X					X	X		X					
Virtual Phone [23]			X		X			X						X				
Single-Server Offloading [24]	X			X			X						X					
Cloud Operating System [24]		X			X			X			X		X					
Android Extensions [25]			X		X			X					X					
ThinAV [26]		X	X		X				X				X					

Continued on next page

Table 1 – *Continued from previous page*

System	Where			When		What												
						Granularity					Payload							
	Prox .Disconnected	Prox. Connected	Remote	Runtime Decision	Always Offload	Process	Function	Component	Service	Application	Computation	Partitioning Algo.	Parameters	Application State	Device Context	Source Location	Setup Instructions	Continuous Data
Cuckoo [27]	X		X	X				X			X		X					
ThinkAir [28]			X	X			X				X		X					
MACS [29]	X		X	X				X			X		X					
Scavenger [30]	X			X				X			X		X					
AMCO [31]	X		X	X				X					X	X				
MCo [32]			X		X			X			X		X					
PowerSense [33]	X			X					X				X					
AIDE [34]	X		X	X				X					X					
Application Virtualization [35]	X				X					X	X		X					
PARM [36]	X			X				X					X					
Resource Furnishing System [37]		X	X		X					X								X
Cloud Personal Assistant [38]		X	X		X				X				X					
SOME [39]			X		X		X						X					
SmartVirtCloud [40]		X		X			X				X		X					
Odessa [41]	X		X	X				X					X					
Smartphone-Based Social Sensing [42]	X		X	X				X					X					
MAPCloud [43]		X	X	X					X								X	
VM-Based Cloudlets [44]	X				X					X	X		X					
IC-Cloud [45]	X		X	X			X						X					
SPADE [46]			X		X			X					X					
Slingshot [47]		X			X					X			X					
Offloading Toolkit and Service [48]	X		X	X				X			X		X					
Mobile Data Stream Application Framework [49]			X	X				X				X	X					
Heterogeneous Auto-Offloading Framework [50]			X	X				X					X					
Weblets [51]			X	X				X					X					
DPartner [52]	X		X	X				X					X					
Elastic HTML5 [53]	X		X	X				X					X			X		

The second largest set of systems in the studies (15/60 or 25%) offloads to *Remote* resources, such as an enterprise cloud or data center. However, unless connectivity to an enterprise cloud is necessary for the system to function, these systems could also offload to proximate connected or disconnected nodes.

Tied for the second largest set of systems in the studies (also 15/60 or 25%) are those that offload to *Remote* or *Proximate Disconnected* resources. In gen-

Table 2. Data Staging Systems in Primary Studies

System	Where			When		What							
						Data Type				Data Operations			
	Prox .Disconnected	Prox. Connected	Remote	Runtime Decision	Always Offload	Data Updates	Application Data	Data Files	Field-Collected Data	Pre-Fetching	In-Bound Processing	Out-Bound Processing	Storage
Edge Proxy [54]		X			X	X					X		
Mobile Information Access Architecture for Occasionally-Connected Computing [55]	X				X		X			X			
Trusted and Unmanaged Data Staging Surrogates [56]	X				X			X		X			
Android Extensions [25]			X		X		X						X
Telemedik [57]	X		X		X		X				X		
Feel the World [58]	X	X	X		X				X				X
Large-Scale Mobile Crowdsensing [59]		X			X				X			X	
Sonora [60]	X	X	X		X				X			X	

eral, these systems have offload targets that can function stand-alone and are accessible over an IP network, whether local or remote.

The next set of systems (7/60 or 12%) offloads to *Remote* or *Proximate Connected* resources, which are surrogates located in single-hop proximity of the mobile device that need to be connected at runtime to a cloud resource. The offload targets in these systems need access to a cloud resource in order to operate properly, whether to obtain the code to be offloaded, access application data, or offload computation or data to other cloud resources (i.e., surrogate acts as an intermediary).

Finally, five out of 60 systems (5/60 or 8%) offload to only *Proximate Connected* resources, and there are two data staging studies that can offload to all three options (2/60 or 3%).

Most systems in the studies offload to a single known surrogate or cloud resource. The reason is that the focus of the studies is on demonstrating the validity or efficiency of portions of the architecture, such as optimization engines or partitioning algorithms, and not the operation of the full system. Some systems include a component in the architecture to discover and select offload targets based on (1) offload target broadcast, (2) a cloud directory service, (3) surrogate managers that manage available surrogates, (4) local offload target lists, or (5) an application or service directory.

3.2 When to Offload

In general, offloading is beneficial when large amounts of computation are needed with relatively small amounts of communication [61].

For most of the systems in the studies (33/60 or 55%) offloading is a *Runtime Decision*. The majority of these systems perform a runtime calculation (often called a utility function) to determine whether it is better to execute locally or to offload computation by comparing predicted local execution cost against predicted remote execution cost. Local execution cost typically takes into consideration the energy consumed by local execution as well as the local execution time. Remote execution cost typically considers the energy consumed by communication based on payload size and network conditions, the communication time based on payload size and network conditions, and remote execution time.

The systems that perform runtime calculations require developer input or static profiling to obtain the initial values or models that are used in the calculation, such as required compute cycles, payload size based on input and output parameters, and required energy for execution and communication. Other parameters such as current network conditions or load of the mobile device and offload target are obtained at runtime. In addition, some systems use runtime profiling to collect data at runtime to adjust the initial values. The goal is to obtain more realistic values based on actual execution data.

The rest of the systems in the studies (27/60 or 45%) *Always Offload* computation or data. For computation offload systems, the parts of the system that are considered computation-intensive, or that simply cannot run on a mobile device, are pre-determined and executed on offload targets. All the data staging systems fall in this category, which is expected, because by definition the idea is for the mobile device to send and receive data to and from an enterprise cloud, either directly or via a surrogate. The decision-making process is not whether it is efficient or not to stage data but when is the right time to do so.

3.3 What to Offload

What to offload involves two architecture decisions, but these are different for computation offload and data staging systems.

Computation Offload Systems. For computation offload, one decision has to do with the *Granularity* of the computation that is offloaded to the surrogate or cloud resource and another has to do with the *Payload* that is sent from the client to the surrogate or cloud resource in order to execute the offloaded computation. Although these seem like low-level decisions, they have architecture implications because they determine the components that are needed on the client and the offload target.

All the systems in the studies have an *offload client* that runs on the mobile device and an *offload server* that runs on the offload target to coordinate the offload operation. The majority of the systems are designed such that the applications at runtime are not aware that computation is being offloaded. What changes between systems based on granularity are the development, build and runtime dependencies between the offload client and target, as well as the amount of state synchronization to guarantee the correct execution of applications.

For *Granulairity*, most systems offload at the *Component, Class, Module, or Task* level (27/52 or 52%). The type of element that is offloaded varies greatly between systems, but in general they are software elements that execute inside specific containers or runtime environments such as Java Virtual Machines (JVMs), OGSi platforms, or custom-built environments that enable migration between local and remote execution. The advantage of offloading at this level of granularity is that for the most part these are self-contained elements, meaning that they store their own state. Once an element is offloaded there is no need to synchronize state with the local device unless the execution is returning to the local device. However, except for the systems that rely on more standard environments, such as JVMs and OGSi platforms, there are very tight dependencies between the mobile execution environment and the execution environment on the offload target, which creates limitations in terms of programming languages and increases the effort required for application reuse because of the need to use specific libraries and constructs to enable computation offload.

The second largest set of systems offloads *Functions, Methods, or Operations* (11/52 or 21%). In many of these systems, developers manually mark the functions that they consider offloadable. In addition to the same types of constraints and requirements for applications and offload targets outlined for the first set of systems, the challenge for these types of system is guaranteeing fidelity of results, which means that executing locally or remotely should produce the same results. Functions, methods and operations are part of a larger programming constructs such as classes or programs that maintain state at runtime, typically expressed as class attributes or global variables. This means that the system has to synchronize state such that it is the same locally and remotely, either periodically or sending it as an additional input/output of the offload operation.

Systems that offload full *Applications, Programs or Servers* of a client/server application represent the third largest set of systems in the studies (7/52 or 13%). The advantage of offloading at this level of granularity is that execution environments are much more generic, such as virtual machines or application servers. This also increases application reuse because servers do not have to be adapted to run on mobile devices. Clients are very thin and perform the functionality that cannot be offloaded, such as user interface and sensor operations. However, the rest of the computation is always offloaded, regardless of whether it would be more efficient or not to be executed on the mobile device.

The fourth largest set of systems in the studies offload *Services* (6/52 or 12%). Services in these studies are coarse-grained capabilities accessed via standardize interfaces that have been identified by system developers as computation-intensive. These systems do not have the requirements or constraints of the systems that offload functions or components because by definition services are self-contained. Once a decision is made to offload, the service is invoked and the system either waits for a reply or receives the reply when it is ready.

Finally, there is one system that offloads at the process level (1/52 or 2%). In this system the mobile device is fully cloned inside a VM running on the offload target. When the system encounters a computation block that is marked for offload, the process enters into a sleep state and process state is transferred from the mobile device to the clone VM. The clone VM integrates the process state, executes the computation block from beginning to end, and then transfers its process state back to the mobile device. The mobile device reintegrates the process state and wakes up the sleeping process to continue its execution. This system allows very fine-grained control of what portions of an application to offload, but requires a very stable network connection to support state synchronization.

Concerning *Payload*, for the majority of the systems the payload is the *Invocation Parameters* to execute the remote computation (27/52 or 52%). All these systems assume that the offloaded computation already exists on the offload target, which leads to a small payload that simply depends on the size of the parameter data types. However, these systems completely rely on the existence and currency of the offloaded computation on the offload target, which in turn would require more complex deployment processes.

For the next largest set of systems the payload is *Computation* <u>and</u> *Invocation Parameters* (12/52 or 23%). This means that both the actual computation (code) and its invocation parameters are sent from the mobile device to the offload target.The offload target deploys the computation inside a container or execution environment, executes it directly in a runtime environment, or distributes it to other offload targets for deployment. Once the computation is running, the mobile device sends the invocation parameters for the actual execution.

For the next set of systems the payload is *Application State* (2/52 or 4%). The state of the application on the mobile device is synchronized with the offload target so that the remote computation can be executed with the same state as that on the application running on the mobile device. In both of these systems the execution returns to mobile device and state is synchronized back in the same way.

For a small set of systems the payload is *Setup Instructions* <u>and</u> *Invocation Parameters* (2/52 or 4%). This means that the initial payload is the instructions of how to set up the computation on the offload target. Once the computation is running, the mobile device sends the invocation parameters for the actual execution.

In the next set of systems (2/52 or 4%) the payload is *Continuous Data from Offload Target to Mobile Device*. In Kahawai [13], a system targeted at GPU-intensive applications such as games, the offload target maintains a high-fidelity version of the graphics and a low fidelity version that matches the fidelity of the mobile device. It compares both and sends a compressed video stream of delta frames to the mobile device. The mobile device decompresses the stream and applies the deltas to the frames that it renders locally. In the Resource Furnishing System [37] the interaction with the offload target is done via a VNC client which means that GUI updates are continuously sent from the offload target to mobile devices and applied locally.

In addition to *Invocation Parameters*, two systems offload the *Partitioning Algorithm* that is part of the "When to Offload" decision to determine what computation executes locally and what computation is offloaded (2/52 or 4%).

For two systems the initial payload is local *Application State* so that the mobile device and the offload target can synchronize state before invoking the offloaded computation (2/52 or 4%). Once the computation is running, the mobile device sends the *Invocation Parameters* for the actual execution.

On a smaller scale, for one system the initial payload is the *Device Context* (1/52 or 2%), which in this case is device type, browser type, supported codecs, screen size, network bandwidth, and latency, such that the appropriate media processing components are selected. Once the computation is running, the mobile device sends the *Invocation Parameters* for the actual execution. For one system (1/52 or 2%), the initial payload is the *Source Location*, or where to obtain the computation for installation on the offload target. *Application State* is then transferred from the mobile device to the offload target. Once the computation is running and the state is synchronized, the mobile device sends the *Invocation Parameters* for the actual execution. Finally, for one system, the initial payload is the *Source Location* (URL) of the offloaded computation and then it sends the *Invocation Parameters* for the actual execution (1/52 or 1%).

Data Staging Systems. For data staging, one architecture decision has to do with the type of data that is being staged and the other has to do with the operations that are offloaded to the surrogate or cloud resource to be performed on the data. As with computation offload, the answer to this question has architecture implications because it requires different components on both sides depending on how data is stored and forwarded.

Concerning *Data Type*, *Field-Collected Data* is sent to an offload target for staging in three of the systems (3/8 or 38%). Staging sensor data addresses storage limitations on mobile devices. In addition, data collected by a surrogate can be shared by other mobile devices connected to the same surrogate or can be fused or pre-processed before sending it to the enterprise.

Application Data is staged in three of the systems (3/8 or 38%). Data that is like to be used by an application on the mobile device is retrieved from a cloud resource and staged on a surrogate. The advantage in this case is lower latency because the data resides in a nearby surrogate and not in a remote cloud.

One system uses the surrogate as an intermediary for *Data Updates* (1/8 or 13%). In Edge Proxy [54] the surrogate informs the mobile device when marked areas of a web page have changed, so that the mobile device is only notified when there are data updates. therefore limiting the amount of direct communication to remote resources.

Finally, one system stages *Data Files* (1/8 or 13%). In Trusted and Unmanaged Data Staging Surrogates [56] a surrogate stages data files that might be needed by the mobile device. The advantage, as in staging application data, is lower latency because the files reside on a nearby surrogate and not in a remote server. Access to the remote server is done by the surrogate and only when the file is not available on the surrogate (similar to a cache miss) or when data on the surrogate has changed and need to be consolidated with the data in the remote server.

Concerning *Data Operations on Surrogate*, two systems perform *Pre-Fetching* operations on the surrogate (2/8 or 25%). The goal is to pre-determine data that is likely to be used by connected mobile devices, retrieve that data from cloud resources, and then store it to reduce the latency of direct cloud access.

Two systems perform *In-Bound Filtering or Pre-Processing* of data that flows from the enterprise (or cloud) to the mobile device (2/8 or 25%). The goal is to pre-process data that is retrieved or pushed from cloud resources so that data is ready to be consumed, or filtered such that the mobile device only receives the data that it needs. The advantage is that the heavy computation and communication to remote servers happens on the surrogates and not on the mobile devices.

Two systems perform *Out-Bound Filtering or Pre-Processing* of data that flows from the mobile device to the enterprise (or cloud) (2/8 or 25%). The goal is for the surrogate to process data that is received from mobile devices such that the data that is sent on to the cloud resource is ready for consumption by the cloud resource (e.g., cleaned, filtered or merged data).

Finally, two systems use the offload target as an extension of the mobile device's storage system for *Data Storage* (2/8 or 25%). All data operations (i.e., CRUD operations) are performed on the surrogate.

4 Observations and Findings from Primary Studies

The primary studies show different and novel computation offload and data staging systems targeted at guaranteeing fidelity of results, and optimizing attributes such as energy consumption, network bandwidth usage, and performance. For computation offload systems, the offload mechanisms range from dynamic approaches in which the computation is provisioned from the mobile device to more static approaches where the computation already exists on the offload target. For data staging systems, the capabilities of the offload target range from an extension of the mobile device's storage to sophisticated algorithms that predict and stage the data that will likely be needed by the mobile device. As far as distribution, the number of computation offloading systems (52) is much larger than the number of data staging systems (8).

A preliminary analysis of the data shows the following gaps and opportunities for architecture strategies for cyber-foraging systems.

- Understanding of quality attributes beyond energy, performance, network usage, and fidelity of results: Many of the cyber-foraging systems, especially those that perform runtime partitioning and offloading decisions, have very complex algorithms for guaranteeing fidelity of results, and optimizing energy consumption, network bandwidth usage and performance. Disconnected operations and fault tolerance are supported by some systems in which the local computation is a fallback mechanism if the remote computation fails. However, there is very little consideration of other quality attributes that are relevant to cyber-foraging systems, such as ease of distribution and installation, resiliency, and security.

– System-level architecture analysis: Related to the previous point, the systems in the studies tend to focus on enabling cyber-foraging between one mobile device and one offload target. However, there is very little discussion of system-level attributes that have to be considered when moving from experimental prototypes to operational systems. For example:
  - How do the systems perform when there are multiple devices trying to offload to the same target?
  - If there are multiple offload targets available, how does the mobile device select the target that best fits its requirements?
  - What happens if the mobile device loses connectivity to the offload target?
  - In those mechanisms that require custom infrastructures or middleware, what are the mechanisms for ensuring currency and compatibility of mobile-side and server-side components if these may not have the same distribution mechanisms?
  - What are the tradeoffs between the quality attributes promoted by the system and other quality attributes such as ease of distribution and installation, resiliency and security?

– Large-scale evaluations: Most of the studies have very limited case studies or evaluations. For example, even though studies talk about mobile cloud computing the experiments are done in controlled environments over WiFi connections, which is not representative of a real mobile cloud environment with disconnections, high latency and multiple heterogeneous users and devices.

– Architectures for data staging systems: The low number of primary studies related to architectures for data staging, combined with an increasing number of data collection devices in the field and the Internet of Things (IoT), show that it is a potential area for developing architecture patterns or tactics that can be leveraged by software architects and developers of these types of systems.

5 Related Work

There are several studies that survey the field of MCC and identify cyber-foraging as a research area and challenge, but are not systematic literature reviews and do not have an architecture focus. Abolfazli et al [62] present a survey of cloud-based mobile augmentation (CMA) approaches, one of which is cyber-foraging. One of the challenges stated by this work is the lack of a reference architecture for CMA. Dinh at al [63] present a survey on MCC. Computation offload is discussed as a technique for extending battery lifetime of mobile devices and listed as one of the challenges for MCC. Fernando et al [64] present a more complete survey on mobile cloud computing. Some of the research that addresses efficient computation offload and distribution to the cloud and how it differs from traditional distributed systems is discussed in this paper. Kumar et al [65] present a survey on computation offloading but focus primarily on the algorithms used to partition and offload programs in order to improve performance or save energy. Finally, Yu et al [66] present a survey on seamless application mobility, which is the continuous or uninterrupted computing experience as a user moves across devices. Code offloading is mentioned as a future direction for seamless application mobility. The work that is most similar to ours is by Flinn et al [67] that presents a discussion of representative cyber-foraging systems and their characteristics. However, it is limited to a small number of systems and does not follow a systematic process. To the best of our knowledge, ours is the first systematic literature review related to architectures for cyber-foraging.

6 Conclusions and Next Steps

We presented preliminary results of an SLR in architectures for cyber-foraging systems and analyzed the primary studies using a categorization of architecture decisions related to what, when and where to offload computation and data from mobile devices. The analysis allowed us to identify gaps and opportunities for research in (1) quality attributes that are relevant to cyber-foraging systems, such as ease of distribution and installation, resiliency, and security, (2) system-level architecture analysis, (3) large-scale evaluations, and (4) architectures for data staging systems. Our next steps are to further refine the analysis and cluster the results to identify architectural tactics that can be employed by system architects to build systems that use cyber foraging, with an analysis of the quality attributes and tradeoffs related to each tactic.

Acknowledgments

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. This material has been approved for public release and unlimited distribution (DM-0001173).

References

1. T. Dyba, T. Dingsoyr, and G. Hanssen, "Applying systematic reviews to diverse study types: An experience report," in *Empirical Software Engineering and Measurement, 2007. ESEM 2007. First International Symposium on*, September 2007, pp. 225–234.
2. B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," Keele University and Durham University Joint Report, Tech. Rep. EBSE 2007-001, 2007.

3. N. Balasubramanian, A. Balasubramanian, and A. Venkataramani, “Energy consumption in mobile phones: A measurement study and implications for network applications,” in *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement Conference*, ser. IMC ’09. New York, NY, USA: ACM, 2009, pp. 280–293.
4. J. Ahnn and M. Potkonjak, “mhealthmon: Toward energy-efficient and distributed mobile health monitoring using parallel offloading,” *Journal of Medical Systems*, vol. 37, no. 5, pp. 1–11, 2013.
5. P. Angin and B. Bhargava, “An agent-based optimization framework for mobile-cloud computing,” *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)*, vol. 4, pp. 1–17, 2013.
6. A. Aucinas, J. Crowcroft, and P. Hui, “Energy efficient mobile m2m communications,” in *Proceedings of ExtremeCom ’12*, 2012.
7. R. K. Balan, D. Gergle, M. Satyanarayanan, and J. Herbsleb, “Simplifying cyber foraging for mobile devices,” in *Proceedings of the 5th international conference on Mobile systems, applications and services*, ser. MobiSys ’07. New York, NY, USA: ACM, 2007, pp. 272–285.
8. Y.-S. Chang and S.-H. Hung, “Developing collaborative applications with mobile cloud-a case study of speech recognition,” *Journal of Internet Services and Information Security (JISIS)*, vol. 1, no. 1, pp. 18–36, 2011.
9. G. Chen, B.-T. Kang, M. Kandemir, N. Vijaykrishnan, M. J. Irwin, and R. Chandramouli, “Studying energy trade offs in offloading computation/compilation in java-enabled mobile devices,” *Parallel and Distributed Systems, IEEE Transactions on*, vol. 15, no. 9, pp. 795–809, 2004.
10. B. Cheng and M. Probst, “Hbb-next i d4.4.1: Intermediate middleware software components for cloud service offloading,” HBB-NEXT Consortium 2013, Tech. Rep., 2013.
11. H.-h. Chu, H. Song, C. Wong, S. Kurakake, and M. Katagiri, “Roam, a seamless application framework,” *Journal of Systems and Software*, vol. 69, no. 3, pp. 209–226, 2004.
12. B.-G. Chun and P. Maniatis, “Augmented smartphone applications through clone cloud execution,” in *Proceedings of the 12th conference on Hot topics in operating systems*. USENIX Association, 2009, pp. 8–8.
13. E. Cuervo, “Enhancing mobile devices through code offload,” Ph.D. dissertation, Duke University, 2012.
14. N. Duga, “Optimality analysis and middleware design for heterogeneous cloud hpc in mobile devices,” Master’s thesis, Addis Ababa University, 2011.
15. H. Endt and K. Weckemann, “Remote utilization of opencl for flexible computation offloading using embedded ecus, ce devices and cloud servers,” in *Volume 22: Applications, Tools and Techniques on the Road to Exascale Computing*, ser. Advances in Parallel Computing. IOS Press EBooks, 2011, vol. 22, pp. 133–140.
16. R. G. Esteves, M. D. McCool, and C. Lemieux, “Real options for mobile communication management,” in *GLOBECOM Workshops (GC Wkshps), 2011 IEEE*. IEEE, 2011, pp. 1241–1246.
17. T. Fjellheim, S. Milliner, and M. Dumas, “Middleware support for mobile applications,” *International Journal of Pervasive Computing and Communications*, vol. 1, no. 2, pp. 75–88, 2005.
18. J. Flinn, S. Park, and M. Satyanarayanan, “Balancing performance, energy, and quality in pervasive computing,” in *In Proceedings of the 22nd International Conference on. Distributed Computing Systems*, 2002, pp. 217–226.

19. 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*, ser. Lecture Notes in Computer Science, J. Bacon and B. Cooper, Eds., vol. 5896. Springer Berlin Heidelberg, 2009, pp. 83–102.
20. S. Goyal, “A collective approach to harness idle resources of end nodes,” Ph.D. dissertation, School of Computing, University of Utah, 2011.
21. T. Guan, “A system architecture to provide enhanced grid access for mobile devices,” Ph.D. dissertation, University of Southampton, 2008.
22. K. Ha, G. Lewis, S. Simanta, and M. Satyanarayanan, “Cloud offload in hostile environments,” Carnegie Mellon University, Tech. Rep., 2011.
23. S.-H. Hung, J.-P. Shieh, and C.-P. Lee, “Migrating android applications to the cloud,” *International Journal of Grid and High Performance Computing (IJGHPC)*, vol. 3, no. 2, pp. 14–28, 2011.
24. S. Imai, “Task offloading between smartphones and distributed computational resources,” Master’s thesis, Rensselaer Polytechnic Institute, 2012.
25. A. N. Iyer *et al.*, “Extending android application programming framework for seamless cloud integration,” in *Mobile Services (MS), 2012 IEEE First International Conference on*. IEEE, 2012, pp. 96–104.
26. C. Jarabek, D. Barrera, and J. Aycock, “Thinav: truly lightweight mobile cloud-based anti-malware,” in *Proceedings of the 28th Annual Computer Security Applications Conference*. ACM, 2012, pp. 209–218.
27. R. Kemp, N. Palmer, T. Kielmann, and H. Bal, “Cuckoo: a computation offloading framework for smartphones,” in *Mobile Computing, Applications, and Services*. Springer, 2012, pp. 59–79.
28. S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, “Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading,” in *INFOCOM, 2012 Proceedings IEEE*. IEEE, 2012, pp. 945–953.
29. D. Kovachev and R. Klamma, “Framework for computation offloading in mobile cloud computing,” *International Journal of Interactive Multimedia and Artificial Intelligence*, vol. 1, no. 7, pp. 6–15, 2012.
30. M. D. Kristensen, “Empowering mobile devices through cyber foraging,” Ph.D. dissertation, Aarhus University, 2010.
31. Y.-W. Kwon and E. Tilevich, “Reducing the energy consumption of mobile applications behind the scenes,” in *Proceedings of the 29th IEEE International Conference on Software Maintenance (ICSM 2013)*, 2013.
32. B.-D. Lee, “A framework for seamless execution of mobile applications in the cloud,” in *Recent Advances in Computer Science and Information Engineering*. Springer, 2012, pp. 145–153.
33. J. Matthews, M. Chang, Z. Feng, R. Srinivas, and M. Gerla, “Powersense: power aware dengue diagnosis on mobile phones,” in *Proceedings of the First ACM Workshop on Mobile Systems, Applications, and Services for Healthcare*. ACM, 2011, p. 6.
34. A. Messer, I. Greenberg, P. Bernadat, D. Milojicic, D. Chen, T. Giuli, and X. Gu, “Towards a distributed platform for resource-constrained devices,” in *Distributed Computing Systems, 2002. Proceedings. 22nd International Conference on*. IEEE, 2002, pp. 43–51.
35. D. Messinger and G. A. Lewis, “Application virtualizaton as a strategy for cyber foraging in resource-constrained environments,” Carnegie Mellon Software Engineering Institute, Tech. Rep., 2013.
36. S. Mohapatra and N. Venkatasubramanian, “Optimizing power using a reconfigurable middleware,” UC Irvine, Tech. Rep., 2003.

37. M. Ok, J.-W. Seo, and M.-s. Park, “A distributed resource furnishing to offload resource-constrained devices in cyber foraging toward pervasive computing,” in *Network-Based Information Systems*. Springer, 2007, pp. 416–425.
38. M. J. O’Sullivan and D. Grigoras, “The cloud personal assistant for providing services to mobile clients,” in *Service Oriented System Engineering (SOSE), 2013 IEEE 7th International Symposium on*, 2013, pp. 478–485.
39. S. Park, Y. Choi, Q. Chen, and H. Yeom, “Some: Selective offloading for a mobile computing environment,” in *Cluster Computing (CLUSTER), 2012 IEEE International Conference on*, 2012, pp. 588–591.
40. L. Pu, J. Xu, X. Jin, and J. Zhang, “Smartvirtcloud: virtual cloud assisted application offloading execution at mobile devices’ discretion,” in *2013 IEEE Wireless Communications and Networking Conference (WCNC): SERVICES and APPLICATIONS*, 2013.
41. M.-R. Ra, A. Sheth, L. Mummert, P. Pillai, D. Wetherall, and R. Govindan, “Odessa: enabling interactive perception applications on mobile devices,” in *Proceedings of the 9th international conference on Mobile systems, applications, and services*, ser. MobiSys ’11. New York, NY, USA: ACM, 2011, pp. 43–56.
42. K. K. Rachuri, “Smartphones based social sensing: Adaptive sampling, sensing and computation offloading,” Ph.D. dissertation, University of Cambridge, 2012.
43. M. R. Rahimi, N. Venkatasubramanian, S. Mehrotra, and A. V. Vasilakos, “Mapcloud: mobile applications on an elastic and scalable 2-tier cloud architecture,” in *Proceedings of the 2012 IEEE/ACM Fifth International Conference on Utility and Cloud Computing*. IEEE Computer Society, 2012, pp. 83–90.
44. 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.
45. C. Shi, P. Pandurangan, K. Ni, J. Yang, M. Ammar, M. Naik, and E. Zegura, “Ic-cloud: Computation offloading to an intermittently-connected cloud,” Georgia Institute of Technology, Tech. Rep., 2013.
46. J. N. Silva, L. Veiga, and P. Ferreira, “Spade: scheduler for parallel and distributed execution from mobile devices,” in *Proceedings of the 6th international workshop on Middleware for pervasive and ad-hoc computing*. ACM, 2008, pp. 25–30.
47. Y.-Y. Su and J. Flinn, “Slingshot: deploying stateful services in wireless hotspots,” in *Proceedings of the 3rd international conference on Mobile systems, applications, and services*, ser. MobiSys ’05. New York, NY, USA: ACM, 2005, pp. 79–92.
48. K. Yang, S. Ou, and H.-H. Chen, “On effective offloading services for resource-constrained mobile devices running heavier mobile internet applications,” *Communications Magazine, IEEE*, vol. 46, no. 1, pp. 56–63, 2008.
49. L. Yang, J. Cao, Y. Yuan, T. Li, A. Han, and A. Chan, “A framework for partitioning and execution of data stream applications in mobile cloud computing,” *ACM SIGMETRICS Performance Evaluation Review*, vol. 40, no. 4, pp. 23–32, 2013.
50. Y. Zhang, X.-t. Guan, T. Huang, and X. Cheng, “A heterogeneous auto-offloading framework based on web browser for resource-constrained devices,” in *Internet and Web Applications and Services, 2009. ICIW’09. Fourth International Conference on*. IEEE, 2009, pp. 193–199.
51. X. Zhang, A. Kunjithapatham, S. Jeong, and S. Gibbs, “Towards an elastic application model for augmenting the computing capabilities of mobile devices with cloud computing,” *Mobile Networks and Applications*, vol. 16, no. 3, pp. 270–284, 2011.

52. Y. Zhang, G. Huang, W. Zhang, X. Liu, and H. Mei, "Towards module-based automatic partitioning of java applications," *Frontiers of Computer Science*, vol. 6, no. 6, pp. 725–740, 2012.
53. X. Zhang, W. Jeon, S. Gibbs, and A. Kunjithapatham, "Elastic html5: Workload offloading using cloud-based web workers and storages for mobile devices," in *Mobile Computing, Applications, and Services.* Springer, 2012, pp. 373–381.
54. T. Armstrong, O. Trescases, C. Amza, and E. de Lara, "Efficient and transparent dynamic content updates for mobile clients," in *Proceedings of the 4th international conference on Mobile systems, applications and services.* ACM, 2006, pp. 56–68.
55. A. Bahrami, C. Wang, J. Yuan, and A. Hunt, "The workflow based architecture for mobile information access in occasionally connected computing," in *Services Computing, 2006. SCC'06. IEEE International Conference on.* IEEE, 2006, pp. 406–413.
56. J. Flinn, S. Sinnamohideen, N. Tolia, and M. Satyanarayanan, "Data staging on untrusted surrogates," in *Proceedings 2nd USENIX Conference on File and Storage Technologies (FAST03), Mar 31-Apr 2, 2003, San Francisco, CA.*, 2003.
57. S. Kundu, J. Mukherjee, A. K. Majumdar, B. Majumdar, and S. Sekhar Ray, "Algorithms and heuristics for efficient medical information display in pda," *Computers in Biology and Medicine*, vol. 37, no. 9, pp. 1272–1282, 2007.
58. T. Phokas, H. Efstathiades, G. Pallis, and M. Dikaiakos, "Feel the world: A mobile framework for participatory sensing," in *Mobile Web Information Systems*, ser. Lecture Notes in Computer Science, F. Daniel, G. Papadopoulos, and P. Thiran, Eds., vol. 8093. Springer Berlin Heidelberg, 2013, pp. 143–156.
59. Y. Xiao, P. Simoens, P. Pillai, K. Ha, and M. Satyanarayanan, "Lowering the barriers to large-scale mobile crowdsensing," in *Mobile Computing Systems and Applications*, 2013.
60. F. Yang, Z. Qian, X. Chen, I. Beschastnikh, L. Zhuang, L. Zhou, and J. Shen, "Sonora: A platform for continuous mobile-cloud computing," Technical Report. Microsoft Research Asia, Tech. Rep., 2012.
61. K. Kumar and Y.-H. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" *Computer*, vol. 43, no. 4, pp. 51–56, Apr. 2010.
62. S. Abolfazli, Z. Sanaei, E. Ahmed, A. Gani, and R. Buyya, "Cloud-based augmentation for mobile devices: Motivation, taxonomies, and open challenges," *IEEE Communications Surveys Tutorials*, vol. 16, no. 1, pp. 337–368, 2014.
63. H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: architecture, applications, and approaches," *Wireless Communications and Mobile Computing*, vol. 13, pp. 1587–1611, 2011.
64. N. Fernando, S. W. Loke, and W. Rahayu, "Mobile cloud computing: A survey," *Future Generation Computer Systems*, vol. 29, p. 84106, 2012.
65. K. Kumar, J. Liu, Y.-H. Lu, and B. Bhargava, "A survey of computation offloading for mobile systems," *Mobile Networks and Applications*, vol. 18, no. 1, pp. 129–140, Feb. 2013.
66. P. Yu, X. Ma, J. Cao, and J. Lu, "Application mobility in pervasive computing: A survey," *Pervasive and Mobile Computing*, vol. 9, pp. 2–17, 2012.
67. J. Flinn, "Cyber foraging: Bridging mobile and cloud computing," in *Synthesis Lectures on Mobile and Pervasive Computing*, M. Satyanarayanan, Ed. Morgan & Claypool Publishers, 2012.