A Survey of Adaptation Techniques in Computation Offloading

Arani Bhattacharya[a,b,*], Pradipta De[c]

[a]Department of Computer Science, Stony Brook University, New York, 11794, USA
[b]Department of Computer Science, SUNY Korea, 119, Songdo Moonwha-Ro, Yeonsu-Gu, Incheon, Republic of Korea, 21985
[c]Department of Computer Sciences, Georgia Southern University, Statesboro, Georgia, 30460, USA

Abstract

Computation offloading is a method of saving energy and time on resource-constrained mobile devices by executing some tasks on the cloud. A computation offloading mechanism determines portions of the application that can be offloaded for remote execution. The offloading decision problem depends on various parameters, like application characteristics, network conditions, hardware features, that influence the operating environment of an offloading system. Variations in these parameter values make it challenging to design effective offloading solutions that can adapt gracefully to all the changes. Hence a number of work has focused on offloading solutions that can adapt to changes in the parameter values. In this paper, we survey adaptation techniques used by offloading systems. We provide a panoptic view of the task offloading problem by identifying the variable parameters in the offloading ecosystem, present offloading solutions that adapt to these parameters, and point out the associated improvements in Quality of Experience of users.

Keywords: Mobile Cloud Computing, Application Offloading, Distributed Systems, Smartphones, Adaptive offloading

1. Introduction

Computation offloading involves partitioning power-hungry mobile applications to utilize remote cloud resources. An offloading framework identifies portions of code that are profiled to be computation and energy intensive to execute on cloud servers. The benefits of offloading to save energy on mobile devices has been demonstrated in several prototype systems [31, 29, 60]. However, computation offloading is yet to be in mainstream use on mobile devices [14] [44].

One of the challenges towards practical use of computation offloading is the unpredictable operating environment. There are several sources of variation due to application characteristics, network conditions and platform differences. The unstable bandwidth of wireless networks can hurt the gains derived from using offloading. Similarly, diversity in application workload on the device, or on the cloud servers can diminish the potential gains from offloading.

At the core of it, the problem lies in how the task offloading mechanism selects the tasks for remote execution. If the offloading decision cannot factor in variations in the operating environment, it can lead to poor performance. Several recent works have explored offloading techniques that can adapt at run-time to changes in the operating parameters. Adaptive offloading solutions are complex given the presence of several parameters and their unpredictable variations. Hence the solutions have explored techniques that range from adapting to single parameter to multiple parameters.

In this paper, we survey the state-of-the-art adaptive algorithms used for computation offloading. In order to define the sources of variation in the operating environment, we describe the entities in a practical mobile-cloud ecosystem. We use this ecosystem to define the parameters that can vary at runtime, and its effect on user experience. We classify the solutions for adaptive offloading based on the parameters that the solution can adapt to. We also present the effect of the parameters on the Quality of Experience metrics suitable for offloading environment, like energy saved, application completion time, monetary cost, and security features.

A number of surveys have studied offloading frameworks from various viewpoints (Table 1). For example, Dinh et al. [34] summarize the overall idea of computation offloading, and discuss its different applications. Shiraz et al. [100] and Sharifi et al. [95] discuss different methods of implementing offloading from smartphones. Abolfazli et al. [2] study the impact of types of cloud resources used on computation offloading. A more recent survey, Sanaei et al. [88], provides a taxonomy of the environment in which offloading is used. Liu et al. [75] focus on the offloading algorithm used by different offloading frameworks. Finally, Shaukat et al. [96] provide a taxonomy of solutions that use cloudlets, which are defined as resource-rich computers in proximity to mobile devices. However, these surveys do not study the adaptation of offloading algorithms to different environmental conditions. Since offloading is known to improve performance of mobile applications, an important factor behind its adoption lies in the way it works in the real environment. Thus, we focus on adaptive offloading techniques aimed at handling changes in the operating environment.

The contributions of this paper are summarized as follows:

*Corresponding author
Email addresses: arbhattachar@cs.stonybrook.edu (Arani Bhattacharya), pde@georgiasouthern.edu (Pradipta De)

Table 1: A summary of related surveys, and their relation to our work.

Work	Topic
Sharifi et al. [95]	Working of different computation offloading frameworks
Shiraz et al. [100]	Implementation techniques of different computation offloading frameworks
Dinh et al. [34]	Overall survey of mobile cloud computing, including offloading to cloud servers
Fernando et al. [42]	Survey of different design choices faced by design of computation offloading framework
Kumar et al. [61]	Classification of offloading algorithms for computation offloading
Khan et al. [58]	Survey of different ways of modeling a mobile application for computation offloading
Abolfazli et al. [2]	Survey of cloud resources used to augment mobile applications and its effect on their performance
Khan [59]	Survey of strategies to speed up application execution using computation offloading
Sanaei et al. [88]	Taxonomy of environmental variation in mobile cloud computing
Liu et al. [75]	Working of different computation offloading algorithms
Ahmed et al. [7]	Survey of different objectives and constraints used in offloading
Alizadeh et al. [8]	Authentication techniques in mobile cloud computing
Shiraz et al. [101]	Analysis of time, energy and cost overhead of deploying offloading framework on smartphones
Ahmed et al. [6]	Seamless execution of mobile cloud applications
Shaukat et al. [96]	Taxonomy of cloudlet-based solutions (computing resources close to mobile devices)
Our Work	**Adaptation of offloading frameworks to environmental variation**

Table 2: A list of acronyms used in this survey.

DAG	Directed Acyclic Graph
DVFS	Dynamic Voltage and Frequency Scaling
GPU	Graphical Processing Unit
MEC	Mobile Edge Computing
SDN	Software-defined Networking
QoE	Quality of Experience
VM	Virtual Machine

- We identify parameters that are prone to variation in a practical mobile cloud ecosystem.
- We discuss adaptation techniques used by offloading frameworks to handle varying parameters.
- We explain the effect of different parameters of the mobile-cloud ecosystem on user experience.

The rest of this paper is organized as follows. Section 2 presents the working of an offloading framework and explains the necehassity of making it adaptive to the operating environment. In Section 3, we introduce our classification of the environment, and the Quality of Experience (QoE) metrics. Section 4 shows the decision problem that an offloading framework solves to execute an application. Sections 5, 6, 7 and 8 discuss the way offloading frameworks adapt to diverse mobile applications, network conditions, execution platforms and cloud management techniques respectively. In Section 9, we discuss different parameters that affect the user experience. In Section 10 we highlight some existing challenges hindering adoption of offloading frameworks. Section 11 concludes this survey.

2. Background

In this section, we first explain the working of an offloading framework. Then we explain the concept of adaptation in the context of offloading.

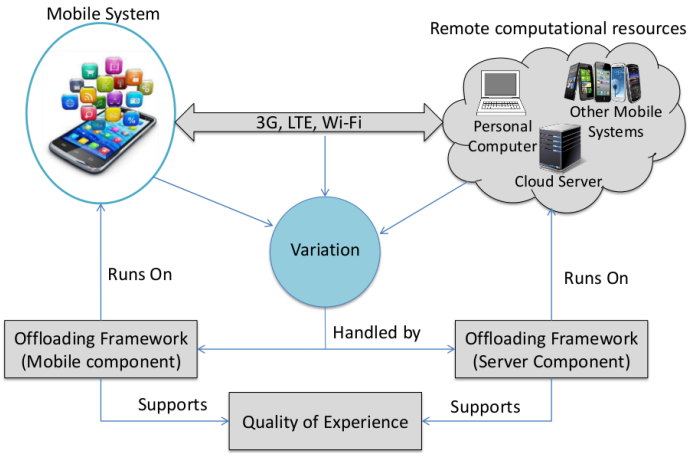

Figure 1: Typical architecture of a computation offloading system. The mobile device uses the wireless network to offload computation to the remote servers. The offloading system supports userSh expectations of Quality of Experience (QoE) while handling variations in different components.

2.1. Computation Offloading

A user expects the mobile system to run a variety of applications. However, a mobile system is constrained by the residual battery capacity at any point in time, and the limited computation power of existing mobile processors [90]. Computation offloading or cyber-foraging aims to enable energy or computation intensive applications on mobile systems by distributed execution of mobile applications. This is done by migrating a portion of the application state from the mobile device to remote computation resources.

Figure1 shows the components of an offloading system. The user executes a variety of mobile applications on the smartphone. The smartphone can access the Internet through one or more wireless network channels. It migrates some portions of the running applications to other computation resources available using the network. Such computation resources may be the user's own personal computer [84], a processor attached to

an wireless access point [92], other mobile devices [36], a cloud server [29] or even routers and switches in the network [50]. On completion of the task on the remote server, the new program state is transferred back to the smartphone.

In order to enable distributed execution of a mobile application, the offloading framework must determine how to partition an application for scheduling on mobile device and cloud servers. This is decided by the offloading decision engine, which may be present either on the smartphone or on a pre-defined server. The offloading decision engine needs to identify the most energy or computation intensive tasks of the given application. Using this information and the condition of the environment, it selects the part of the program to be offloaded for remote execution. The decision taken by it determines the amount of resources saved on the mobile device and the quality of experience (QoE) of the user.

2.2. Adaptation in Mobile Cloud Computing Systems

A mobile system is used in varying conditions. For example, users can move while talking to someone on the phone, or view videos while sitting in a car. This affects the network performance. Similarly, users may switch to another application, and thus change the workload on the offloading system. The computation power of remote resources and the mobile device also vary. A mobile system must be *adaptive* to these changing conditions [89]. *We define a system as adaptive if it is designed to continuously monitor its environment and then modify its behavior in response to changing environmental conditions* [25].

Offloading systems also need to be adaptive to changes in operating environment [44]. They are affected by wireless network characteristics and the capabilities of available computation resources. The workload also varies depending on the applications that are running on the smartphone. These variations observed in the environment are shown in Figure 1. An offloading system has to adapt to these changes while maintaining a good QoE for the user.

In order to build an adaptive offloading system, a proper understanding of the range of environmental variables and user expectations is essential. The architect of an offloading framework can then incorporate the user expectations while implementing it. To meet the user expectations, the use case scenarios can help in determining the importance of different parameters for adaptation.

3. Mobile Cloud Ecosystem: Parameters and QoE Metrics

In this section, we define the mobile cloud ecosystem and its various components. To study the impact on the user, we define ways of measuring Quality of Experience (QoE).

Proper functioning of an offloading system depends on a lot of operating parameters. The values of different parameters can be transient and are hard to predict. Moreover, many of them keep on changing during execution of mobile applications. Thus offloading systems need to ensure that they adapt to these changes.

Figure 2 illustrates the effect of the mobile cloud ecosystem on the user's QoE. The ecosystem consists of three distinct components – mobile, network and cloud system. These three components of the mobile cloud ecosystem together affect the QoE of the user. The QoE in turn, has four components – completion time, energy consumption, monetary cost and security.

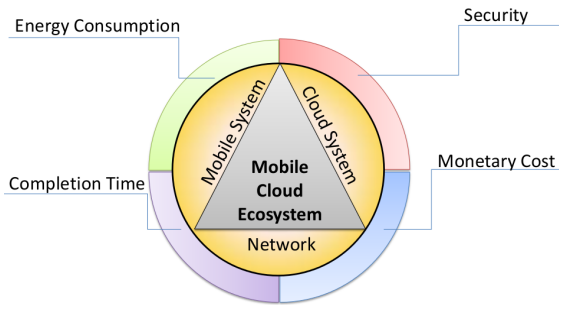

Figure 2: The figure depicts the source of different parameters in the mobile cloud ecosystem, and the impact of the parameters on the QoE of the user.

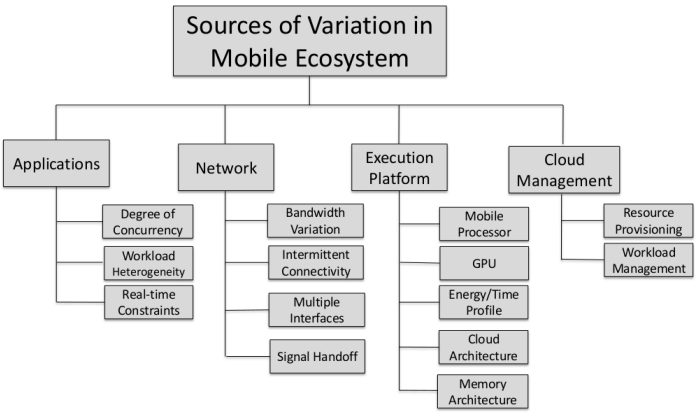

Figure 3: A taxonomy of sources of variation in the entire mobile cloud ecosystem. Each source of variation has various parameters associated with it, each of which vary in different ways.

3.1. Mobile Cloud Ecosystem

Based on different parameters of an offloading system, we define an ecosystem. Each source of variation in the ecosystem forms one component. The parameters used to measure such variation are then listed for each component.

Figure 3 provides a taxonomy of the overall variations seen in the environment. The environmental parameters are categorized into four components based on the source of variation – application, network, execution platform and cloud management. Each of these components have a number of environmental parameters in which variation is observed. The application component has three parameters – degree of concurrency, workload heterogeneity and real-time constraints. The network component has four parameters – bandwidth variation, intermittent connectivity, multiple interfaces and signal handoff. The execution platform has five parameters – mobile processor, presence of GPU, energy or time profile, memory architecture and cloud architecture. Finally, cloud management consists of resource allocation and workload management.

We now define the different sources of variation in a mobile-cloud ecosystem.

Table 3: A list of applications that have been optimized using offloading.

Application	Offloading Solution
Face recognition	MAUI [31], [67]
Image search	CloneCoud [29]
Virus-scanning	CloneCloud [29], [119]
Speech recognition	COSMOS [97]
Text Editor	Namboodiri & Ghose [79]
Chess game	CloneCloud [29]
Gesture recognition	Odessa [84]
Object recognition	Glimpse [23]
Optical Character Reader	Lin & Kung [71]
Indoor map reconstruction	Chen et al. [20]
Video transcoding	Zhang et al. [117]
Video encoding	Native Offloader [64]
Web browser	Native Offloader [64]
Social networking (Facebook, Twitter)	CDroid [14]
Data compression	Native Offloader [64]

3.1.1. Application

Smartphones allow users to install applications developed by third-party developers. Leading mobile software repositories such as Google Play[1] or Apple App Store[2] have over a million different applications. Computation offloading can improve the performance of a variety of applications, including text editors [79], virus scanners [119], games [31], object and gesture recognition [23, 84], face detection [67], speech recognition [19] and indoor map reconstruction [20]. A list of applications that have been optimized by different frameworks is given in Table 3. These applications impose varying requirements on the mobile system. Handling these variations for an offloading system is a major challenge.

Applications have different levels of concurrency. They may be interactive or non-interactive. Moreover, the number of applications executing simultaneously on the mobile device varies. An offloading system must adapt depending on the changes in all these parameters. The application level variations are reflected in the following parameters.

- **Degree of concurrency**: Most smartphone applications are concurrent. There are two types of concurrency – task-level and data-level. Applications that execute independent tasks have task-level concurrency [60]. Programs with stream data, such as multimedia applications, have data-level concurrency [84].
- **Workload Heterogeneity**: Smartphones usually execute multiple applications with different user expectations. For example, a user might use a social networking app while listening to music. Each of these different applications may impose varying workload on the mobile system. A video streaming application imposes high computation and network costs. On the other hand, a social networking app imposes less computation cost, but utilizes the network after a fixed duration.
- **Real-time Constraints**: Mobile applications may or may not have real-time constraints. There are two types of real-time constraints. Soft constraints, usually found in streaming applications, require limited response time for only a proportion of the given tasks. On the other hand, hard constraints require guaranteed response time for each task. They are found in gaming applications.

3.1.2. Network

Offloading systems use the wireless network to communicate with the cloud system. The condition of wireless networks is typically unstable, leading to frequent changes in the network condition. Mobile systems need to adapt to these changes. The variation in network conditions are typically measured using the following parameters:

- **Bandwidth Variation**: The bandwidth of mobile networks varies depending on the user's position and network congestion. The variation in bandwidth is higher in wireless networks as compared to wired networks. This is due to multiple factors – user mobility, interference from other users and distance of the user from the mobile device to the cloud [3]. Bandwidth and latency of wireless channels are usually difficult to predict. This makes adapting to them challenging. For applications with deadline constraints, variation in bandwidth or latency is a major problem [79].
- **Intermittent Connectivity**: A mobile user may lose connectivity to the network. This can happen due to mobility of the user, or resource constraints of the server. A mobile system must tolerate loss of connectivity, and continue to function seamlessly [6].
- **Multiple Interfaces**: Mobile systems have multiple network interfaces, like 3G, LTE or Wi-Fi. Each of these interfaces have different characteristics related to their coverage area, latency and bandwidth, and energy consumption. An adaptive offloading system has to determine what to offload based on the network interface that is used to connect to the Internet.
- **Signal Handoff**: Users of mobile devices may move from one place to another while using them. This forces the network interface to smoothly switch to a different access point or hub while maintaining connectivity. This process is called signal handoff. Handling signal handoff while maintaining stable connectivity is challenging [68]. This makes it essential to incorporate adaptation techniques to handle signal handoff for offloading systems.

3.1.3. Execution Platform

The execution platform refers to both the mobile device, and the remote cloud servers. The organization and hardware of mobile devices as well as cloud systems have a lot of variation. A mobile system must be adaptive to these changes, since it is not feasible to develop a separate system for each individual configuration. Variation in execution platform can be analyzed using the following parameters.

[1] http://play.google.com
[2] http://store.apple.com

- **Mobile Processor**: The design of processors differs across different mobile systems. For example, modern smartphones may have up to 8 processors [82]. Moreover, some processors even on the same device may have different frequencies [27]. Apart from these variations, processors can run at lower than maximum frequency to save energy using Dynamic Voltage and Frequency Scaling (DVFS). An adaptive mobile system that aims to improve performance must schedule tasks according to the mobile processor types.

- **GPU**: Most modern mobile systems have a Graphical Processing Unit (GPU) to perform special tasks like graphics rendering. Usually, the instructions for GPUs are more complex than ordinary processors and thus, consume more energy. An adaptive system has to decide how to use the GPU based on the battery condition [74].

- **Energy and Time Profile**: The mobile device comprises of many hardware components, like the processor system, memory, network interfaces, storage, GPS and camera. Each of these components have their own energy characteristics. There is also significant variation across phones. Thus, adaptive offloading systems need to automatically determine the amount of energy consumed, and develop the energy profile for different applications.

  Apart from energy characteristics, each component of a mobile device also has its own time characteristic. For example, execution time increases with a reduction in processor frequency. Thus, variation in processor frequency, both within and across mobile devices, affects the time profile [11].

- **Memory Architecture**: The mobile and cloud system may have different memory layout and endianness. Memory layout refers to the address size of a single memory location. For example, most server systems use 64-bit virtual address, whereas mobile devices use 32-bit virtual addresses. Endinanness refers to the order in which bytes of a single word are stored in memory. Most cloud data centers use little-endian storage layout, whereas mobile devices may use either big-endian or little-endian. An offloading system needs to handle these variations to ensure its correct working [64].

- **Cloud Architecture**: We define cloud architecture as the type and network layout of remote computing resources available to the smartphone user. A variety of computing resources are nowadays available, such as user desktops, other mobile devices in the vicinity, or cloud data centers. Depending on the environment, the offloading framework may use any of these computing resources to improve smartphone application performance. These resources have different processing power, and communication latencies with the smartphone. The type of computing resources used, therefore, affects the execution time of an application.

  Computing resources are organized in a hierarchical system in the network, where the processors closer to the smartphones have less powerful processors. For example, data center servers have high computation power, but have high network latency. In contrast, a user desktop processor has low computation power, but has lower latency since it is closer to the smartphone network. Some offloading systems use cloud servers [31], while others use processors attached to wireless access points [91]. Yet another type of systems use mobile devices in the vicinity [77]. A combination of these systems can also be used. For optimal usage, an offloading system must adapt to resources that are available, and decide which utilizing which remote computing resources will provide good application performance.

3.1.4. Cloud Management

There are multiple ways of managing a cloud server. The performance of cloud server affects the speed and cost of computation. Variations in cloud management can be broadly measured using the following parameters:

- **Resource Provisioning**: Cloud resources include processors, energy and software resources such as Operating Systems and APIs. The user also usually has the option to choose the number of processors within a server and specify the speed of a particular processor. An offloading framework needs to recognize and adapt to the type of resources available in the cloud system [103].

- **Workload Management**: The workload on cloud servers varies depending on the number of users they support. The response times of the server varies depending on the amount of workload on a single processor. Cloud server systems use different load balancing techniques to handle workload variation. Thus, offloading frameworks have to adapt to the varying workload of the cloud server [99].

3.2. Quality of Experience (QoE) Metrics

The objective of offloading systems is to deliver high QoE. To attain this objective, the architect of an offloading system must have a clear idea of the way QoE is measured. For mobile device users, the QoE consists of four parameters – energy consumption, completion time, monetary cost and security.

3.2.1. Energy Consumption

The energy consumption is the total energy spent on the mobile device to execute an application. This comprises of the energy cost of computation, the energy spent in transmitting data, and the energy spent by other mobile components like memory, storage and sensors. However, the total energy spent is higher than the sum of energy spent by individual components [22]. Thus, realistic techniques of measuring energy consumption are essential.

We consider only the energy consumption of the mobile device. This is because the objective is to prolong the battery life of the mobile device. Thus, although the energy spent in the

cloud has become increasingly important, it does not affect the life of the battery. We therefore exclude it from this study.

3.2.2. Completion time

The completion time is the total time taken by the system to complete the execution of the entire program for processor-intensive applications. For applications with hard real-time constraints, it is the amount of time taken to execute a particular task. Such applications require a *deterministic guarantee* of completing a task within a fixed duration. Applications with soft-real time constraints specify the amount of time along with the percentage of tasks that must be completed. Soft-real time constraints require *statistical guarantees* of performance [18].

3.2.3. Monetary Cost

The monetary cost consists of two distinct components:

- **Network Usage:** Network usage refers to the cost of communication between the mobile device and the cloud system. This may be charged by the mobile or the cloud service provider or both. It is equal to the sum of inbound and outbound traffic from the mobile to the cloud.
- **Server Usage:** The cost of renting the server constitutes the second component of monetary cost. This usually depends on the number of virtual servers, and the type of cloud organization that are used by the offloading system.

3.2.4. Security

In the context of offloading systems, security refers to maintaining confidentiality of private data and integrity of computed data. Offloading transfers data to other computation resources. This data might contain information private to the user. Moreover, the mobile device has to trust the computation resource that the software component is executed correctly [109].

4. Task Offloading Problem

The offloading decision problem is solved by the decision engine, which is part of an offloading system. The solution of this problem gives the points of execution of each task of an application. Thus, it plays a critical role in the amount of computation resources saved and the quality of experience (QoE) of the user.

In this section, we describe a formulation of the offloading decision problem. We first discuss the ways of modeling the system, and then explain the formulation based on this model. Finally, we describe how this formulation can be used to develop an adaptive solution, and the challenges of increasing adaptiveness.

4.1. System Model

An offloading system consists of three main components – the mobile application, the network and the execution platform. The decision problem formulation needs to have a mathematical model of each of these components.

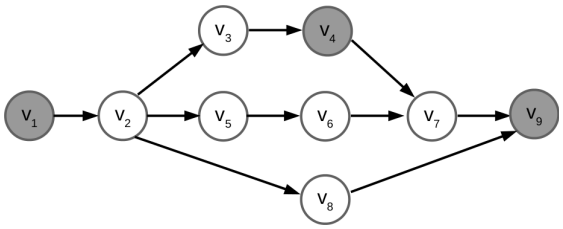

Figure 4: Directed Acyclic Graph (DAG) representation of a mobile application where the vertices represent tasks and the edges represent dependencies among tasks. The shaded vertices represent tasks that must be executed locally.

A mobile application is represented as a Directed Acyclic Graph (DAG). The vertices of the DAG represent different *tasks* or components of the software. The edges represent the data dependencies among the tasks. For example, an edge from the task v_i to v_j (usually denoted as (v_i, v_j) or (i, j)) denotes that the output of task v_i is needed to execute task v_j. Moreover, some of the tasks might depend on some sensors available only on the mobile device, such as camera or GPS. These tasks may only be executed on the mobile device.

Each task and dependency of an application has one or more costs associated with it. This cost can be either energy, time, bandwidth or any other limited resource. The cost associated with a task is incurred when the task is executed locally. Execution on a remote computation resource incurs a fraction of the cost, depending on the type of cost and computation resource available. For example, the energy cost incurred on executing a task remotely is zero, whereas the time cost depends on the speed of the remote computation resource compared to the mobile device. The cost associated with a dependency is incurred when the two tasks are executed on different points of execution. This is known as migration cost, since it is incurred in order to migrate the output from one execution device to another. Moreover, this migration cost decreases with an increase in the quality of network performance.

The execution platform consists of one mobile device (smartphone) and some other remote computation resources, like cloud server or wireless access points. All these computation resources contain one or more processors, each having its own time and energy characteristics. We also assume that execution of an application must begin and end at the mobile device.

An example of a DAG representation is shown in Figure 4. In this figure, the tasks v_1, v_4 and v_9 must be executed on the mobile device. Three tasks, v_3, v_5 and v_8 are dependent on task v_2. Moreover, the tasks v_7 and v_9 also depend on two different tasks. This model is general in nature, and can be used to describe any mobile application.

4.2. Formulation

We now use our system model to develop a formulation of the decision problem. This allows us to discuss how adaptation can be incorporated into this decision problem in later sections.

Let E^{tot} denote the total energy to execute the application. Also let E^{loc} and E^{mig} denote local execution energy and energy to migrate data between mobile device and cloud server.

Table 4: Adaptive offloading techniques along with parameters that are considered during adaptation. Each marked cell denotes that the offloading technique in the proposed offloading framework adapts to changes in that parameter.

Offloading solution	Year	Application			Network					Execution Platform					Cloud Management	
		Concurrency	Multi-tasking	Real-time Constraints	Bandwidth Variation	Intermittent Connectivity	Multiple Interfaces	Signal Handoff		Mobile Processors	GPU	Energy/Time Profile	Memory Architecture	Cloud Architecture	Resource Provisioning	Workload Management
Implementation-based Studies																
MAUI [31]	2010			✓	✓		✓						✓			
Misco [36]	2010	✓											✓	✓		
Odessa [84]	2011	✓		✓	✓		✓					✓	✓			
CloneCloud [29]	2011	✓		✓	✓		✓						✓			
ECOS [47]	2012	✓											✓	✓		✓
Serendipity [98]	2012	✓			✓			✓					✓	✓		
Yang et al. [114]	2013	✓			✓							✓	✓			
Abebe & Ryan [1]	2012	✓											✓	✓		
Kwon & Tilevich [63]	2012					✓							✓			
ThinkAir [60]	2012	✓			✓		✓					✓	✓			✓
Eom et al. [40]	2013	✓			✓	✓	✓						✓	✓		
ENDA [66]	2013				✓			✓					✓			
TDM [74]	2013	✓					✓				✓		✓			
COSMOS [97]	2014	✓			✓	✓	✓						✓			
Gao et al. [45]	2014	✓			✓							✓	✓			
Hermes [56]	2015	✓		✓	✓		✓						✓			
DREAM [62]	2015	✓	✓		✓		✓	✓				✓	✓			✓
Tango [48]	2015	✓	✓	✓		✓							✓			
Li & Gao [69]	2015			✓		✓							✓			
Simulation-based Studies																
Fesehaye et al. [43]	2012				✓		✓	✓						✓		
MapCloud [85]	2012	✓			✓											
Balakrishnan & Tham [11]	2013	✓		✓	✓					✓						
CARMS [57]	2013				✓									✓		
Chen et al. [21]	2013				✓	✓				✓						
Lin et al. [72]	2013	✓														
MuSIC [86]	2013	✓			✓			✓								
Mobile fog [50]	2013	✓			✓	✓	✓	✓						✓		
RMCC [4]	2014													✓		✓
Ready-Set-Go [110]	2014	✓	✓													
Niu et al. [80]	2014	✓			✓											
Foreseer [113]	2014				✓	✓		✓								
Chen et al. [22]	2014				✓	✓						✓				
Lin et al. [73]	2014	✓		✓	✓					✓						
Deng et al. [33]	2015	✓			✓	✓		✓								
Chen [24]	2015			✓												
Zhang et al. [118]	2015			✓		✓										
Sood & Sandhu [103]	2015	✓													✓	
MALMOS [39]	2015				✓			✓								✓
Tong & Gao [104]	2016	✓	✓	✓			✓									

Then, the total execution energy E^{tot} is the sum of local execution energy (E^{loc}) and migration energy (E^{mig}):

$$E^{tot} = E^{loc} + E^{mig} \quad (1)$$

Similarly, let T^{tot} denote the finish time of the application. Also, let T^{loc}, T^{ser} and T^{mig} denote the execution time on mobile device, time to execute on server and time to migrate data. Then, the finish time T^{tot} is:

$$T^{tot} = T^{loc} + T^{ser} + T^{mig} \quad (2)$$

The objective of the formulation is to minimize time and/or energy costs. The optimal solution of the various individual costs can vary. One common approach is to minimize one cost while applying some constraint on other costs. For example, a possible way is to minimize energy, while limiting the completion time:

$$\text{Min } E^{tot} \quad (3)$$

$$\text{subject to: } T^{tot} < D, \quad (4)$$

where D represents a time deadline by which the execution of an application must be completed.

We now explain how our problem formulation can be utilized to study adaptiveness of an offloading system. The values of E^{mig} and T^{mig} are affected by changes in network conditions. The values of E^{loc} and T^{loc} depend on the nature of the workload and concurrency within an application, and the architecture of the mobile device. Finally, T^{ser} is determined by the nature of the cloud server and cloud management.

Frequent sampling of these parameters at run-time for making offloading choices makes the decisions more adaptive. However, solving the problem formulation at run-time consumes time and energy. If all the parameters are varying, solving the problem formulation becomes impractical at run-time. Thus, to obtain a solution in acceptable time and energy, offloading systems do not handle all variations of the ecosystem. Instead, each offloading system handle some of the variations, while making some assumptions about other parameters. Table 4 summarizes adaptiveness of offloading solutions to the different parameters of the mobile cloud ecosystem.

5. Handling Application Variations

A large variety of mobile applications are run on mobile devices. The expectations of the user from each of these mobile applications vary. For example, a user expects an application having lot of interaction to have fast response time. However, for a computation-intensive application running in the background, it is more important to reduce its energy consumption. An offloading system needs to handle these multiple applications and sometimes the conflicting expectations from its users.

In this section, we discuss how offloading systems handle the variation in applications. Variation within applications affect all aspects of the user's quality of experience (QoE).

5.1. Degree of Concurrency

The amount of parallelism that can be exploited increases with an increase in the concurrency of an application. Thus, applications having higher amount of concurrency tend to give better performance using offloading [60]. However, the offloading decision problem for a sequential application is much more scalable. Moreover, there can be different types of concurrency in an application – at the task (or method) level, and at the data level. The type of concurrency available for an application also has an influence in the performance of an offloading system.

We first discuss the scalability issues related to concurrency. We then discuss how the type of concurrency influences the design decision of an offloading system.

Improvement in scalability has two major advantages. First, a highly scalable solution allows the offloading decision algorithm to run on the mobile device. Secondly, it allows the offloading system to take a more optimal decision. Both these advantages lead to energy and time savings.

The offloading decision problem is polynomial for sequential applications [73]. For such cases, the decision problem formulation can be resolved to the shortest path problem. This allows an offloading system to schedule larger applications in less time.

For applications having task-level concurrency, the offloading decision problem is known to be NP-Complete [106]. Thus, to build more scalable techniques for concurrent applications, efficient algorithms to partition such applications are required [30]. One heuristic used by ThinkAir is to compare the amount of computation involved for each thread and the migration costs [60]. If the local computation cost exceeds the migration cost, only then execution is performed on the cloud system using migration. Another work, Hermes gives a polynomial approximation algorithm for applications with limited amount of parallelism [56].

Applications having data-level concurrency have more flexibility in execution. For some applications, like computer vision, the output precision reduces with an increase in the number of threads. This reduction in precision is acceptable upto a certain level. Thus, the offloading system may decide the optimal level of concurrency. Odessa is an example of such an offloading system [84].

5.2. Workload Heterogeneity

Most offloading systems developed so far have discussed only the execution of a single application at a time. Smartphone devices have multi-tasking operating systems and usually run multiple applications at the same time.

Offloading systems need to model multiple applications for dealing with realistic use cases. This is because, the energy consumption involved in communication depends not only on the amount of data, but also on the time when the network interface was last used. This occurs because interfaces of radio networks remain switched on for a few seconds even after transmission is complete. A request for transmitting data takes less energy if the network interface is already on. However, if no such request is sent, then the energy required to keep the interface running is

Table 5: A summary of adaptation techniques used by offloading frameworks to handle application variations.

Variation Parameter	Offloading Framework	Adaptation Technique
Degree of Concurrency	ThinkAir [60]	Uses past execution history to modify partition of application graph
	Hermes [56]	Uses approximation scheme to generate partition of application graph
	Odessa [84]	Modifies the number of threads depending on workload
Workload Heterogeneity	Ready-Set-go [110]	Merges offloading data from multiple applications to reduce network tail energy
Real-time Constraints	Li et al. [67]	Modifies accuracy of face-recognition depending on power constraint

wasted. Experimental results show energy saving upto 60% by careful timing of communication requests when radio networks are used[94].

Studies have shown that finding the optimal solution for multiple applications is NP-hard [32]. One such study has proposed using coalesced offloading, where the algorithm tries to consolidate the network transmissions of different applications. Experiments on real-world use cases have shown energy saving of 21% over naive scheduling [110].

5.3. Real-Time Constraints

The amount of interactiveness and computation involved in an application largely determines how suitable it is for offloading to the cloud. This is because users can interact with an application only from the mobile device. Thus, with an increase in the amount of user interaction involved in an application, it becomes more challenging to offload applications [107] [79]. In order to satisfy such constraints, the offloading system has to bring back the state of execution to the mobile device in order to allow the user to interact with the application. This increases the number of migration from the mobile device to the cloud and vice-versa.

For applications having very little amount of user interaction, it is possible for the developer to partition the application. This is because, in such cases, much of the computation can be performed on the cloud. Applications for scientific computing are suitable for such use cases. Such applications, like Matlab Mobile[3], are available in the mobile applications market.

Applications with hard real-time constraints are more difficult to offload. For such applications, such as multiplayer games or augmented reality, the architecture of the cloud is an important factor [43]. This is because of the fact that the latency involved in migrating the tasks of such applications must be low. Such applications are not yet widely used along with cloud systems.

Applications with soft real-time constraints, such as streaming, can be augmented using offloading. Such systems usually monitor the network carefully to ensure that the available connectivity is able to support the required amount of communication. Such offloading techniques can even enable more complex applications, such as gesture recognition [84], face recognition [67] or peer-to-peer streaming [114].

The presence of real-time constraints makes it crucial to enforce privacy of communications. This is because offloaded applications with real-time constraints frequently migrate data back to the mobile device. This makes it easier to intercept the user's data used by applications.

5.4. Discussion

The workload on the mobile device differ based on the number and type of applications being run. Thus, for some applications, such as scientific computing, the amount of data transfer needed for offloading is fixed. Other applications, like face recognition or peer-to-peer streaming have more scope of reducing the amount of data transfer by compromising on accuracy of output. So far, each offloading framework handles a particular type of application in order to avoid the complexity of adapting to the different types of running applications.

One major challenge towards utilizing offloading for more applications is the presence of native applications. Many computation intensive applications, such as video players, web browsers and games have a lot of native code. The instruction set architecture and memory layout of mobile devices and the cloud servers are usually different. This makes it much harder to offload such applications with native code. Recently, Lee et al. [64] showed a way of offloading native code by utilizing a compiler that generates intermediate code for both mobile device and cloud server. At run-time, it translates the two types of machine code before migrating data. Using this technique, optimizing more computation-intensive applications has become possible.

Most studies have focused on utilizing offloading for one application at a time. From Table 4, we note that few studies have focused on offloading multiple applications. Executing multiple applications lead to higher utilization of processors and thus increases energy consumption on the mobile device. It also increases the amount of workload variation on the mobile device. Since smartphones run multiple applications, it is necessary for offloading systems to handle such use cases. Thus, better techniques of adapting to these cases are needed to handle different workloads.

6. Handling Network Variations

The communication channel is one of the most important components of a mobile cloud system. Mobile systems utilize wireless network interfaces, which have varying bandwidth. This makes it challenging to utilize the network channel to provide predictable quality of experience (QoE). In this section, we discuss handling of these challenges by offloading systems.

[3] http://www.mathworks.com/mobile/

Table 6: A summary of adaptation techniques used by offloading frameworks to handle network variations.

Variation Parameter	Offloading Framework	Adaptation Technique
Bandwidth Variation	MAUI [31], CloneCloud [29]	Detects bandwidth before offloading
	Foreseer [113]	Modifies static partition at run-time
	Niu et al. [80]	Set bandwidth variable value at run-time
	Wang & Dey [108]	Bit-rate modification at run-time based on bandwidth
Intermittent Connectivity	MAUI [31]	Periodically checks connectivity of mobile device with server
	Kwon & Tilevich [63]	Checkpoints applications at fixed points
	COSMOS [97]	Predicts loss of connectivity and checkpoints applications
Multiple Interfaces	Einsiedler et al. [38]	Allows network operator to choose interface based on congestion
	Qato [52]	Uses Wi-Fi Direct to find best network interface among multiple mobile devices
Signal Handoff	ENDA [66]	User position prediction to modify the decision design
	MuSIC [86]	Decides the server platform for mobile device based on predicted user position
	Deng et al. [33]	Assumes random movement of mobile device to predict signal strength
	CloudNet [93]	Migrates VM without any user downtime

6.1. Bandwidth Variation

In order to deal with varying bandwidth, MAUI and CloneCloud detect the bandwidth by sending data packets before offloading. This bandwidth is then used to calculate the components that should be offloaded to the cloud. The drawback of this approach is that bandwidth variation in the middle of execution cannot be handled. Thus, the application performance becomes unpredictable if the bandwidth changes suddenly.

Bandwidth variation in the middle of execution is handled by Foreseer [113]. Foreseer also uses static partitioning like MAUI and CloneCloud. However, it makes modifications to the static partition at run-time based on the bandwidth variation. Although this technique has not been tried in real implementations, simulation results suggest energy savings of over 35% compared to static partitioning techniques.

Niu et al. [80] propose using a different technique to handle bandwidth variation. Their work performs static partitioning, but uses the bandwidth as a variable. By changing the value of the bandwidth variable, it is possible to generate different partitions. At run-time, it senses the bandwidth value and partitions the graph by setting it to the right value.

Streaming applications can gracefully degrade in response to lower bandwidth by lowering their bit rate [108]. Thus, offloading of these applications are more tolerant to varying bandwidth.

6.2. Intermittent Connectivity

It has been observed that wireless network interfaces are prone to loss of connectivity [97]. This is due to variation in the number of users, and also mobility of the user. Loss of connectivity in the middle of execution may force execution from the beginning.

Many studies have tried to improve the tolerance of offloading systems to loss of network connectivity. MAUI periodically checks the mobile device's connectivity with the server after migrating [31]. If connection is lost, it waits for a pre-specified time to begin local execution. The actual time that the offloading system should wait is an important factor determining the execution time in case of failure. Xu et al. [112] show that using passive measurements of network connection, it is possible to estimate the probability of network failure. Park et al. [83] show that the loss of connectivity can be explained by the mobility pattern of the mobile device. It uses a Markov Model to decide the likelihood of losing access to the server. Kwon & Tilevich [63] suggest intelligent checkpointing of methods in order to make the execution more tolerant to loss of connectivity. Finally, these techniques are utilized in the offloading system COSMOS [97].

Although these techniques handle loss of connectivity, they reduce predictability of performance of the offloading system. Thus, energy consumption and execution time may increase more than expected. They do not provide acceptable performance for applications having high user interaction [79].

6.3. Multiple Interfaces

Modern mobile devices contain multiple network interfaces, such as Wi-Fi, EDGE, 3G and LTE. Most offloading systems currently available do not explicitly select the network interface. Instead, the mobile operating system picks the interface with the strongest signal, or with the highest bandwidth. For example, if access to Wi-Fi is available, then the offloading system prefers Wi-Fi to 3G because Wi-Fi offers better bandwidth.

Mehmeti & Spyropoulos [78] show that efficiency of many of the offloading tasks can be improved by waiting for access to Wi-Fi. Thus, much of the traffic from cellular network can be diverted to Wi-Fi interface, leading to much higher energy savings. However, this method may increase the completion time of applications. Background applications, where completion time is not an important objective, may utilize this technique.

Another proposed technique of dealing with varying interfaces is to allow the network operator to determine the appropriate interface [38]. Network operators can choose an interface based on the amount of traffic, the type of service needed by the user and amount of congestion in the network. This allows a more efficient usage of the available network resources. Moreover, based on the user's requirements, the network operator can even allow two different interfaces to be used at the same time. However, the gains are obtained at the cost of flexibility for the user.

A similar technique of utilizing the best available network interface is shown in Qato [52]. In this system, several mobile devices in the vicinity can collaborate through Wi-Fi Direct to find the network interface having the least congestion. This interface can then be used for offloading by all the devices in the vicinity. The challenge in this case is to develop sufficient economic incentive so that users find it beneficial to use this facility.

One development likely to have major impact on future offloading systems is the development of 5G network interface. 5G provides a parameter known as edge rate, which is the minimum bandwidth guaranteed to each user [9]. This makes predicting bandwidth much easier, and is likely to enable better handling of real-time applications using cloud systems.

6.4. Signal Handoff

One serious challenge that has hindered the adoption of offloading is handling user's mobility [68]. Mobility of the user leads to signal handoff and makes the connectivity unstable.

Much of the work related to managing signal handoffs involves predicting the user's mobility patterns. This can then be used to decide the server to which the application should be offloaded. Bartendr showed that user track prediction is possible by observing prior movement of the user [94]. ENDA utilized user prediction to optimize the decision design of its offloading system [66]. MuSIC showed that if the user's mobility pattern are known in advance, it is possible to optimally decide the best server platform [86].

An alternative way of handling signal handoff is to assume random movement of mobile devices. Based on this assumption, the signal strength within a particular region is used to obtain the data rate. This data rate can provide a better estimate of the time required to offload a particular task. This technique is used in Deng et al. [33].

The approaches discussed so far do not deal with unpredictable user movement. Moreover, there is no guarantee that the same network bandwidth is available whenever the user changes position. For robust mobility support, these challenges must be handled.

Offloading systems with multiple layers of cloud must handle another problem associated with mobility. For such systems one layer of computation resources, known as cloudlet, is usually closer to the mobile device. If the user moves away from it, the virtual machine (VM) should be migrated. CloudNets proposes a technique of handling such migrations without any user downtime [93]. In this technique, known as live migration, the data is copied from one cloud to another in the background. During this process, the VM goes on providing service to the user.

6.5. Discussion

The variation in network performance hurts the quality of experience (QoE) of the user. A lot of techniques have been explored to mitigate the effect of such variation. These techniques try to predict network performance and alter the behavior of the offloading framework accordingly. Such prediction is usually challenging due to the interplay of various factors, such as network congestion, type of network used and movement of user.

An alternative to network prediction is proposed in Tango [48]. In this technique, the application is executed both on the mobile device and the cloud server. The output from the replica that comes first is forwarded to the user, while the other replica's output is suppressed by the offloading framework. Using this technique, it is possible to avoid the complex process of network prediction and runtime modification of graph partition. The disadvantage of this approach is that this does not guarantee any energy saving.

A technique that is commonly used to mitigate the impact of network variation is reduction of migration data. Careful selection of data by optimizing the offloading framework can reduce the amount of migration data [69].

One development that is likely to have a major impact on mobile cloud systems is Software-defined Networking (SDN) and network virtualization [28]. SDN enables developers to program the routing algorithms through software. Thus, developers can route network packets based on the requirements of the application. The study me-SDN (Mobile Extension of SDN) proposed extending the SDN paradigm to mobile devices in order to make the network traffic application-aware [65]. This allows it to better handle network traffic having real-time constraints. However, so far it is not possible to provide any performance guarantee using me-SDN.

An interesting factor involved in managing these variations is the monetary cost that the user is willing to spend. A more predictable network can be obtained by the user by spending more on network. For example, offloading performs better using LTE than under 3G, but LTE costs more. Similarly, if the user pays for better cloud service, it is possible for the cloud service provider to reduce the server latency of offloading requests.

We also note from Table 4 that most studies do not consider signal handoff. However, user mobility leads to signal handoff and bandwidth variation. Thus, adaptation to user mobility is essential. So far, most studies that handle user mobility are based on simulation. These studies predict the position of the user based on past user positions. More real-world experiments using offloading systems are needed to validate these studies, and provide support for signal handoff.

7. Handling Execution Platform Variations

The underlying architecture of the offloading system is an important consideration in the design of system architecture. Moreover, the architecture of the mobile devices as well as that of the cloud system is constantly evolving. This makes it even more important for an offloading framework to adapt to different systems available in the market.

7.1. Mobile Processors

Modern smartphones have up to 8 processors [82]. An increase in the number of mobile processors reduces the completion time of an application, provided it has sufficient parallelism. However, it increases energy consumption.

Table 7: A summary of adaptation techniques used by offloading frameworks to handle execution platform variations.

Variation Parameter	Offloading Framework	Adaptation Technique
Mobile Processors	Balakrishnan & Tham [11]	Uses an optimization solver to change frequency levels
	DREAM [62]	Sets frequency level using Lyupanov optimization to minimize energy
GPU	TDM [74]	Offloads tasks for GPU to server based on data size
Energy and Time Profile	ThinkAir [60]	Uses energy model to estimate energy of individual tasks
Memory Architecture	MAUI [31], CloneCloud [29]	Uses intermediate language to hide memory architecture differences
	Native Offloader [64]	Generates two application binaries to handle memory architecture differences
Cloud Architecture	Zhang et al. [120]	Uses both cloudlet and cloud server
	CARMS [57]	Uses user location to determine best combination of cloud resources
	Abebe & Ryan [1]	Adapts execution partition depending on available cloud resources
	CDroid [14]	Tightly integrates mobile device with cloud server
	Misco [36]	Offloads data-intensive tasks to nearby mobile devices
	RMCC [4]	Uses centralized controller to distribute mobile tasks
	Mobile Fog [50]	Proposes a programming model to offload to multiple devices

The first offloading systems developed assumed a single processor system [31, 29]. Recent works have shown that modifying the algorithm to optimize the completion time for multiprocessor systems is possible [11]. However, the effect on energy consumption is still not clear due to lack of a relevant energy model.

In order to limit the amount of energy consumption, mobile processors now have an inbuilt scheme of running at different frequency levels. This is known as Dynamic Voltage and Frequency Scaling (DVFS). In this scheme, a processor that runs at lower frequency takes more time to execute a task, but consumes less energy. It allows the application scheduler to suitably decrease the energy consumption at the cost of higher completion time.

An offloading system must determine the number of frequency levels that each mobile processor supports. It needs to have an algorithm that can adapt to the possible frequency levels. Balakrishnan & Tham [11] propose an optimization formulation to partition the application considering the possible frequency levels. Chen et al. [21] show that the battery discharge rate depends on the interaction between the strength of wireless signal and the DVFS level of processors. DREAM uses Lyupanov optimization to determine the frequency level that minimizes energy consumption while keeping the number of waiting tasks below a fixed level [62]. Thus, the energy consumed by these components must be considered together in an offloading system.

7.2. GPU

Modern mobile systems have an inbuilt Graphics Processor Unit (GPU) in them. This enables faster processing of applications with data-level parallelism such as image processing or video decoding.

The relationship between energy consumption and completion time in the presence of GPU is shown in Ternary Decision Maker (TDM) [74]. This work shows that for moderately computation-intensive jobs, local GPU processing can consume less energy than offloading to the cloud. Thus, the algorithm used by the offloading system must consider the presence of a local coprocessor.

7.3. Energy and Time Profile

The energy consumed by the execution of an application is affected by the type of hardware used. This includes the type of processor used, the presence of the coprocessor, the type of network interface and the capacity of the battery. For an offloading system to take correct decisions, it must be able to adapt to these changes in hardware without any manual involvement.

Most offloading systems, such as MAUI, CloneCloud or ThinkAir, depend on *energy models* to determine the energy consumption of an application. These energy models compute the energy consumption by looking at either the source or the intermediate code of the application. Thus, these systems developed their own energy models using specialized high-precision power monitors. However, this is only feasible on a limited number of different mobile devices. For wide use of offloading, a more automatic system of estimating energy is needed.

Studies have shown that an accurate energy model can be developed automatically. The tool PowerBooter develops an automatic energy model for each phone [116]. It calibrates the energy model by running a specific hardware component on the mobile device over a period of time. The difference in voltage can be used to compute the amount of energy consumed by looking at the discharge rate curve available in the battery's manual. While this can handle the variation in mobile hardware, the power model for each battery has to be calibrated manually.

Sesame shows a method of handling multiple batteries automatically [35]. It uses the registers exposed by the battery's interface to measure the remaining energy. This can be used to study the applications running, and to find their energy profile over a period of time. In this way, an energy model can be built automatically for each mobile device and battery.

One problem with these profiling techniques is that they do not consider the impact of user input on the execution pattern. The user input plays an important role in determining which methods of an application get executed. Thus, it has a major impact on energy consumption. Gao et al. [45] model these variations using a semi-Markov model to arrive at a more accurate estimate of the energy consumption.

7.4. Memory Architecture

The memory architecture, such as layout and endianness, can vary between mobile device and cloud server. An offloading framework must handle such differences for it to be usable across multiple mobile devices and servers.

The most common way of handling such variations is to use virtual architectures such as Dalvik VM [29] or Microsoft .NET Common Language Runtime [31]. This hides the architectural variations of the two systems and lets the developer use a cloud server without any additional software engineering. However, much of the computation-intensive code of common applications such as video decoders or web browser is written in native code (C or C++). Native code cannot be offloaded using this method. If access to source code is available, one proposed way is to generate two distinct versions of the code [64].

7.5. Cloud Architecture

We define cloud architecture as the type and network layout of computing resources, such as user desktops, other smartphones in the vicinity and cloud data centers, that are available to the smartphone. For example, a user desktop usually consists of a single multi-core system, and is usually close to the smartphone. A public cloud data center has multiple processors, with low latency among them, but is much farther from the smartphone. Thus, the type of computing resource used has a major effect on the application execution time.

There is a trade-off among computation power of processors and network latency. This is because processors closer to the smartphone, such as desktop systems, have lower computation power compared to the ones that are farther away, such as data center servers. The rapid increase in the number of available computing devices makes offloading to them a feasible option. This is especially true for latency-sensitive applications, where increase in latency has a major impact on quality of experience (QoE). Thus, reducing latency by offloading to nearby devices but with less computation power might be more attractive than offloading to cloud data centers.

One proposed way of lowering latency is to maintain computation resource closer to the mobile device, known as cloudlet. Unlike a cloud server, a cloudlet is located at a one-hop distance from the smartphone. However, it has less computation power compared to a cloud server. Cloudlets are attached to wireless access points to enable easy access from mobile devices.

In order to handle applications with real-time constraints, multiple hierarchy of cloud systems has been suggested. This includes using a combination of cloudlet and cloud systems. Satyanarayanan et al. [92] show that cloudlets can help applications improve performance in hostile environments, such as war and disaster-relief. Zhang et al. [120] show that the QoE can be improved using both cloudlets and cloud systems as compared to just one of them. MAP-Cloud proves that finding the optimal solution to determine the execution points of components is NP-hard [85]. It proposes some heuristics that can be used to partition the application to save energy. CARMS uses locationing to select the best combination of cloud resources [57]. However, these approaches require installation of additional infrastructure, since cloudlets are not readily available on wireless access points [1].

A multi-level hierarchical mobile cloud system requires a relevant pricing model. So far, most studies have not considered the cost that the user has to pay to the cloud service provider. For widespread adoption, a pricing scheme acceptable to both the user and the cloud service provider is needed.

Another technique of reducing latency is utilized by CDroid [14]. CDroid tightly integrates the cloud server to a mobile device. It sends the state information of an application when the mobile device is idle. When a computation-intensive request is received by the mobile device, most of the information required to handle is thus already available on the server. In this way, it utilizes a dedicated server to cache application state information in order to reduce the amount of data transmission.

One offloading technique proposed is utilization of other mobile devices available nearby. The advantage of this approach is that at a particular point, a large number of mobile devices are usually available. Misco offloads data-intensive applications in order to enable data-level parallelism [36]. Serendipity implements fine-level offloading at task level to nearby mobile devices to speed up execution [98]. ECC investigates the devices to which offloading is attractive based on their proximity, current situation of battery and processor, and user objective [15]. RMCC suggests having a centralized controller to allocate tasks across mobile devices in the vicinity [4].

Offloading to other mobile devices provide a relevant economic incentive to the users of the remote devices. Execution on the remote devices consumes energy, reducing their battery life. One study has shown using mathematical modeling that cooperation among remote mobile devices improves the battery life of all of them. Chilipirea et al. [26] propose using a monetary incentive scheme where some mobile users may specify a selling price for utilizing their computation resources. The user who needs access to remote resources can specify the maximum buying price. If this condition is satisfied, then the mobile device offloads to the device offering the lowest selling price. These techniques assume that the same mobile devices are always available in the system.

Although utilization of other mobile devices has potential, it carries a high security risk. This is because, it is possible for a malicious remote device to permanently store the user's data. It might also be possible for the remote device to make the mobile application more vulnerable to security threats. Abolfazli et al. [5] suggest sandboxing and signing of the remote system to detect any modifications. However, the security of such a system has not been tested yet.

A recent proposal suggests offloading to network devices such as routers and switches. This is known as fog computing [16]. This does not require new infrastructure, and also has low latency. However, the challenge is to handle the very high level of heterogeneity among the different network devices. Mobile fog proposes a programming model to enable fog computing for mobile devices [50]. The primary challenge of using fog computing is in the handling of architectural heterogeneity. Conventional frameworks that support architectural

heterogeneity like openCL are not supported by network devices [40]. So far, no offloading system is available that utilizes fog computing.

7.6. Discussion

In this section, we explained the different approaches of handling variation in underlying system architecture. A number of techniques, such as automatic power model generation, has been developed to handle such variation. However. offloading frameworks do not leverage many of these possible techniques. As shown in Table 4, offloading frameworks avoid the complexity of adapting to different mobile architectures by studying its working on one execution platform. For example, few of the available offloading systems consider the impact of GPU on the offloading decision problem. They also do not handle different mobile processor systems. Further research is needed to understand the impact of such hardware differences on the QoE.

There is also increasing focus on inter-operation among different cloud systems. So far, there are few offloading systems that support utilization of multiple cloud systems. For widespread adoption, offloading systems need to handle these variations in both mobile and cloud systems. More studies are required to understand the behavior of offloading systems to such changes in execution platform.

8. Handling Cloud System Management Variations

There are many different cloud service providers with a variety of policies on cloud management. The management of resources directly affect the response time of cloud servers. It also affects the monetary cost of execution, since cloud service providers charge a cost to the users based on the amount of resources used. The security and privacy policies of cloud service providers also affect the security of mobile users.

8.1. Resource Provisioning

Cloud resources include hardware resources such as processors, memory and storage, software resources such as Operating Systems (OS) and APIs as well as power. Resource provisioning policies directly affect the response time of the cloud server and the monetary cost of computation. For example, increasing the number of processors and amount of memory made available to a user reduces the response time of the cloud server. Similarly, cost of energy direct affects the monetary cost of using cloud service. Sometimes, users may also have to pay to use OS and APIs.

A lot of research has been done to deal with resource provisioning for different types of cloud workloads. For mobile workloads, OSullivan & Grigoras [81] present a cost model based on the number of VMs used and the amount of communication data. Sood & Sandhu [103] propose a technique of estimating the amount of cloud resources needed for a mobile device. It checks the mobile device's Operating System (OS), number and nature of installed applications and network condition of mobile device to initially allocate cloud resources. When an application is running, it uses a neural network to predict the amount of cloud resources that will be required by the offloading framework. This helps the cloud server use the resources available to serve more users.

8.2. Workload Management

Cloud servers need to serve multiple mobile devices. Depending on the number of mobile devices being served by a cloud server, the response time of a cloud server changes.

The first offloading system that handled the problem of varying load on the server was ThinkAir [60]. ThinkAir utilized the current latency to decide whether to offload. However, this approach cannot handle increase of server load during execution. This may increase the total execution time of the application.

Loadsense showed that the latency of cellular networks is primarily dependent on the load on the base station [17]. Using passive measurements of the total power emitted, and using clustering techniques to compare the different latency values, it was able to obtain an estimate of the total network latency. This can be used to improve by offloading systems to estimate latency. Chen [24] designed an algorithm based on game theory to calculate the best possible time to offload a task.

Other works focused on managing the workload of the cloud server. One way is to vary the amount of approximation in streaming applications [99]. The amount of energy consumption and execution time reduce with an increase in the amount of approximation. Thus, the increase in latency is compensated by reducing the execution time through variation in streaming quality.

The concept of approximate computing can even be extended to handle varying network latency. Ravindranath et al. [87] showed that a server can study the network condition to obtain an estimate of the network latency. Based on this, the authors proposed setting a response time target of each task, and adjusting the approximation level accordingly. This allows a more consistent and predictable execution of applications with real-time constraints.

The problem of high and variable server load is especially serious for platforms with lower computation power such as cloudlets. For such cases, Hoang et al. [49] proposed restricting the number of users who can access such a resource based on the computation request received from the mobile device. This supports a better QoE to users who have already started executing their applications using the cloud system.

ECOS proposes using time-sharing of communication among multiple users to handle server load [47]. This allows many users to utilize the same server system while communicating only during fixed time periods. Time-sharing of network traffic was implemented using software-defined networking.

8.3. Discussion

The amount of cloud resources allocated and the workload affects the performance of offloaded mobile applications. However, since application offloading has yet to be in mainstream use, there is no real implementation to understand the effect of offloading from multiple devices on cloud resources. Thus, resource provisioning for offloading applications has still not

Table 8: A summary of adaptation techniques used by offloading frameworks to handle cloud system management variations.

Variation Parameter	Offloading Framework	Adaptation Technique
Resource Provisioning	OSullivan & Grigoras [81]	Proposes a cost model for mobile workloads
	Sood & Sandhu [103]	Allocates cloud resources by predicting mobile workload
Workload Management	ThinkAir [60]	Uses server latency value to determine server load
	Ravindranath et al. [87]	Varies the level of approximation to adapt to latency requirement
	Hoang et al. [49]	Restricts number of users to cloudlet to reduce workload
	ECOS [47]	Time-sharing of communication across multiple users

been widely explored. For mainstream use, efficient provisioning is necessary to ensure predictable performance and monetary cost. Another key issue not explored yet is pricing of cloud servers for mobile workloads. So far, there are no studies to find out the monetary cost that users are ready to pay for improving mobile application performance.

9. Impact on Quality of Experience (QoE)

The variations in the environmental parameters affect the quality of experience (QoE) of the user. Although the objective of the offloading system is to maintain high QoE in a difficult environment, this is not always possible. In this section, we study the parameters that affect the QoE of the user. We also briefly discuss how the importance of different QoE parameters may vary depending on the context.

Table 9 shows the impact of different parameters on user's QoE. While discussing the effect on each QoE component, we have assumed that other parameters remain constant. Moreover, we also assume that the offloading system considers only one component of the QoE at a time.

9.1. Energy Consumption

We first discuss the effect of the application component on energy consumption. Increasing the amount of concurrency and multitasking allow higher energy savings by making it easier to offload software components. However, increase in real-time constraints increases amount of local execution, and reduces energy savings.

The network component of the ecosystem is also an important factor influencing energy consumption. Higher bandwidth reduces the energy cost of transmission, thus saving energy. Intermittent connectivity leads to checkpointing of applications, and more local execution, thus leading to lower energy savings. Latency has no direct effect on energy. However, latency can increase the energy consumption indirectly by increasing the execution time. Network interfaces have different energy consumption patterns, and thus the energy saving varies. More signal handoffs increases energy consumption and lowers energy gain by increasing the cost of hand-off from one base station to another [12].

The execution platform also affects the energy consumption. Increase in number of mobile processors increases the energy cost, and lowers energy savings. Intelligent use of Dynamic Voltage and Frequency Scaling (DVFS) can lead to higher energy savings, by allowing a processor to run at lower speed, while offloading more tasks. Having a coprocessor also leads to lower energy savings, since running it increases the energy consumption. Having a higher energy profile gives the offloading system more scope to offload, and leads to higher energy savings. A cloud architecture having more computation resources nearby reduces the energy cost of transmission, and thus leads to higher energy savings.

Management of the cloud server can also impact energy savings. Having sufficient resources on the cloud server is essential to execute tasks, and thus increases energy savings. However, variation in workloads does not directly impact energy savings, since we only consider the energy savings of mobile battery.

9.2. Completion Time

The completion time is affected by the number and type of applications given by the user. Higher concurrency and more multitasking allows better utilization of parallelism, leading to greater time saving. Real-time constraints force the offloading system to schedule the tasks in a way that satisfies the constraints. This may lead to higher or lower completion times, depending on the condition of the channel in the offloading system.

The network condition has a major impact on completion time. Higher bandwidth and lower latency lead to faster transmission of data, and thus greater time saving. Intermittent connectivity leads to loss of contact with the cloud system, leading to more local execution and thus lower time saving. Some network interfaces inherently provide higher bandwidth and lower latency, thus affecting the completion time. Higher user mobility (signal handoffs) forces the cloud system to migrate data across different servers, in order to better support the user. This leads to lower time saving.

The execution platform also affects the completion time. Increase in the number of mobile processors leads to faster execution, but decreases the scope of offloading, and thus reduces time saving. A similar result is obtained by adding a coprocessor on the mobile system. However, DVFS has no effect on the completion time, unless the offloading system optimizes energy consumption. More computation-intensive jobs gives the offloading system more scope to offload, leading to higher time saving. A cloud system with more layers and having some layers closer to the mobile device reduces latency and thus leads to higher time saving.

Efficient cloud management techniques are important to ensure fast completion times. Fast resource allocation allows lower setup delays and leads to faster execution. Similarly, efficient

Table 9: Summary of parameters that affect Quality of Experience (QoE) for a user.

Source of Variation	Environmental Parameter	Energy Consumption	Completion Time	Monetary Cost	Security
Application	Degree of concurrency	✓	✓		
	Workload Heterogeneity	✓	✓		
	Real-time Constraints	✓	✓	✓	✓
Network	Bandwidth Variation	✓	✓		
	Intermittent Connectivity	✓	✓		
	Network Interface	✓	✓	✓	✓
	Signal Handoff	✓	✓		✓
Execution Platform	Mobile Processors	✓	✓		
	GPU	✓	✓		
	Energy/Time Profile	✓	✓		
	Memory Architecture		✓		
	Cloud Architecture	✓	✓	✓	✓
Cloud Management	Resource Provisioning	✓	✓	✓	
	Workload Management		✓	✓	

load-balancing of workloads across cloud servers also leads to faster response times and thus faster execution.

9.3. Monetary Cost

User surveys suggest that controlling monetary cost incurred by mobile software is the greatest concern of users [41]. Thus, an adaptive offloading system must control the monetary cost.

The application component of the mobile ecosystem has a significant impact on the monetary cost. Utilizing higher amount of concurrency and more multitasking require more server processors, thus increasing the server cost. Having more real-time constraints increases the amount of communication, leading to higher bandwidth cost.

The monetary cost of offloading is affected by only the network interface used by the offloading system. This is because network interfaces have different pricing models. For example, since 3G has lower bandwidth and higher latency than LTE, data transfer over 3G usually costs lower. Thus, using an interface that provides better service increases the monetary cost.

Among the many parameters of execution platform, only the cloud architecture affects the monetary cost. This is because hiring more computation resources is expensive. Thus, the cost goes up with an increase in more layers of cloud system.

Cloud management plays an important role in determining the monetary cost. The cost of acquiring resources is dependent on the type of resource allocation techniques used. Moreover, the resources used to allocate the workloads across multiple servers also has a monetary cost.

9.4. Security

Mobile devices usually contain a lot of private data of users. Moreover, ensuring the integrity of the data that has been computed remotely is also important. Thus, security forms an important component of the QoE.

The type of application has an impact on privacy of data. Presence of more real-time constraints in the application increases the number of migrations from the cloud to the mobile device. This makes the system more vulnerable to a man-in-the-middle attack, where the attacker alters the packets during transmission.

The variations in the parameters of the network component also affect security. Some network interfaces, like Wi-Fi, are more vulnerable to man-in-the-middle attacks, as compared to 3G or LTE [111]. More signal handoffs increase security by making it more difficult to sniff data [105].

Only the cloud architecture in execution platform component has an impact on security. A malicious remote computation resource used by the mobile device for offloading can subtly modify the execution of the application. This can affect the integrity of the running application, making the system vulnerable to attacks.

9.5. Discussion

Offloading systems currently focus on one or two QoE metrics. However, the expectations of users vary depending on the context in which they use the mobile device. For example, if the mobile device is at the user's home, then the user has access to a charging point. In such a scenario, energy consumption is not an important component of the QoE. Similarly, for background applications, completion time is not considered to be an important consideration. In this way, the importance of energy consumption and completion time vary depending on the context in which an offloading system is used.

An offloading system can utilize such context-awareness to better optimize the relevant QoE parameters. Such a context-aware offloading system was proposed in Lin et al. [72]. In this study, the importance of energy consumption as part of the QoE was decided based on the previous values of inter-charging intervals. A mobile device having higher inter-charging interval requires more energy saving, and thus energy consumption becomes a more important component of QoE. The objective of offloading can be modified to increase the importance of energy saving in such cases.

User expectations also affect the monetary cost. A user running more energy or computation-intensive applications needs more support from remote computation resources. In such a case, the user might be willing to pay a higher monetary cost to save time and energy. In this way, user expectation of energy and time affects the monetary cost.

Users may have different expectations of security depending on the type of applications and sensitivity of data. Supporting security also affects other parameters of QoE. Providing better security increases the overhead of encryption and decryption, and thus requires higher reservation of resources [70]. In this way, a careful balance must be made between providing enough security and optimizing other parameters of QoE.

Handling these different user expectations is important for adoption of offloading. However, current offloading systems do not consider the varying expectations of users. Further work to explore ways of using such context-awareness is needed to better support users.

10. Open Issues and Future Directions

In this section, we summarize the challenges and open issues that still remain towards adoption of application offloading. We have seen so far that offloading frameworks currently adapt to some of the parameters, and focus on improving only a few Quality of Experience (QoE) parameters. Adapting to all the diverse variations possible in the environment, while balancing different parameters of QoE still remains a major challenge. In the rest of this section, we discuss the most important parameters to which offloading frameworks need to adapt to enable their mainstream use.

10.1. Leveraging Heterogeneity in Cloud Architecture

One major factor that affects the performance of offloading frameworks is the latency of accessing cloud servers from smartphones. Currently cloud computing resources are mostly in the form of large data centers. While data centers have large computing and storage resources, they depend on the Internet for connecting with mobile devices. Thus, managing latency expectations is usually challenging. Moreover, many computation intensive applications on mobile devices are latency-sensitive.

One of the most important factors affecting latency is the distance of the server from the smartphone. Moving cloud resources closer to mobile users lowers network latency and enables faster response. Such resources may include the user's own desktop, laptop, set-top boxes, routers, tablets or special-purpose hardware devices. This is variously referred to as fog computing [16], mobile edge computing [37, 54] or edge-centric computing [46]. So far, frameworks such as Hyrax [77], virtual mobile cloud computing provider [53] and RMCC [4] that utilize other mobile devices in the vicinity have been used to speed up applications. Another possible approach is utilizing computing resources provided by telecom operators [115]. However, utilizing such varieties of computing resources depending on their availability remains an open issue .

10.2. Ensuring Predictability of Application Performance

Most offloading frameworks developed so far focus on showing the improvement in performance over a limited number of benchmarks [102]. However, the impact of offloading on performance of applications differ based on their workloads and characteristics. Implementing a single offloading framework that can manage all applications is complex due to the unpredictable ways applications interact with the operating system.

One possible way of resolving this problem is to develop an emulation framework that can study the performance of offloading. Emulation frameworks have been shown to be effective in network protocol testing, and can bridge the gap between implementation and simulation based offloading works [55]. An offloading framework emulator will allow designers to propose new solution techniques and validate their approach without complex implementations.

10.3. Adapting to Workload of Multiple Users

An important challenge of utilizing offloading is to provide acceptable performance under different load. The performance of offloading depends on the quality of wireless network, and the availability of server resources. Both these factors are affected by the number of users. Managing these variations requires both better guarantee of service from network providers and cloud service providers.

Network service providers are gradually moving towards better QoE guarantees. For example, the cellular protocol currently under development, 5G aims to guarantee 100 Mbps worst data rate, as compared to 1 Mbps offered by LTE [9]. Similarly, the 802.11e standard for Wi-Fi contains contention-free provisions for time-critical data [76]. These techniques can be utilized to provide performance guarantees to mobile applications.

Managing QoE expectations while avoiding over-provisioning of resources is a major area of researching in cloud computing [10]. A promising way of managing such requirements is to develop better mathematical models of workloads. Mathematical models allow the cloud service providers to better predict user expectations and provide adequate computing resources, such as servers and data storage. We discussed one approach of estimating the amount of resources needed on the cloud server in Section 8. However, it does not take into account the presence of multiple users. To reduce cost for cloud service providers and help provision adequate resources, formal mathematical models considering different workloads of users are needed.

10.4. Making Offloading Frameworks Context-Aware

An offloading framework has to balance multiple user objectives. These user objectives are often dynamic and depend on the context in which the mobile device is used. Moreover, available cloud resources also vary depending on user location.

A possible way of handling such cases is to make the offloading framework context-aware. Context-aware systems adapt their operations to the current context without explicit user intervention [13]. This improves the usability and performance of a system. For example, a mobile device can select a better cloud service provider if it is aware of its location.

A variety of techniques have been explored to make a mobile system context-aware. Mobile systems now contain accelerometers which allow the system to detect the user's mobility pattern. Similarly, the residual capacity of the battery can be used to determine if energy-intensive tasks should be offloaded. Integrating such techniques into offloading frameworks remains an open research problem.

10.5. Ensuring Interoperability Across Cloud Service Providers

Users may prefer to use services from multiple cloud service providers to ensure greater reliability, greater availability of cloud resources and faster performance depending on their location. Seamless moving from one service to another requires a mechanism of communication data across multiple cloud server systems.

Current offloading frameworks do not look at cloud interoperability. However, a framework CloudNet showed that migration of VM from one cloud system to another is possible without any downtime for the user [93]. A key requirement for cloud interoperability is to have widely used standards for communication across multiple service providers. Thus, efforts towards developing protocols and standardization of communication are needed to better serve users.

10.6. Managing Multi-tasking Environments

Recently multi-tasking has become very popular on smartphones. Multi-tasking refers to simultaneous execution of multiple applications on the smartphone. Multi-tasking presents a major challenge for offloading frameworks, since it is difficult for the offloading framework to predict when the user may switch applications. It may also lead to a queue of offloading requests in the smartphone from the different applications.

So far, offloading requests from multiple applications has been studied in Ready-Set-Go [110]. This study used a simulation of offloading requests from multiple applications to show how each particular offloading request can be scheduled. However, the effectiveness of offloading with multiple applications has not been studied in actual multi-tasking environments. Efforts to extend offloading to this scenario are needed to ensure good Quality of Experience for users.

10.7. Meeting Application-specific Objectives

There are some application-specific optimizations that can further improve user experience. For example, Hu et al. [51] showed that performing some additional computation on the smartphone can reduce the amount of data transfer and energy in the case of object detection from video [51]. Similarly, Odessa increases the amount of parallelism when it needs to improve throughput in gesture recognition algorithms [84].

Current offloading frameworks usually model all applications in the same way. While this simplifies the modeling of the problem, this does not give the full benefit of offloading to the user. A major challenge that still remains is to ensure that application-specific optimizations can be performed by a single offloading framework.

11. Conclusion

In this paper, we survey adaptation techniques utilized for code offloading in computation offloading systems. Mobile systems are used in a changing and unpredictable environment. Thus, to enable widespread adoption, offloading systems need to be adaptive to such an environment.

We first identify that an offloading system is influenced by parameters from three categories - applications characteristics, network properties, and execution platform features. Several parameters within each category are described that influence the benefits of using the offloading framework. We also explained the methods used by offloading systems to adapt to such variations, and their impact on the quality of experience (QoE) of the user. We summarized our findings in Table 4.

Although research in offloading systems has made a lot of progress, they are still not adaptive to the entire range of parameters that influence the operating environment. Using Table 4 as a reference, we identified variations in different parameters which current offloading systems do not handle. We suggested more effort towards offloading of multiple applications. We also showed that offloading systems need better adaptation techniques to handle user mobility. Moreover, they need to adapt to a variety of different mobile and cloud systems to support more users. Increasing adaptiveness to these environmental parameters will enable offloading systems to better support user expectations.

Acknowledgment

This research was supported by MSIP (Ministry of Science, ICT and Future Planning), Korea, under the ICT Consilience Creative Program (IITP-2015-R0346-15-1007) supervised by IITP (Institute for Information & Communications Technology Promotion).

References

[1] Abebe, E., & Ryan, C. (2012). Adaptive application offloading using distributed abstract class graphs in mobile environments. *Journal of Systems and Software*, *85*, 2755–2769. doi:10.1016/j.jss.2012.05.091.

[2] Abolfazli, S., Sanaei, Z., Ahmed, E., Gani, A., & Buyya, R. (2014). Cloud-based augmentation for mobile devices: motivation, taxonomies, and open challenges. *IEEE Communications Surveys & Tutorials*, *16*, 337–368. doi:10.1109/SURV.2013.070813.00285.

[3] Abolfazli, S., Sanaei, Z., Alizadeh, M., Gani, A., & Xia, F. (2014). An experimental analysis on cloud-based mobile augmentation in mobile cloud computing. *IEEE Transactions on Consumer Electronics*, *60*, 146–154. doi:10.1109/tce.2014.6780937.

[4] Abolfazli, S., Sanaei, Z., Gani, A., Xia, F., & Lin, W. M. (2014). Rmcc: Restful mobile cloud computing framework for exploiting adjacent service-based mobile cloudlets. In *2014 IEEE 6th International Conference on Cloud Computing Technology and Science (CloudCom)* (pp. 793–798). doi:10.1109/CloudCom.2014.91.

[5] Abolfazli, S., Sanaei, Z., Shiraz, M., & Gani, A. (2012). Momcc: Market-oriented architecture for mobile cloud computing based on service oriented architecture. In *2012 1st IEEE International Conference on Communications in China Workshops (ICCC)* (pp. 8–13). doi:10.1109/ICCCW.2012.6316481.

[6] Ahmed, E., Gani, A., Khan, M. K., Buyya, R., & Khan, S. U. (2015). Seamless application execution in mobile cloud computing: Motivation, taxonomy, and open challenges. *Journal of Network and Computer Applications*, *52*, 154 – 172. doi:10.1016/j.jnca.2015.03.001.

[7] Ahmed, E., Gani, A., Sookhak, M., Hamid, S. H. A., & Xia, F. (2015). Application optimization in mobile cloud computing: Motivation, taxonomies, and open challenges. *Journal of Network and Computer Applications*, *52*, 52 – 68. doi:10.1016/j.jnca.2015.02.003.

[8] Alizadeh, M., Abolfazli, S., Zamani, M., Baharun, S., & Sakurai, K. (2016). Authentication in mobile cloud computing: A survey. *Journal of Network and Computer Applications*, *61*, 59 – 80. doi:10.1016/j.jnca.2015.10.005.

[9] Andrews, J., Buzzi, S., Choi, W., Hanly, S., Lozano, A., Soong, A., & Zhang, J. (2014). What will 5g be? *IEEE Journal on Selected Areas in Communications*, *32*, 1065–1082. doi:10.1109/JSAC.2014.2328098.

[10] Ardagna, D., Casale, G., Ciavotta, M., Prez, J., & Wang, W. (2014). Quality-of-service in cloud computing: modeling techniques and their applications. *Journal of Internet Services and Applications*, *5*. doi:10.1186/s13174-014-0011-3.

[11] Balakrishnan, P., & Tham, C.-K. (2013). Energy-efficient mapping and scheduling of task interaction graphs for code offloading in mobile cloud computing. In *Proceedings of the 2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing* UCC '13 (pp. 34–41). IEEE Computer Society. doi:10.1109/UCC.2013.23.

[12] Balasubramanian, N., Balasubramanian, A., & Venkataramani, A. (2009). 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* IMC '09 (pp. 280–293). ACM. doi:10.1145/1644893.1644927.

[13] Baldauf, M., Dustdar, S., & Rosenberg, F. (2007). A survey on context-aware systems. *International Journal of Ad Hoc and Ubiquitous Computing*, *2*, 263–277. doi:http://dx.doi.org/10.1504/IJAHUC.2007.014070.

[14] Barbera, M., Kosta, S., Mei, A., Perta, V., & Stefa, J. (2014). Mobile offloading in the wild: Findings and lessons learned through a real-life experiment with a new cloud-aware system. In *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications* (pp. 2355–2363). doi:10.1109/INFOCOM.2014.6848180.

[15] Bhardwaj, K., Sreepathy, S., Gavrilovska, A., & Schwan, K. (2014). Ecc: Edge cloud composites. In *2014 2nd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)* (pp. 38–47). doi:10.1109/MobileCloud.2014.18.

[16] Bonomi, F., Milito, R., Zhu, J., & Addepalli, S. (2012). Fog computing and its role in the internet of things. In *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing* MCC '12 (pp. 13–16). ACM. doi:10.1145/2342509.2342513.

[17] Chakraborty, A., Navda, V., Padmanabhan, V. N., & Ramjee, R. (2013). Coordinating cellular background transfers using loadsense. In *Proceedings of the 19th Annual International Conference on Mobile Computing and Networking* MobiCom '13 (pp. 63–74). ACM. doi:10.1145/2500423.2500447.

[18] Chalmers, D., & Sloman, M. (1999). A survey of quality of service in mobile computing environments. *IEEE Communications Surveys & Tutorials*, *2*, 2–10. doi:10.1109/COMST.1999.5340514.

[19] Chang, Y.-S., Hung, S.-H., Wang, N., & Lin, B.-S. (2011). Csr: A cloud-assisted speech recognition service for personal mobile device. In *2011 International Conference on Parallel Processing (ICPP)* (pp. 305–314). doi:10.1109/ICPP.2011.23.

[20] Chen, S., Li, M., Ren, K., Fu, X., & Qiao, C. (2015). Rise of the indoor crowd: Reconstruction of building interior view via mobile crowdsourcing. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems* SenSys '15 (pp. 59–71). ACM. doi:10.1145/2809695.2809702.

[21] Chen, S., Wang, Y., & Pedram, M. (2013). A semi-markovian decision process based control method for offloading tasks from mobile devices to the cloud. In *2013 IEEE Global Communications Conference (GLOBECOM)* (pp. 2885–2890). doi:10.1109/GLOCOM.2013.6831512.

[22] Chen, S., Wang, Y., & Pedram, M. (2014). Optimal offloading control for a mobile device based on a realistic battery model and semi-markov decision process. In *2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)* (pp. 369–375). doi:10.1109/ICCAD.2014.7001378.

[23] Chen, T. Y.-H., Ravindranath, L., Deng, S., Bahl, P., & Balakrishnan, H. (2015). Glimpse: Continuous, real-time object recognition on mobile devices. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems* SenSys '15 (pp. 155–168). ACM. doi:10.1145/2809695.2809711.

[24] Chen, X. (2015). Decentralized computation offloading game for mobile cloud computing. *IEEE Transactions on Parallel and Distributed Systems*, *26*, 974–983. doi:10.1109/TPDS.2014.2316834.

[25] Cheng, B. H. C., Sawyer, P., Bencomo, N., & Whittle, J. (2009). Model driven engineering languages and systems: 12th international conference, models 2009, denver, co, usa, october 4-9, 2009. proceedings. chapter A Goal-Based Modeling Approach to Develop Requirements of an Adaptive System with Environmental Uncertainty. (pp. 468–483). Berlin, Heidelberg: Springer Berlin Heidelberg. doi:10.1007/978-3-642-04425-0_36.

[26] Chilipirea, C., Petre, A.-C., Dobre, C., & Pop, F. (2015). Enabling mobile cloud wide spread through an evolutionary market-based approach. *IEEE Systems Journal*, *PP*, 1–8. doi:10.1109/JSYST.2015.2415211.

[27] Cho, H.-D., Engineer, P. D. P., Chung, K., & Kim, T. (2012). Benefits of the big. little architecture. *EETimes, Feb*, .

[28] Chowdhury, N. M. K., & Boutaba, R. (2010). A survey of network virtualization. *Computer Networks*, *54*, 862 – 876. doi:http://dx.doi.org/10.1016/j.comnet.2009.10.017.

[29] Chun, B.-G., Ihm, S., Maniatis, P., Naik, M., & Patti, A. (2011). Clonecloud: Elastic execution between mobile device and cloud. In *Proceedings of the Sixth Conference on Computer Systems* EuroSys '11 (pp. 301–314). ACM. doi:10.1145/1966445.1966473.

[30] Chun, B.-G., & Maniatis, P. (2010). Dynamically partitioning applications between weak devices and clouds. In *Proceedings of the 1st ACM Workshop on Mobile Cloud Computing and Services: Social Networks and Beyond* MCS '10 (pp. 7:1–7:5). ACM. doi:10.1145/1810931.1810938.

[31] Cuervo, E., Balasubramanian, A., Cho, D.-k., Wolman, A., Saroiu, S., Chandra, R., & Bahl, P. (2010). Maui: Making smartphones last longer with code offload. In *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services* MobiSys '10 (pp. 49–62). ACM. doi:10.1145/1814433.1814441.

[32] Cui, Y., Xiao, S., Wang, X., Li, M., Wang, H., & Lai, Z. (2014). Performance-aware energy optimization on mobile devices in cellular network. In *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications* (pp. 1123–1131). doi:10.1109/INFOCOM.2014.6848043.

[33] Deng, S., Huang, L., Taheri, J., & Zomaya, A. (2015). Computation offloading for service workflow in mobile cloud computing. *IEEE Transactions on Parallel and Distributed Systems*, *26*, 3317–3329. doi:10.1109/TPDS.2014.2381640.

[34] Dinh, H. T., Lee, C., Niyato, D., & Wang, P. (2013). A survey of mobile cloud computing: architecture, applications, and approaches. *Wireless communications and mobile computing*, *13*, 1587–1611. doi:10.1002/wcm.1203.

[35] Dong, M., & Zhong, L. (2011). Self-constructive high-rate system energy modeling for battery-powered mobile systems. In *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services* MobiSys '11 (pp. 335–348). ACM. doi:10.1145/1999995.2000027.

[36] Dou, A., Kalogeraki, V., Gunopulos, D., Mielikainen, T., & Tuulos, V. H. (2010). Misco: A mapreduce framework for mobile systems. In *Proceedings of the 3rd International Conference on PErvasive Technologies Related to Assistive Environments* PETRA '10 (pp. 32:1–32:8). ACM. doi:10.1145/1839294.1839332.

[37] Drolia, U., Martins, R., Tan, J., Chheda, A., Sanghavi, M., Gandhi, R., & Narasimhan, P. (2013). The case for mobile edge-clouds. In *2013 IEEE 10th International Conference on Autonomic and Trusted Computing (UIC/ATC)* (pp. 209–215). doi:10.1109/UIC-ATC.2013.94.

[38] Einsiedler, H., Bayer, N., Haensge, K., Szczepanski, R., Kurze, M., Rettig, T., Gonzalez-Garcia, F., Roos, A., Berg, S., & Moreno, J. (2014). Efficient transmission of smartphone application traffic in wireless access networks. In *2014 2nd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)* (pp. 186–193). doi:10.1109/MobileCloud.2014.35.

[39] Eom, H., Figueiredo, R., Cai, H., Zhang, Y., & Huang, G. (2015). Malmos: Machine learning-based mobile offloading scheduler with online training. In *2015 3rd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)* (pp. 51–60). doi:10.1109/MobileCloud.2015.19.

[40] Eom, H., St.Juste, P., Figueiredo, R., Tickoo, O., Illikkal, R., & Iyer, R. (2013). Opencl-based remote offloading framework for trusted mobile cloud computing. In *2013 International Conference on Parallel and Distributed Systems (ICPADS)* (pp. 240–248). doi:10.1109/ICPADS.2013.43.

[41] Felt, A. P., Egelman, S., & Wagner, D. (2012). I've got 99 problems, but vibration ain't one: A survey of smartphone users' concerns. In *Proceedings of the Second ACM Workshop on Security and Privacy in Smartphones and Mobile Devices* SPSM '12 (pp. 33–44). ACM. doi:10.1145/2381934.2381943.

[42] Fernando, N., Loke, S. W., & Rahayu, W. (2013). Mobile cloud computing: A survey. *Future Generation Computer Systems*, *29*, 84–106. doi:http://dx.doi.org/10.1016/j.future.2012.05.023.

[43] Fesehaye, D., Gao, Y., Nahrstedt, K., & Wang, G. (2012). Impact of cloudlets on interactive mobile cloud applications. In *2012 IEEE 16th International Enterprise Distributed Object Computing Conference (EDOC)* (pp. 123–132). IEEE. doi:10.1109/EDOC.2012.23.

[44] Flores, H., Hui, P., Tarkoma, S., Li, Y., Srirama, S., & Buyya, R. (2015). Mobile code offloading: from concept to practice and beyond. *IEEE Communications Magazine*, *53*, 80–88. doi:10.1109/MCOM.2015.7060486.

[45] Gao, W., Li, Y., Lu, H., Wang, T., & Liu, C. (2014). On exploiting dynamic execution patterns for workload offloading in mobile cloud applications. In *2014 IEEE 22nd International Conference on Network Protocols (ICNP)* (pp. 1–12). doi:10.1109/ICNP.2014.22.

[46] Garcia Lopez, P., Montresor, A., Epema, D., Datta, A., Higashino, T., Iamnitchi, A., Barcellos, M., Felber, P., & Riviere, E. (2015). Edge-centric computing: Vision and challenges. *SIGCOMM Comput. Commun. Rev.*, *45*, 37–42. doi:10.1145/2831347.2831354.

[47] Gember, A., Dragga, C., & Akella, A. (2012). Ecos: Leveraging software-defined networks to support mobile application offloading. In *Proceedings of the Eighth ACM/IEEE Symposium on Architectures for Networking and Communications Systems* ANCS '12 (pp. 199–210). ACM. doi:10.1145/2396556.2396598.

[48] Gordon, M. S., Hong, D. K., Chen, P. M., Flinn, J., Mahlke, S., & Mao, Z. M. (2015). Accelerating mobile applications through flip-flop replication. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services* MobiSys '15 (pp. 137–150). ACM. doi:10.1145/2742647.2742649.

[49] Hoang, D. T., Niyato, D., & Wang, P. (2012). Optimal admission control policy for mobile cloud computing hotspot with cloudlet. In *2012 IEEE Wireless Communications and Networking Conference (WCNC)* (pp. 3145–3149). doi:10.1109/WCNC.2012.6214347.

[50] Hong, K., Lillethun, D., Ramachandran, U., Ottenwälder, B., & Koldehofe, B. (2013). Mobile fog: A programming model for large-scale applications on the internet of things. In *Proceedings of the Second ACM SIGCOMM Workshop on Mobile Cloud Computing* MCC '13 (pp. 15–20). ACM. doi:10.1145/2491266.2491270.

[51] Hu, W., Amos, B., Chen, Z., Ha, K., Richter, W., Pillai, P., Gilbert, B., Harkes, J., & Satyanarayanan, M. (2015). The case for offload shaping. In *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications* HotMobile '15 (pp. 51–56). ACM. doi:10.1145/2699343.2699351.

[52] Hu, W., & Cao, G. (2014). Quality-aware traffic offloading in wireless networks. In *Proceedings of the 15th ACM International Symposium on Mobile Ad Hoc Networking and Computing* MobiHoc '14 (pp. 277–286). ACM. doi:10.1145/2632951.2632954.

[53] Huerta-Canepa, G., & Lee, D. (2010). A virtual cloud computing provider for mobile devices. In *Proceedings of the 1st ACM Workshop on Mobile Cloud Computing and Services: Social Networks and Beyond* MCS '10 (pp. 6:1–6:5). ACM. doi:10.1145/1810931.1810937.

[54] Jararweh, Y., Doulat, A., AlQudah, O., Ahmed, E., Al-Ayyoub, M., & Benkhelifa, E. (2016). The future of mobile cloud computing: Integrating cloudlets and mobile edge computing. In *Proceedings of the 23rd International Conference on Telecommunications (ICT)*.

[55] Johnson, D., Stack, T., Fish, R., Flickinger, D., Stoller, L., Ricci, R., & Lepreau, J. (2006). Mobile emulab: A robotic wireless and sensor network testbed. In *IEEE INFOCOM 2006 - IEEE Conference on Computer Communications* (pp. 1–12). doi:10.1109/INFOCOM.2006.182.

[56] Kao, Y.-H., Krishnamachari, B., Ra, M.-R., & Bai, F. (2015). Hermes: Latency optimal task assignment for resource-constrained mobile computing. In *IEEE INFOCOM 2015 - IEEE Conference on Computer Communications* (pp. 1894–1902). doi:10.1109/INFOCOM.2015.7218572.

[57] Khalifa, A., & Eltoweissy, M. (2013). Collaborative autonomic resource management system for mobile cloud computing. In *2013 Fourth International Conference on Cloud Computing, GRIDs, and Virtualization* (pp. 115–121).

[58] Khan, A., Othman, M., Madani, S., & Khan, S. (2014). A survey of mobile cloud computing application models. *IEEE Communications Surveys & Tutorials*, *16*, 393–413. doi:10.1109/SURV.2013.062613.00160.

[59] Khan, M. A. (2015). A survey of computation offloading strategies for performance improvement of applications running on mobile devices. *Journal of Network and Computer Applications*, *56*, 28 – 40. doi:10.1016/j.jnca.2015.05.018.

[60] Kosta, S., Aucinas, A., Hui, P., Mortier, R., & Zhang, X. (2012). Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading. In *IEEE INFOCOM 2012 - IEEE Conference on Computer Communications* (pp. 945–953). doi:10.1109/INFCOM.2012.6195845.

[61] Kumar, K., Liu, J., Lu, Y.-H., & Bhargava, B. (2013). A survey of computation offloading for mobile systems. *Mobile Networks and Applications*, *18*, 129–140. doi:10.1007/s11036-012-0368-0.

[62] Kwak, J., Kim, Y., Lee, J., & Chong, S. (2015). Dream: Dynamic resource and task allocation for energy minimization in mobile cloud systems. *IEEE Journal on Selected Areas in Communications*, *33*, 2510–2523. doi:10.1109/JSAC.2015.2478718.

[63] Kwon, Y.-W., & Tilevich, E. (2012). Energy-efficient and fault-tolerant distributed mobile execution. In *2012 IEEE 32nd International Conference on Distributed Computing Systems (ICDCS)* (pp. 586–595). doi:10.1109/ICDCS.2012.75.

[64] Lee, G., Park, H., Heo, S., Chang, K.-A., Lee, H., & Kim, H. (2015). Architecture-aware automatic computation offload for native applications. In *Proceedings of the 48th International Symposium on Microarchitecture* MICRO-48 (pp. 521–532). ACM. doi:10.1145/2830772.2830833.

[65] Lee, J., Uddin, M., Tourrilhes, J., Sen, S., Banerjee, S., Arndt, M., Kim, K.-H., & Nadeem, T. (2014). mesdn: Mobile extension of sdn. In *Proceedings of the Fifth International Workshop on Mobile Cloud Computing and Services* MCS '14 (pp. 7–14). ACM. doi:10.1145/2609908.2609948.

[66] Li, J., Bu, K., Liu, X., & Xiao, B. (2013). Enda: Embracing network inconsistency for dynamic application offloading in mobile cloud computing. In *Proceedings of the Second ACM SIGCOMM Workshop on Mobile Cloud Computing* MCC '13 (pp. 39–44). ACM. doi:10.1145/2491266.2491274.

[67] Li, J., Peng, Z., Xiao, B., & Hua, Y. (2015). Make smartphones last a day: Pre-processing based computer vision application offloading. In *2015 12th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)* (pp. 462–470). doi:10.1109/SAHCN.2015.7338347.

[68] Li, W., Zhao, Y., Lu, S., & Chen, D. (2015). Mechanisms and challenges on mobility-augmented service provisioning for mobile cloud computing. *IEEE Communications Magazine*, *53*, 89–97. doi:10.1109/MCOM.2015.7060487.

[69] Li, Y., & Gao, W. (2015). Code offload with least context migration in the mobile cloud. In *IEEE INFOCOM 2015 - IEEE Conference on Computer Communications* (pp. 1876–1884). doi:10.1109/INFOCOM.2015.7218570.

[70] Liang, H., Huang, D., Cai, L., Shen, X., & Peng, D. (2011). Resource allocation for security services in mobile cloud computing. In *2011 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)* (pp. 191–195). doi:10.1109/INFCOMW.2011.5928806.

[71] Lin, C.-K., & Kung, H. T. (2014). Mobile app acceleration via fine-grain offloading to the cloud. In *Proceedings of the 6th USENIX Conference on Hot Topics in Cloud Computing* HotCloud'14 (pp. 8–8). USENIX Association.

[72] Lin, X., Wang, Y., & Pedram, M. (2013). An optimal control policy in a mobile cloud computing system based on stochastic data. In *2013 IEEE 2nd International Conference on Cloud Networking (CloudNet)* (pp. 117–122). doi:10.1109/CloudNet.2013.6710565.

[73] Lin, X., Wang, Y., Xie, Q., & Pedram, M. (2014). Energy and performance-aware task scheduling in a mobile cloud computing environment. In *Proceedings of the 2014 IEEE International Conference on Cloud Computing* CLOUD '14 (pp. 192–199). IEEE Computer Society. doi:10.1109/CLOUD.2014.35.

[74] Lin, Y.-D., Chu, E.-H., Lai, Y.-C., & Huang, T.-J. (2013). Time-and-energy-aware computation offloading in handheld devices to coprocessors and clouds. *IEEE Systems Journal*, *PP*, 1–13. doi:10.1109/JSYST.2013.2289556.

[75] Liu, J., Ahmed, E., Shiraz, M., Gani, A., Buyya, R., & Qureshi, A. (2015). Application partitioning algorithms in mobile cloud computing: Taxonomy, review and future directions. *Journal of Network and Computer Applications*, *48*, 99–117. doi:10.1016/j.jnca.2014.09.009.

[76] Mangold, S., Choi, S., May, P., Klein, O., Hiertz, G., & Stibor, L. (2002). Ieee 802.11 e wireless lan for quality of service. In *Proc. European Wireless* (pp. 32–39). volume 2.

[77] Marinelli, E. E. (2009). *Hyrax: cloud computing on mobile devices using MapReduce*. Technical Report DTIC Document.

[78] Mehmeti, F., & Spyropoulos, T. (2014). Is it worth to be patient? analysis and optimization of delayed mobile data offloading. In *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications* (pp. 2364–2372). doi:10.1109/INFOCOM.2014.6848181.

[79] Namboodiri, V., & Ghose, T. (2012). To cloud or not to cloud: A mobile device perspective on energy consumption of applications. In *2012 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)* (pp. 1–9). doi:10.1109/WoWMoM.2012.6263712.

[80] Niu, J., Song, W., & Atiquzzaman, M. (2014). Bandwidth-adaptive partitioning for distributed execution optimization of mobile applications. *Journal of Network and Computer Applications*, *37*, 334 – 347. doi:10.1016/j.jnca.2013.03.007.

[81] OSullivan, M. J., & Grigoras, D. (2015). Integrating mobile and cloud resources management using the cloud personal assistant. *Simulation Modelling Practice and Theory*, *50*, 20 – 41. doi:10.1016/j.simpat.2014.06.017. Special Issue on Resource Management in Mobile Clouds.

[82] Panasonic P81 (2014). Panasonic smart phone. http://www.mobile.panasonic.co.in/productP81.html. Online; accessed 26 January 2016.

[83] Park, J., Yu, H., Chung, K., & Lee, E. (2011). Markov chain based monitoring service for fault tolerance in mobile cloud computing. In *2011 IEEE Workshops of International Conference on Advanced Information Networking and Applications (WAINA)* (pp. 520–525). doi:10.1109/WAINA.2011.10.

[84] Ra, M.-R., Sheth, A., Mummert, L., Pillai, P., Wetherall, D., & Govindan, R. (2011). Odessa: Enabling interactive perception applications on mobile devices. In *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services* MobiSys '11 (pp. 43–56). ACM. doi:10.1145/1999995.2000000.

[85] Rahimi, M. R., Venkatasubramanian, N., Mehrotra, S., & Vasilakos, A. V. (2012). 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* UCC '12 (pp. 83–90). IEEE Computer Society. doi:10.1109/UCC.2012.25.

[86] Rahimi, M. R., Venkatasubramanian, N., & Vasilakos, A. (2013). Music: Mobility-aware optimal service allocation in mobile cloud computing. In *2013 IEEE Sixth International Conference on Cloud Computing (CLOUD)* (pp. 75–82). doi:10.1109/CLOUD.2013.100.

[87] Ravindranath, L., Padhye, J., Mahajan, R., & Balakrishnan, H. (2013). Timecard: Controlling user-perceived delays in server-based mobile applications. In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles* SOSP '13 (pp. 85–100). ACM. doi:10.1145/2517349.2522717.

[88] Sanaei, Z., Abolfazli, S., Gani, A., & Buyya, R. (2014). Heterogeneity in mobile cloud computing: taxonomy and open challenges. *IEEE Communications Surveys & Tutorials*, *16*, 369–392. doi:10.1109/SURV.2013.050113.00090.

[89] Satyanarayanan, M. (1996). Fundamental challenges in mobile computing. In *Proceedings of the Fifteenth Annual ACM Symposium on Principles of Distributed Computing* PODC '96 (pp. 1–7). ACM. doi:10.1145/248052.248053.

[90] Satyanarayanan, M. (2015). A brief history of cloud offload: A personal journey from odyssey through cyber foraging to cloudlets. *GetMobile: Mobile Comp. and Comm.*, *18*, 19–23. doi:10.1145/2721914.2721921.

[91] Satyanarayanan, M., Bahl, P., Caceres, R., & Davies, N. (2009). The case for vm-based cloudlets in mobile computing. *IEEE Pervasive Computing*, *8*, 14–23. doi:10.1109/MPRV.2009.82.

[92] Satyanarayanan, M., Lewis, G. A., Morris, E. J., Simanta, S., Boleng, J., & Ha, K. (2013). The role of cloudlets in hostile environments. *IEEE Pervasive Computing*, *12*, 40–49. doi:10.1109/MPRV.2013.77.

[93] Schaffrath, G., Schmid, S., & Feldmann, A. (2012). Optimizing long-lived cloudnets with migrations. In *2012 IEEE Fifth International Conference on Utility and Cloud Computing (UCC)* (pp. 99–106). doi:10.1109/UCC.2012.7.

[94] Schulman, A., Navda, V., Ramjee, R., Spring, N., Deshpande, P., Grunewald, C., Jain, K., & Padmanabhan, V. N. (2010). Bartendr: A practical approach to energy-aware cellular data scheduling. In *Proceedings of the Sixteenth Annual International Conference on Mobile Computing and Networking* MobiCom '10 (pp. 85–96). ACM. doi:10.1145/1859995.1860006.

[95] Sharifi, M., Kafaie, S., & Kashefi, O. (2012). A survey and taxonomy of cyber foraging of mobile devices. *IEEE Communications Surveys & Tutorials*, *14*, 1232–1243. doi:10.1109/SURV.2011.111411.00016.

[96] Shaukat, U., Ahmed, E., Anwar, Z., & Xia, F. (2016). Cloudlet deployment in local wireless networks: Motivation, architectures, applications, and open challenges. *Journal of Network and Computer Applications*, *62*, 18 – 40. doi:http://dx.doi.org/10.1016/j.jnca.2015.11.009.

[97] Shi, C., Habak, K., Pandurangan, P., Ammar, M., Naik, M., & Zegura, E. (2014). Cosmos: Computation offloading as a service for mobile devices. In *Proceedings of the 15th ACM International Symposium on Mobile Ad Hoc Networking and Computing* MobiHoc '14 (pp. 287–296). ACM. doi:10.1145/2632951.2632958.

[98] Shi, C., Lakafosis, V., Ammar, M. H., & Zegura, E. W. (2012). Serendipity: Enabling remote computing among intermittently connected mobile devices. In *Proceedings of the Thirteenth ACM International Symposium on Mobile Ad Hoc Networking and Computing* MobiHoc '12 (pp. 145–154). ACM. doi:10.1145/2248371.2248394.

[99] Shih, C.-S., Chen, J., Wang, Y.-H., & Chang, N. (2014). Imprecise computation over the cloud. In *2014 2nd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)* (pp. 29–37). doi:10.1109/MobileCloud.2014.17.

[100] Shiraz, M., Gani, A., Khokhar, R., & Buyya, R. (2013). A review on distributed application processing frameworks in smart mobile devices for mobile cloud computing. *IEEE Communications Surveys & Tutorials*, *15*, 1294–1313. doi:10.1109/SURV.2012.111412.00045.

[101] Shiraz, M., Sookhak, M., Gani, A., & Shah, S. A. A. (2015). A study on the critical analysis of computational offloading frameworks for mobile cloud computing. *Journal of Network and Computer Applications*, *47*, 47 – 60. doi:10.1016/j.jnca.2014.08.011.

[102] Silva, F. A., Zaicaner, G., Quesado, E., Dornelas, M., Silva, B., & Maciel, P. (2016). Benchmark applications used in mobile cloud computing research: a systematic mapping study. *The Journal of Supercomputing*, *72*, 1431–1452. doi:10.1007/s11227-016-1674-2.

[103] Sood, S. K., & Sandhu, R. (2015). Matrix based proactive resource provisioning in mobile cloud environment. *Simulation Modelling Practice and Theory*, *50*, 83 – 95. doi:10.1016/j.simpat.2014.06.004. Special Issue on Resource Management in Mobile Clouds.

[104] Tong, L., & Gao, W. (2016). Application-aware traffic scheduling for workload offloading in mobile clouds. In *IEEE INFOCOM 2016 - IEEE Conference on Computer Communications*.

[105] Čapkun, S., Hubaux, J.-P., & Buttyán, L. (2003). Mobility helps security in ad hoc networks. In *Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking* MobiHoc '03 (pp. 46–56). ACM. doi:10.1145/778415.778422.

[106] Verbelen, T., Stevens, T., De Turck, F., & Dhoedt, B. (2013). Graph partitioning algorithms for optimizing software deployment in mobile cloud computing. *Future Generation Computer Systems*, *29*, 451–459. doi:http://dx.doi.org/10.1016/j.future.2012.07.003.

[107] Wang, H., Shea, R., Ma, X., Wang, F., & Liu, J. (2014). On design and performance of cloud-based distributed interactive applications. In *2014 IEEE 22nd International Conference on Network Protocols (ICNP)* (pp. 37–46). doi:10.1109/ICNP.2014.25.

[108] Wang, S., & Dey, S. (2010). Rendering adaptation to raddress communication and computation constraints in cloud mobile gaming. In *2010 IEEE Global Telecommunications Conference (GLOBECOM 2010)* (pp. 1–6). doi:10.1109/GLOCOM.2010.5684144.

[109] Wu, H., & Huang, D. (2014). Modeling multi-factor multi-site risk-based offloading for mobile cloud computing. In *2014 10th International Conference on Network and Service Management (CNSM)* (pp. 230–235). doi:10.1109/CNSM.2014.7014164.

[110] Xiang, L., Ye, S., Feng, Y., Li, B., & Li, B. (2014). Ready, set, go: Coalesced offloading from mobile devices to the cloud. In *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications* (pp. 2373–2381). doi:10.1109/INFOCOM.2014.6848182.

[111] Xiong, J., & Jamieson, K. (2013). Securearray: Improving wifi security with fine-grained physical-layer information. In *Proceedings of the 19th Annual International Conference on Mobile Computing and Networking* MobiCom '13 (pp. 441–452). ACM. doi:10.1145/2500423.2500444.

[112] Xu, Q., Mehrotra, S., Mao, Z., & Li, J. (2013). Proteus: Network performance forecast for real-time, interactive mobile applications. In *Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services* MobiSys '13 (pp. 347–360). ACM. doi:10.1145/2462456.2464453.

[113] Yang, L., Cao, J., Tang, S., Han, D., & Suri, N. (2014). Run time application repartitioning in dynamic mobile cloud environments. *IEEE Transactions on Cloud Computing*, *PP*, 1–1. doi:10.1109/TCC.2014.2358239.

[114] Yang, L., Cao, J., Yuan, Y., Li, T., Han, A., & Chan, A. (2013). A framework for partitioning and execution of data stream applications in mobile cloud computing. *SIGMETRICS Perform. Eval. Rev.*, *40*, 23–32. doi:10.1145/2479942.2479946.

[115] Yin, Z., Yu, F., Bu, S., & Han, Z. (2015). Joint cloud and wireless networks operations in mobile cloud computing environments with telecom operator cloud. *IEEE Transactions on Wireless Communications*, *14*, 4020–4033. doi:10.1109/TWC.2015.2416177.

[116] Zhang, L., Tiwana, B., Qian, Z., Wang, Z., Dick, R. P., Mao, Z. M., & Yang, L. (2010). Accurate online power estimation and automatic battery behavior based power model generation for smartphones. In *Proceedings of the Eighth IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis* CODES/ISSS '10 (pp. 105–114). ACM. doi:10.1145/1878961.1878982.

[117] Zhang, W., Wen, Y., & Chen, H.-H. (2014). Toward transcoding as a service: energy-efficient offloading policy for green mobile cloud. *IEEE Network*, *28*, 67–73. doi:10.1109/MNET.2014.6963807.

[118] Zhang, W., Wen, Y., & Wu, D. O. (2015). Collaborative task execution in mobile cloud computing under a stochastic wireless channel. *IEEE Transactions on Wireless Communications*, *14*, 81–93. doi:10.1109/TWC.2014.2331051.

[119] Zhang, W., Wen, Y., & Zhang, X. (2015). Towards virus scanning as a service in mobile cloud computing: Energy-efficient dispatching policy under n-version protection. *IEEE Transactions on Emerging Topics in Computing*, *PP*, 1–1. doi:10.1109/TETC.2015.2471852.

[120] Zhang, Y., Yang, R., Wo, T., Hu, C., Kang, J., & Cui, L. (2013). Cloudap: Improving the qos of mobile applications with efficient vm migration. In *2013 IEEE 10th International Conference on High Performance Computing and Communications, 2013 IEEE International Conference on Embedded and Ubiquitous Computing (HPCC-EUC)* (pp. 1374–1381). doi:10.1109/HPCC.and.EUC.2013.195.