Augmentation Techniques for Mobile Cloud Computing: A Taxonomy, Survey, and Future Directions

BOWEN ZHOU and RAJKUMAR BUYYA, The University of Melbourne, Australia

Despite the rapid growth of hardware capacity and popularity in mobile devices, limited resources in battery and processing capacity still lack the ability to meet increasing mobile users' demands. Both conventional techniques and emerging approaches are brought together to fill this gap between user demand and mobile devices' limited capabilities. Recent research has focused on enhancing the performance of mobile devices via augmentation techniques. Augmentation techniques for mobile cloud computing refer to the computing paradigms and solutions to outsource mobile device computation and storage to more powerful computing resources in order to enhance a mobile device's computing capability and energy efficiency (e.g., code offloading). Adopting augmentation techniques in the heterogeneous and intermittent mobile cloud computing environment creates new challenges for computation management, energy efficiency, and system reliability. In this article, we aim to provide a comprehensive taxonomy and survey of the existing techniques and frameworks for mobile cloud augmentation regarding both computation and storage. Different from the existing taxonomies in this field, we focus on the techniques aspect, following the idea of realizing a complete mobile cloud computing system. The objective of this survey is to provide a guide on what available augmentation techniques can be adopted in mobile cloud computing systems as well as supporting mechanisms such as decision-making and fault tolerance policies for realizing reliable mobile cloud services. We also present a discussion on the open challenges and future research directions in this field.

CCS Concepts: • **General and reference** → Surveys and overviews; • **Computer systems organization** → Cloud computing; • **Software and its engineering** → Cloud computing; • **Human-centered computing** → Mobile computing;

Additional Key Words and Phrases: Mobile cloud computing, mobile device augmentation technique

ACM Reference format:
Bowen Zhou and Rajkumar Buyya. 2018. Augmentation Techniques for Mobile Cloud Computing: A Taxonomy, Survey, and Future Directions. *ACM Comput. Surv.* 51, 1, Article 13 (January 2018), 38 pages.
https://doi.org/10.1145/3152397

1 INTRODUCTION

Recent years have seen the exponential development of smart mobile devices, which have gained enormous popularity among mobile device users through more advanced mobile applications to enrich the user experience. However, compared to the growing demand for more enhanced user experience mobile applications from users, mobile devices still lack adequate resources such as processing capability, storage, and battery lifetime to provide such applications. The gap between

Authors' addresses: B. Zhou and R. Buyya, The Cloud Computing and Distributed Systems Laboratory, School of Computing and Information Systems, The University of Melbourne, Australia; emails: bowenz@student.unimelb.edu.au, rbuyya@unimelb.edu.au.

https://doi.org/10.1145/3152397

mobile hardware and users' demand will hinder the mobile devices from providing experience-rich mobile services.

Due to the slow development of battery technology, mobile resource augmentation has been considered as a critical approach to tackling the gap. Augmentation techniques for resource-constrained machines have been investigated for two decades. In the early 1990s, the ideas of remote execution and interprocess communication were introduced to help leverage the computing resources of computer clusters and manage message-passing traffic (Gropp et al. 1994; Stevens 1990; Nelson 1981). Remote execution aims to augment the computing capacity of resource-limited computers, such as mobile devices and computers, to run computation-intensive applications in a *client-server* mode. However, utilizing distributed resources can bring many challenges, including parameter marshaling, client and service binding, and the semantics of remote invocation.

Later, with the development of the Internet and remote execution, a new service paradigm called Service-Oriented Architecture (SOA) was introduced. Mobile Web Services (MWS) (Pashtan 2005; Srirama et al. 2006) incorporate SOA with mobile devices to enable mobile device users to share existing software and services on other devices to augment capability and conserve energy. Mobile devices in MWS can be either service clients or service providers.

Since remote execution and SOA rely on stable network connections, the performance of these systems can be unstable. Researchers have studied the possibility of utilizing proximal computing resources (e.g., mobile devices nearby) through short-range wireless communications to provide an ad-hoc computing augmentation service. The earliest idea of the wireless ad-hoc network, the "packet radio" network, was introduced by Defence Advanced Research Projects Agency (DARPA) in the early 1970s (Jubin and Tornow 1987). Due to the development of the mobile device and 802.11/WiFi, the Mobile Ad-Hoc Wireless Network (MANET) was then developed in the mid-1990s to provide a self-configuring mobile device network without the requirement of network infrastructures for collaborative mobile computing. The disadvantage of MANET is that the dynamic nature of network topology caused by the mobility of devices requires high adaptability of network functions. Moreover, this type of mobile device augmentation can only provide limited resources within the resource pool built with other mobile devices. Later, *pervasive computing* (Satyanarayanan 2001) was introduced for a ubiquitous computing paradigm that enables computing to migrate from any device to any other devices via any type of network as the user moves. This paradigm essentially makes it possible to merge the computing resources of desktops, servers, and mobile devices to provide continuing services.

Recently, *cloud computing* emerged as a promising computing paradigm in both industry and academia. Cloud computing aims to leverage virtualization technologies to provide computing resources such as computation, storage, and networks as a utility. The advantages, such as on-demand self-service, broad accessibility, elastic scalability, and pay-as-you-go services, make cloud computing a potential computing resource for mobile device augmentation. The computing paradigm that leverages cloud computing resources to enhance the performance of resource-constrained mobile devices is called *mobile cloud computing*. In some works, it is also described as *cyber-foraging* (Lewis and Lago 2015). We use the term "mobile cloud computing" throughout this article. A significant amount of research has been proposed in this area regarding the approaches for mobile cloud augmentation, resource provisioning, and management (Cuervo et al. 2010; Kosta et al. 2012; Zhou et al. 2015a, 2013; Soyata et al. 2012; Yang et al. 2015; Xiang et al. 2014). These existing works focus on four main issues: the solutions of leveraging cloud resources, the efficiency of outsourcing the computation, the service availability and reliability, and the platforms for mobile cloud services.

As the hardware capabilities of mobile devices are developing rapidly, researchers have shown that the benefits of using the public cloud as an augmentation resource for the mobile cloud are

decreasing (Srirama 2017), and that network conditions can be another performance bottleneck. On the other hand, utilizing a Heterogeneous Mobile Cloud (HMC)—a hybrid of the mobile ad-hoc cloud, computers in proximity, and public clouds—as an augmentation resource can solve the issue. However, it also brings new challenges to the mobile cloud augmentation system.

Many surveys have been done on mobile cloud computing (Shiraz et al. 2013; Abolfazli et al. 2014; Sanaei et al. 2014; Li et al. 2015; Shuja et al. 2016). Shiraz et al. (2013) reviewed the existing distributed application processing frameworks for mobile devices to offload computationally intensive applications to remote servers. They presented a detailed taxonomy and survey on the offloading frameworks, including application partitioning, VM migration, partitioning granularity, migration objectives. Abolfazli et al. (2014) presented a survey on the existing mobile code offloading frameworks based on the different types of cloud-based resources, which include distant immobile clouds, proximate immobile computing entities, proximate mobile computing entities, and hybrid clouds. However, their main focus is on a cloud-based augmentation frameworks comparison, without a detailed classification and discussion of the techniques that can be used to implementa mobile cloud system. Therefore, the interoperability of the techniques mentioned may not fit the HMC environment. Moreover, the survey only provided a brief discussion on the supporting techniques of mobile cloud augmentation, such as decision-making, context monitoring, and fault tolerance. Sanaei et al. (2014) looked into heterogeneity in mobile cloud computing. They presented a taxonomy of heterogeneity for both mobile device and clouds on four levels, including hardware, platform, features, and APIs. The existing approaches for handling heterogeneity in mobile clouds were discussed. However, that work does not include discussion on realizing computation and storage augmentation of the hybrid mobile cloud. Li et al. (2015) focused on the survey of mobility-augmented cloud service provisioning and provided a taxonomy of cell network and cloud service provisioning mechanisms. Shuja et al. (2016) focused on a multimedia application-oriented survey that discussed application virtualization, dynamic binary translation techniques, and native code offloading- based mobile cloud frameworks. These two surveys provided discussion only on specific problems, and they do not meet the technical demands of the new HMC systems. Differing from these existing surveys, in this article, we first introduce the concepts and issues of the HMC, and then we present a comprehensive taxonomy and survey of software techniques applied in the HMC for both computation and storage augmentation.

The taxonomy covers the software augmentation-related techniques for computation and storage and classifies with regards to implementation an HMC system in practice. Figure 1 illustrates a typical set of modules for the HMC system framework, where the techniques presented in the taxonomy are needed. The HMC network consists of several HMC system-powered devices. On each HMC device, the *Augmentation Engine* implements the main technique for computation and storage augmentation. The application submits offloading task requests to the *Decision Engine*, which makes offloading decisions based on its decision logic and the context data observed from the *Resource Monitors* and *Profilers*. The *Scheduler* further schedules the tasks upon receiving offloading decisions from the *Decision Engine* to other HMC devices for remote execution. The *Fault Tolerant Manager* module supports the task execution in the event of failures. The *Communication* module deals with all the data-related transmission. The taxonomy presented in this paper is organized around modules of this system architecture.

In summary, the main contributions of this survey are:

- It introduces the main concepts and most challenging issues in realizing mobile cloud augmentation systems.
- It classifies the state-of-the-art techniques and works in a taxonomy of two parts regarding mobile computing augmentation and mobile storage augmentation, respectively. Each type

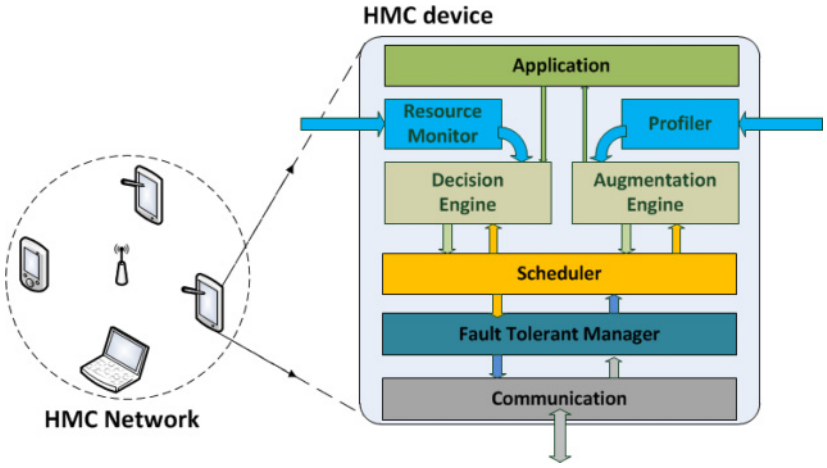

Fig. 1. Typical modules of a heterogeneous mobile cloud system framework.

of techniques and work is discussed with advantages and disadvantages, as well as the suitable conditions for adopting that type of technique.
- The taxonomy is organized following the idea of necessary modules needed for realizing a practical mobile cloud service, including outsourcing techniques, supportive decision-making mechanisms, resource management, and service reliability.
- It analyzes the gaps still existing in mobile cloud augmentation systems and discusses future directions.

The remaining article is organized as follows. We explain relevant terms and definitions, the motivation of mobile augmentation in the mobile cloud computing, and the key issues of mobile cloud augmentation in Section 2. Then we propose the taxonomy of computing and storage augmentation techniques and existing frameworks in Section 3 and Section 4, respectively, followed by an analysis of the gaps and open challenges of mobile cloud augmentation in Section 5. Finally, we summarize the article in Section 6.

2 BACKGROUND — MOTIVATION AND ISSUES

Different types of approaches have been applied for mobile computing and storage augmentation in mobile cloud computing. In this section, we explain the terms used in previous works regarding augmentation techniques. Furthermore, we discuss the motivation and key issues that have been targeted by existing works.

2.1 Terms and Definitions

Due to the variety of techniques adopted in mobile cloud computing, we explain some of the important terms and conceptions that will be mentioned throughout this article.

2.1.1 Cloud Computing. Cloud computing emerged in recent years and gained popularity among both academia and industry. NIST defines cloud computing as "a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model

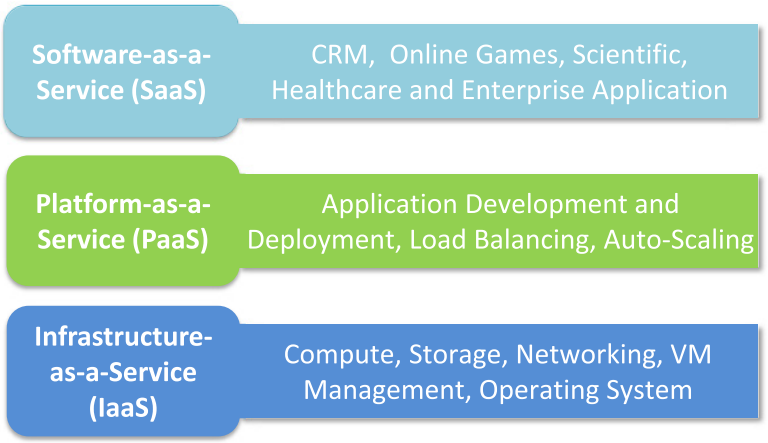

Fig. 2. Cloud service models.

is composed of five essential characteristics, three service models, and four deployment models (Mell and Grance 2011)."

A. Essential Characteristics:

On-demand self-service: Customers can quickly provision the services automatically according to their demand (e.g., process capacity, network, and storage).

Broad network access: Customers can access the cloud service over the Internet by using heterogeneous client platforms such as mobile devices, laptops, and workstations.

Resource pooling: The infrastructure providers pool many resources from data centers and provide resource renting for clients in a multitenant service manner, with multiple physical and virtual resources dynamically assigned upon consumer demand.

Highly scalable: The service provider can easily scale up and down with the resources to meet user's demand automatically or manually when needed.

B. Service Models:

Generally, there are three types of service models provided by cloud service providers. The models can be summarized in a service stack, which is shown in Figure 2. From the bottom to the top, IaaS provides virtualized data center infrastructure resources such as compute, storage, and networking. Examples include Amazon EC2, Microsoft Azure, and Google Compute Engine; PaaS provides platforms and tools for application and other development, testing, and deployment. Examples include Majsoft Aneka and Apache Stratos; SaaS represents the largest part of cloud services. It provides web-based applications where the backend services are managed on the cloud. Examples of SaaS applications include Facebook, Gmail, and Office 365.

C. Deployment Models:

Based on the difference of the administrative domain of the cloud, it can be further referred to as the *public cloud* for general public users, a *private cloud* for private groups and companies, and as a *community cloud*. Cloud computing services have been adopted by a significant amount of research work as a potential approach for resource augmentation in mobile cloud computing.

2.1.2 Mobile Cloud Computing. In the beginning, mobile cloud computing was defined as a restricted computing paradigm that considers interactions between only mobile devices and public cloud services. Later, due to the instability of mobile devices as well as wireless networks, more

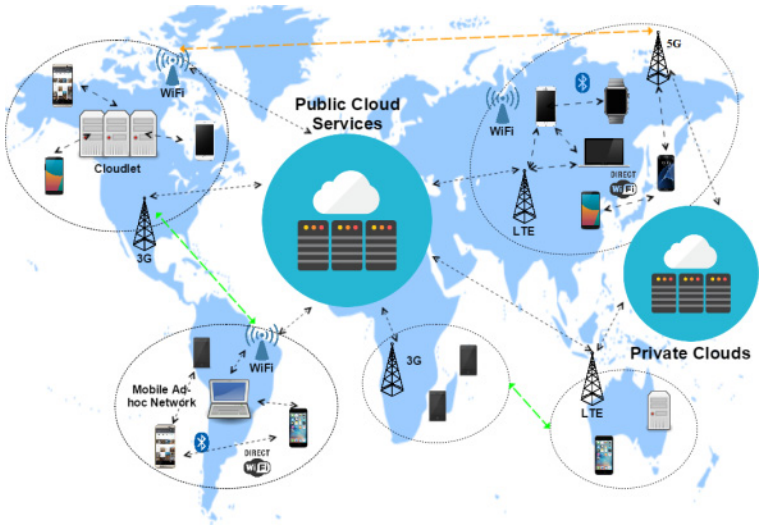

Fig. 3. An overview of the mobile cloud service environment.

types of platforms and computing resources were introduced to mobile cloud computing to achieve seamless mobile cloud services. An overview of the HMC service environment is shown in Figure 3. Based on the previous definitions and new demands, we define mobile cloud computing as follows:

> Mobile cloud computing is a computing paradigm that enables resource-constrained mobile devices to utilize heterogeneous computing resources (e.g., public clouds, private clouds, and MANETs) over multiple types of wireless networks (e.g., cellular network, WiFi, Bluetooth, and Femtocell) to provide mobile device users with a seamless, on-demand, and scalable mobile service that has rich user experience.

The term "cloud" in mobile cloud computing refers to multiple types of computing resources. A brief classification of different clouds follows.

- **Infrastructure-based cloud.** The infrastructure refers to public cloud services including IaaS, PaaS, and SaaS. The mobile device only outsources its computation and storage to the public cloud services via WiFi or cellular network.
- **Cloudlet-based cloud.** *Cloudlet* refers to a "trusted, resource-rich form-factor computer that is well-connected to the Internet and available for use by nearby mobile devices (Satyanarayanan et al. 2009)." Cloudlets are deployed as middle layer servers between a public cloud and mobile devices to reduce network latency.
- **Mobile device cloud.** This refers to a MANET, which consists of a set of mobile devices connected to each other via short-range wireless networks such as WiFi-direct and Bluetooth in dynamic topologies, with no support of networking infrastructure (Conti and Giordano 2014). A mobile device cloud can further reduce the data transmission overhead and provide augmentation services in case WLAN or public cloud services are unavailable.

2.2 Motivations

Mobile devices have already seen a significant improvement, with more powerful and functional hardware such as multicore processors, mobile graphic cards, and improved memory. Nevertheless,

realizing mobile cloud computing can bring numerous benefits to both mobile device users and cloud service providers. From the user's perspective, user experience on mobile applications and the functions of mobile devices can be improved by introducing mobile cloud computing. For cloud service providers, mobile clouds enable a large user community to use their services. Additionally, providers can apply machine learning mechanisms on the service data of mobile device users to further provide more customized cloud services.

2.2.1 Computing Capability. Despite recent hardware improvements seen in mobile devices, slow processing speed and low RAM still hinder mobile devices from providing experience-rich applications to end users (Satyanarayanan 2011). Since mobile devices have become part of many individuals' daily activities such as social networking, office work, and gaming, they are expected to have PC-approximate computing capacities. Mobile cloud computing provides opportunities for developers to overcome this gap and enrich their applications by outsourcing computing-intensive tasks to cloud resources. Moreover, by migrating computation to other computing resources, mobile devices are released to complete light tasks which eventually will improve the fluency of mobile devices.

2.2.2 Energy Efficiency. One of the most urgent technical challenges for mobile devices is battery life. Currently, lithium-ion batteries used on mobile devices can only supply a few hours of power for extensive computing. As more powerful processors, bigger displays, and different types of sensors are installed on mobile devices, larger energy consumption is expected. However, battery manufacturers are only able to increase the battery capacity by 5% annually (Robinson 2009). Multiple efforts have been provided by phone manufacturers, such as battery-saving mode that dims displays or switches off wireless interfaces when the battery level is too low. This type of approach neither solves the problem nor enhances user experiences. Therefore, mobile cloud augmentation can be a promising alternative solution that shifts computation from mobile devices to other resources to save energy consumption on the battery.

2.2.3 Mobile Storage. In the era of big data and mobile computing, many digital contents such as photos, videos, and large files have seen a drastic increase. Unlike the storage on PCs, mobile devices provide only limited flash storage space despite the demands of large and elastic storage from mobile applications and services. The elastic cloud storage resources can be a solution to lack of mobile storage. The characteristics of cloud storage, such as elasticity, on-demand, and low cost, make it a potential extension to mobile devices. However, due to unstable wireless connections and mobile device mobility, the offline usability of mobile cloud storage augmentation is a challenge. Moreover, the migration of sensitive personal data from mobile applications to cloud storage raises the concern of data security and privacy.

2.2.4 Seamless and Collaborative Mobile Application. With the advent of smart mobile devices, mobile users expect continuous mobile application experiences while using mobile cloud computing services. However, the wireless mediums that carry out data migrations are the performance bottleneck for mobile cloud services due to their instability and accessibility. In the event of network infrastructure failures, public cloud services may not be available, and mobile applications running computation intensive tasks will be interrupted. In such a case, an ad-hoc network of mobile devices as a collaborative resource pool can be an alternate solution. Another example is the Mobile Social Network in Proximity (MSNP) (Chang et al. 2012), which enables mobile device users to interact with other users nearby via peer-to-peer wireless communications and complete tasks collaboratively.

2.3 Key Issues

The trends of utilizing heterogeneous computing resources and multiple types of wireless networks in mobile cloud augmentation bring many challenges. We present the major challenges related to techniques used in realizing mobile cloud augmentation systems.

2.3.1 Outsourcing Computation. First, how to outsource computation is one of the main challenges in mobile cloud computing. Specifically, selecting the appropriate augmentation approach based on the characteristics of heterogeneous cloud resources is a nontrivial task. For instance, code offloading is one of the common approaches for outsourcing mobile tasks to cloud Virtual Machine (VM)-based resources, while remote procedure calling is more suitable for SOA-based task integration. A few existing works, such as MAUI (Cuervo et al. 2010), ThinkAir (Kosta et al. 2012), CloneCloud (Chun et al. 2011), MuSIC (Rahimi et al. 2013), and Cloudlet (Satyanarayanan et al. 2009) have been proposed to solve the issue, but the efficient use of heterogeneous computing resources is yet to be studied more comprehensively.

2.3.2 Unstable Wireless Communications. Due to the unique characteristics of wireless communication techniques, network throughput, signal range, and connection stability can be downgraded by environmental changes like signal interference when compared to wired networks. Channel capacity is another issue of wireless networks in the mobile cloud computing environment. As the number of mobile device users grows, the Service Level Agreement (SLA) regarding network performances is at risk of being violated. Therefore, handover strategies and seamless connection in case of a network SLA violation need to be studied when developing systems for mobile cloud augmentation.

2.3.3 Mobile Device Mobility. Mobile device management is vital to the performance of the mobile cloud augmentation system. On the one hand, mobile device movement, together with unstable wireless networks, can cause frequent service failures in mobile cloud augmentation systems. On the other hand, the performance of certain types of cloud resources like the mobile device cloud (i.e., mobile ad-hoc networks) depends on routing protocols that are affected by mobile device mobility. As a result, mobility management and fault tolerance need to be considered to provide a reliable mobile cloud service. Location-based device mobility prediction under different user contexts is another issue that needs to be investigated.

2.3.4 Context-Aware Augmentation Decision-Making Strategies. Outsourcing computation to cloud resources is not always beneficial because the context of the mobile cloud environment changes rapidly. As discussed earlier, device mobility and unstable wireless networks are the main reasons for change of context. Moreover, the heterogeneity of cloud resources and mobile applications can significantly affect the performance of the augmentation. For example, in computation outsourcing such as code offloading, transferring a significant amount of data for remote execution may consume more time and energy than running it on the mobile device itself. Instead, offloading this type of computation to a nearby mobile ad-hoc network may gain benefits from parallel execution and delay-free network transmission. Hence, it is necessary to develop augmentation decision-making strategies to achieve context-awareness of the mobile cloud augmentation systems.

2.3.5 Data Security. Computation and data outsourcing bring the concern of data security and privacy. Existing security challenges such as authentication, data integrity, privacy, and trust are inherited in mobile cloud computing due to the use of remote cloud resources (Zissis and Lekkas 2012). Ensuring users' privacy when running their mobile applications on unknown computing resources is the main challenge of realizing mobile cloud augmentation systems in terms of security.

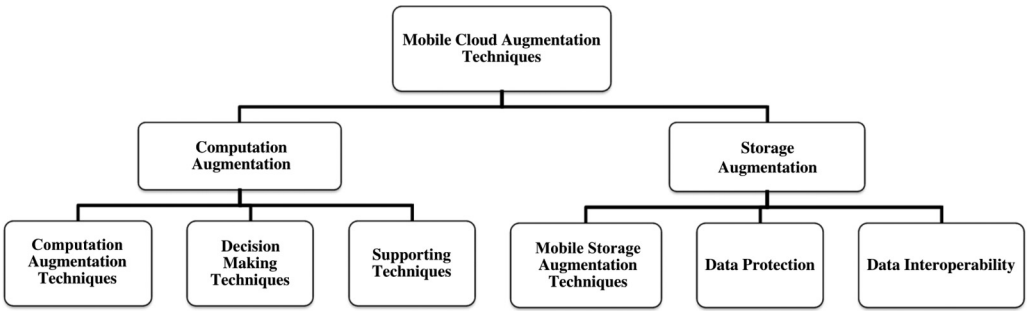

Fig. 4. A thematic taxonomy of software techniques for mobile cloud augmentation.

Many research works have proposed different techniques to tackle these issues in HMCs, and hardware augmentation technologies have improved battery life (Ali et al. 2014), processors, and the like. However, since our focus is on the development of HMC systems for mobile cloud applications, this taxonomy covers software augmentation technologies for HMC. A thematic taxonomy of software mobile cloud augmentation techniques is depicted in Figure 4. It is divided into two main categories: computation augmentation and storage augmentation. For computation augmentation, we present augmentation techniques related to task offloading for the *Augmentation Engine* in Figure 1; decision-making techniques for devising offloading decisions for the *Decision Engine*, with observed context monitoring data from *Context Monitor* and *Profiler*; and supporting techniques such as scheduling and load balancing, fault detection, and recovery for the *Scheduler* and *Fault tolerance* modules. For storage augmentation, we present techniques related to storage offloading, data protection, and data interoperability. Detailed taxonomies of computation and storage augmentation techniques are proposed in the following two sections, respectively.

3 TAXONOMY OF COMPUTING CAPACITY AUGMENTATION

One benefit of the mobile cloud computing paradigm is that it brings cloud resources to device proximity to enhance the computing capability of mobile devices. Multiple techniques can be leveraged to augment the computing capability of mobile devices. As shown in Figure 5, we discuss these techniques in two aspects: augmentation models and augmentation architectures. Augmentation models, which include code offloading models and task delegation models, provide solutions for how computation augmentation of mobile devices is realized by utilizing cloud resources. Augmentation architecture, which includes parallel execution and opportunistic mobile collaboration, describes the system architecture of the mobile cloud system for performing computation augmentation. The two aspects work together to provide efficient mobile cloud augmentation services. A detailed discussion of these techniques is presented in the next four subsections.

3.1 Code Offloading

Offloading computation for remote execution is an idea that has been studied ever since the computer network was developed. It aims at migrating computationally intensive code from resource-limited machines to remote computing resources to accelerate the running process of the computation and reduce energy consumption on limited battery devices. Figure 6 shows a general concept of code offloading. In a general code offloading model, the computation-intensive codes of a mobile application are identified first, then are evaluated by a decision0making process based on the objective of the mobile cloud augmentation service (e.g., saving energy) whether to offload or not. Last, the code is offloaded to the remote computing resources by different types of available techniques.

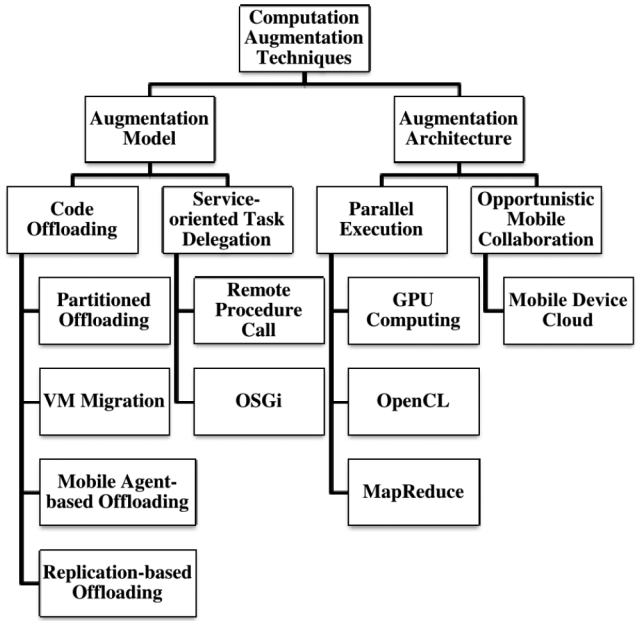

Fig. 5. Taxonomy of computation augmentation techniques.

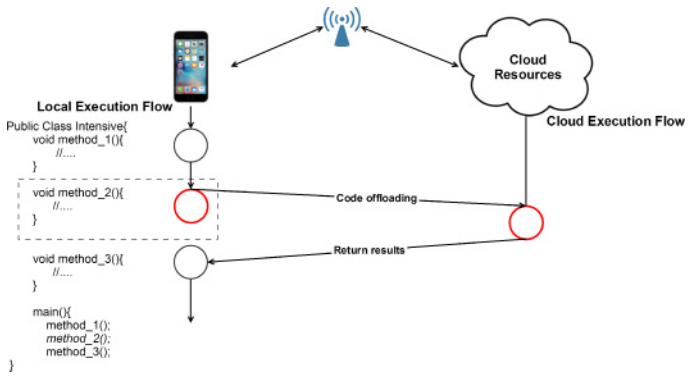

Fig. 6. Code offloading concept.

The code offloading technique assumes the entire application computation originally happens on mobile devices. It enables mobile devices to migrate part of their computation from resource-limited mobile devices to resource-rich computing machines, which makes it flexible in terms of outsourcing. However, the disadvantage of code offloading is that it requires developers to identify and partition a part of the computation to offload, which is a nontrivial task and may impose unnecessary overhead for mobile devices. Various code offloading approaches have been proposed and applied based on various programming models of mobile applications. The approaches can be classified into four categories: partitioned offloading, VM migration, mobile agents-based offloading, and replication-based offloading.

3.1.1 Partitioned Offloading. For partitioned offloading methods, only the computation-intensive part of a mobile application is identified and offloaded to the remote servers. The

conventional client-server computing model is introduced to perform the partitioned offloading for HMC augmentation. Resource-limited mobile devices are considered thin clients, while the remote computing resources that execute the offloaded codes are considered resource-rich servers. The application is partitioned either statically before runtime or dynamically at runtime. In static partitioning, offloading partitions are decided and hard-programmed in the application by the developers. As a result, there is no overhead imposed by deciding partitions when the application is running. However, static partitioning requires comprehensive program-running knowledge for developers, which is a nontrivial task. Also, the static partitions make this offloading approach less adaptive to a dynamic computing environment like mobile clouds. Therefore, static partitioning is rarely adopted in mobile cloud augmentation. To fix this issue, dynamic partitioning was introduced. Different from static partitioning, dynamic partitioning analyzes the processes of an application at runtime with the assistance of offloading decision logic. Most commonly used dynamic partitioned offloading techniques are **flow-based programming, .NET common language runtime programming, Java reflection**, and **distributed shared memory**.

Flow-based Programming (FBP). "FBP is a data flow programming paradigm that models applications as modulized processes connected with each other by pre-defined communication connections (Morrison 2010)." These modules can be reconnected with each other to develop various types of applications without having to be modified. Therefore, each module can dynamically decide whether to offload. To apply FBP in mobile cloud computing, Hung et al. (2015) ported JavaFBP onto an Android system. The Android applications are coded with APIs provided by JavaFBP in the modules. However, using FBP as a code offloading approach requires the installation on both mobile devices and offloaded machines, which hinders the scalability of the applications. To solve this problem, another approach using **Java Reflection** is proposed by Kosta et al. (2012).

Java Reflection, provided by Java, enables the program to inspect and manipulate the annotated classes, interfaces, fields, and methods at runtime. Therefore, it can be decided dynamically at runtime whether to offload classes of an Android application, and any machines with Java runtime can execute the offloaded partitions without requiring additional installations. Kosta et al. (2012) proposed *ThinkAir* for mobile cloud code offloading on the method level using Java Reflection. The annotated methods are evaluated by offloading decision logics dynamically and offloaded to cloud VMs running Android clones. In this way, the offloading services can be easily scaled up and down on cloud VMs. However, by using Java Reflection, ThinkAir is only able to offload one method at a time and may cause lock-in issues.

To overcome this lock-in problem, Gordon et al. designed another solution, *COMET* (Gordon et al. 2012), for the Android system using ***distributed shared memory***. It is built on top of the Dalvik Virtual Machine (Ehringer 2010) on Android that leverages the underlying Java memory model and VM synchronization to form a Distributed Shared Memory (DSM) for code migration between machines. The offloading is implemented by synchronizing the information of heap, memory stacks, register states, and synthetic classes between VMs on cloud and mobile devices.

In addition to Java on Android, other mobile platforms also provide similar functions that can be used for code offloading. ***.NET Common Language Runtime (CLR)*** "provides a managed coding platform featuring cross-language integration and exception handling and a simplified model for component interaction (Box and Pattison 2002)." # Cuervo et al. proposed a framework called *MAUI* that adopts .NET CLR on Windows phones to enable code offloading. Similar to Java Reflection, methods annotated by CLR tags are evaluated by MAUI's decision-making logic at runtime to decide whether to be offloaded. However, different from ThinkAir, MAUI requires users to develop a server version of the mobile application using CLR, which harms the application's scalability in mobile clouds. Moreover, MAUI lacks consideration of multi-mobile device offloading environments and the same lock-in issue as Java reflection remains.

These four types of techniques are the major approaches proposed. Compared with static code offloading, although dynamic code offloading is more flexible and able to adapt to different execution environments, it still requires modifications of mobile applications at the application development stage. This drawback makes it nontrivial and tightly coupled with the underlying programming model of mobile applications. With the development of new smartphone operating systems, many works later presented VM migration techniques for computation offloading.

3.1.2 VM Migration. The live VM migration approach (Clark et al. 2005) for code offloading is based on the idea of moving computation dynamically among machines in the distributed system without interrupting the ongoing execution. It is fairly suitable for computation augmentation in mobile clouds since most mobile applications as well as cloud services are running on virtual machines. VMs can be migrated either partially or completely based on the requirement of the offloading. Applications are not required to be installed on the server side. However, the drawback of live VM migration is the overhead of transferring a VM image, and its state via wireless networks can be costly and inefficient under unstable network conditions. Hence, the focus of VM migration technique is improving energy efficiency while considering network conditions. Several previous works have proposed solutions based on live VM migration.

Satyanarayanan et al. proposed a VM migration-based framework called Cloudlet[1] (Satyanarayanan et al. 2009) to bring computation from the mobile device to nearby form factor servers to overcome the long latency caused by transferring data to the cloud via wireless networks. The term *Cloudlet* refers to "trusted, resource-rich computer or cluster of computers that is well-connected to the Internet and available for use by nearby mobile devices." *Dynamic VM synthesis* performs the transient partial VM migration service. The benefit of using Cloudlets is that this can be an alternative to remote cloud resources to reduce network bottleneck, which often has significant effects on mobile cloud performances. However, the absence of monetary incentives to install the Cloudlet-like machines is an issue to be further studied. Moreover, Cloudlets can only provide offloading service to stationary users. The continuous execution of offloaded tasks would be an issue.

Chun et al. (2011) took a different approach to using VM migration from Cloudlets. They proposed a code offloading framework, *CloneCloud,* that works with the application VM layer, such as DalvikVM and Microsoft .NET. It deploys one or more clones of the mobile applications and data onto Cloud VMs and nearby servers and uses process interception on the VM level to execute parts of the application on the cloud. The running states, including data in the stack and heap, are synchronized between the two processes on the mobile device and cloud VM. However, since the mobile clones have limitations on native data virtualizing, it does not have all the access to data on the cloud side, which narrows the range of functions to be offloaded. This is also a vital issue for all VM migration-based code offloading approaches.

3.1.3 Mobile Agent-Based Offloading. In order to overcome the data access limit of VM migration for code offloading, mobile agent-based offloading was introduced to mobile cloud computing. A mobile agent is movable software that can be transferred from one mobile device to a network and roam among the computing nodes in the network (Chess et al. 1997). When an application using mobile agents needs to request a service from a remote server, like the cloud, it collects the application execution information and passes it to the execution environment of the agent for offloading and remote execution. There are several advantages of using mobile agents in mobile cloud computing. First, the mobile agent uses asynchronous communication to interact with remote servers in the network, which is suitable for mobile devices since the network connections of

[1]Source code can be found on https://github.com/cmusatyalab.

mobile devices are unstable. Moreover, the device can still perform other lightweight computation while waiting for the results from remote execution. Second, mobile agents are robust to server failures since all the information, including executing environment, process states, and services required, is transferable. Some mobile agent-supported frameworks have been proposed.

Angin and Bhargava (2013) proposed Java Agent Development Environment (JADE)[2] for mobile code offloading. In the event of task offloading, the method being offloaded is transferred into an agent-based process by the feature *Behaviour* in JADE on either the class or method level. Then it consults the *cloud directory service* to select one of the cloud hosts for remote execution. JADE provides the offloading code with high movability and running adaptively. However, JADE does not consider the Quality of Service (QoS) of its cloud resources, such as load balancing, application multitenancy, and wireless channel energy efficiency for multiusers.

The multitenancy issue of mobile agent-based offloading is investigated by Liu et al. (2016). They considered the energy efficiency of wireless channels with multiuser scenarios and solved the problem by using Lyapunov optimization framework to minimize the number of transmission time slots based on the queue backlog and channel states on each mobile device. However, this optimization approach only focuses on data transmission without considering local and cloud task execution, which may have a significant impact on the efficiency of the proposed algorithm.

3.1.4 Replication Based-Offloading. Either partitioned offloading, VM migration, or mobile agent-based offloading requires applications to precisely analyze the benefits of whether to run portions of the application on mobile devices or remote machines. As a result, it usually benefits applications with large computation and small data size. In addition, the preceding three techniques do not benefit network-intensive applications. In order to solve this problem, Gordon et al. (2015) took a different approach by using replications. They proposed a framework called Tango for Android that attempted to reduce user-perceived application latency by switching execution and output display automatically between an application and its replica on remote machines for the faster one. However, there are still some limitations to Tango. Only one replica can lead the execution, instead of working simultaneously. This issue can affect the application running time as the leading replica may have no access to native resources such as sensors and user inputs. Also, the overhead of constantly switching is not presented in the work.

Discussion. The above-mentioned types of augmentation techniques adopted for code offloading aim to dynamically offload computation-intensive code to cloud resources in order to conserve energy and speed up execution. However, some code offloading approaches, such as partitioned offloading, require particular environments and comprehensive programming skills, which limits the usage on existing applications. Moreover, the lock-in issue of code offloading exists for approaches such as Java reflection and FBP. Also, the decision process for offloading incurs additional overhead that will reduce its benefits.

3.2 Service-Oriented Task Delegation

Unlike the code offloading approach, service-oriented task delegation augments resource-limited mobile devices by utilizing existing services on remote servers with remote process invocation rather than migrating part of the application to the server for execution. OASIS has defined Service-Oriented Architecture (SOA) as "A paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains. It provides a uniform means to offer, discover, interact with and use capabilities to produce desired effects consistent with measurable preconditions and expectations" (MacKenzie and Laskey 2006).

[2]Source code available at: http://jade.tilab.com/download/jade/.

A service is a self-contained unit that implements at least one reusable action, such as searching a database or rendering a web page. The request to a service and its corresponding response are communicated via protocols such as XML and Simple Object Access Protocol (SOAP). The benefit of SOA for mobile cloud computing is that it provides a solution for multiple service providers and heterogeneous devices to incorporate their services into more comprehensive service combinations, where services are independent of any underlying vendors, products, or techniques. The transparency of SOA-based services can prevent the execution lock-in issue of code offloading. SOA enables conventional applications to deliver their services to mobile devices via cloud resources with the same user experience as on PCs, without the need for developing native mobile applications.

3.2.1 Web Service-Based Mobile Cloud. In the age of cloud computing, many legacy PC applications have been migrated to the cloud to provide their original functions in the form of web services. This SOA-based service paradigm also enables mobile device users to use legacy applications on their devices with the same PC-like user experiences via mobile cloud augmentation systems. One benefit of web service-based mobile cloud systems is that there is no need to develop a native mobile application, but only a thin client to relay the service requests to the cloud servers and render the returned results. In this way, the device is released to process other tasks and is still able to obtain the same good quality of service. Many recent works have been proposed in mobile cloud augmentation. Hani and Dichter (2017) proposed a four-layer service-oriented architecture to provide an energy-efficient solution for web service-based mobile cloud augmentation. The architecture ensures the security of the data handover and guarantees QoS. Rossi et al. (2017) designed a cloud-based service architecture to provide geolocated emergency services on mobile devices. Mobile users report real-time conditions about the emergency via mobile devices to the backend services. Crowdsourcing techniques are applied to evaluate the circumstances on the cloud. Guerrero-Contreras et al. (2017) studied the service availability in mobile cloud computing. They proposed a context-aware SOA-based framework to improve service availability, which can be downgraded by dynamic network changes in mobile cloud systems. The framework contains three services running on the cloud: monitoring, context managing, and replica managing. As we can observe from these proposed works, one drawback of SOA-based mobile cloud augmentation is service binding, which means mobile users can only choose certain provided services.

In an attempt to solve the operating binding issue, Flores and Srirama (2014) proposed mobile cloud middleware *(MCM)*[3] for processing intensive hybrid cloud services regardless of the underlying mobile operating systems. The middleware provides a set of cloud service APIs available to the mobile device users to build the applications in its own language. MCM also fosters the integration and orchestration of mobile tasks delegated with minimal data transfer. However, the cloud services provided in MCM are limited, and the ability of users to add additional cloud services is yet to be delivered.

3.2.2 OSGi (Open Service Gateway Initiative). Another technique adopted in the SOA-based mobile cloud is OSGi[4] (Alliance 2009), which is a Java module management system enabling applications to dynamically load and unload service bundles at runtime. The difference of OSGi from the C-based RPC is that OSGi is object-oriented. The bundles can be invoked by users with a service interface, which is registered in the OSGi service registry with an implementation of the interface. Figure 7 depicts the OSGi system stack. Bundles are OSGi components built by developers. Execution Environment defines the available methods and classes for the user's

[3]Source code available at: https://github.com/huberflores/MobileCloudMiddleware.
[4]More details at https://www.osgi.org/.

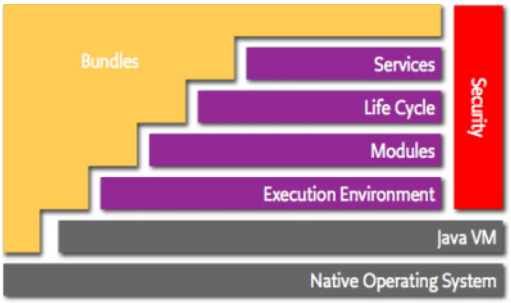

Fig. 7. Layers of OSGi programming model (Alliance 2009).

application. Modules define the way Bundles import methods and classes. Life Cycle provides APIs to install, start, stop, update, and uninstall bundles. Services dynamically bind the bundles with underlying Java objects. Services, life cycle, modules, and execution environment together define how the bundles work with underlying Java VMs.

Verbelen et al. (2012) proposed an OSGi-based middleware called *AIOLOS*[5] for mobile augmentation on Android. The bundle feature is ported to Android. The mobile application is developed with OSGi bundles, service interfaces, and remote services that implement the bundle services. Junior et al. (2017) proposed a mobile offloading system (MOSys) to enable seamless offloading when users move between wireless networks. It is implemented based on OSGi, utilizes software-defined networking and remote caching to reduce the response time, and provides a seamless handover. A few other works (Yang et al. 2016; Wang et al. 2016) have also presented OSGi-based techniques to support mobile cloud augmentation systems.

Discussion. A SOA-based mobile cloud is fundamentally different from code offloading as there is no computation migration. Mobile devices in SOA work as thin clients that are only able to send service requests without any computation, while the main task computation is carried out in the form of services on the cloud. However, this limits the flexibility of mobile applications in terms of functions because they need backend service support.

3.3 Parallel Execution

More than one zettabyte of data is generated over the Internet nowadays (IBM 2016). The needs to process big data surpass the capacity of mobile devices. Big data applications such as mobile forensics applications, data streaming applications, and augmented reality applications have been challenging conventional mobile cloud computing augmentation approaches. Although code offloading and SOA-based task delegation can enhance mobile devices with more computing capability, the overhead of migrating a large amount of data to the remote server is getting higher and higher. As a result, the benefits of outsourcing computation are counteracted.

On the other hand, utilizing parallel execution to distribute computation can significantly reduce the time overhead of handling big data applications. In parallel computing, the computation and related data are divided into subproblems with a chunk of data, which are then processed simultaneously on multiple resources. Regarding data-parallel applications, sets of either homogeneous or heterogeneous tasks are processed in parallel, and the results are merged to generate the final result. The benefits of using parallel execution for mobile cloud computing augmentation depend on the types of applications requiring parallel data processing. Frameworks such as **GPU**

[5]The framework is available to download at http://aiolos.intec.ugent.be/.

computing, **OpenCL** and **MapReduce** have been adopted to cope with big data applications in mobile cloud augmentation.

3.3.1 GPU Computing. The thousands of cores on a GPU enable an application to offload compute-intensive portions to the GPU while the remainder of the application still runs on the CPU. Soyata et al. (2012) proposed a mobile cloud-based hybrid architecture (MOCHA) using GPU computing to enhance the performance of object recognition applications used on the battlefield. Typically, battlefield handheld devices are connected via satellites that incur long latency. The Cloudlet in MOCHA is used to reduce the latency by letting mobile devices connect to a Cloudlet in the vicinity via low-latency links; then the Cloudlet processes either part or all of the computation. GPUs are utilized to perform the massively parallel processing (e.g., object recognition), which provides massively parallel processing ability to applications by using CUDA,[6] which is a C-like programming platform, and programming models designed for GPU parallel programming. However, GPU computing is only suitable for mobile applications that involve matrix-based computation (e.g., image processing due to the structure of cores on GPU). In addition, only intensive computation on small size data instead of streaming applications is preferred so that the speedup on execution will not be hindered by data transmission overhead. More importantly, GPU computing is only available with certain brands of GPU, which causes vendor lock-in.

3.3.2 OpenCL. Different from GPU computing that only works with particular types of applications, OpenCL (Bourd 2016) provides a more general parallel solution. It is an open-source platform for building parallel programs on cross platforms that provides transparent C-compatible programming models for utilizing distributed processors regardless of the underlying architectures. Shih et al. (2015) proposed an elastic computation framework for mobile clouds based on OpenCL frameworks. It federates computing and memory resources on wearable devices, mobile devices, nearby cloudlets, and public cloud VMs as a shared resource pool for completing tasks on mobile devices. Eom et al. (2013) studied the feasibility of applying HPC cluster architectures to the mobile cloud computing environment. They proposed an OpenCL-based offloading framework that migrates OpenCL workloads from the mobile grid to clouds. OpenCL provides more portability and compatibility compared to GPU computing since it works on both CPU and GPU.

3.3.3 MapReduce. Another approach adopted as an augmentation technique for mobile cloud computing is MapReduce. "MapReduce is a programming model and an associated implementation for processing and generating large data sets with a distributed algorithm on a cluster" (Dean and Ghemawat 2008). Marinelli (2009) developed a mobile-cloud computing infrastructure called Hyrax based on Hadoop. Hyrax enables smartphone applications to run in distribution, both regarding data and computation, by adapting Hadoop onto mobile devices within the Hyrax system. However, NameNode and JobTracker instances are not realized in Hyrax. In Hyrax, these must be run on a traditional machine. Dou et al. (2010) implemented the MapReduce paradigm on Nokia smartphones using Python. The master node, including a scheduler, an HTTP server, and an application repository, is implemented on a server to map the jobs to other mobile devices as worker nodes. MapReduce is not widely used in mobile cloud computing since the processing capability of mobile devices is insufficient for a complete MapReduce implementation.

Discussion. Different from the previous two augmentation techniques, parallel execution in mobile cloud aims to reduce transmission overhead by breaking up big data into smaller chunks and processing them individually at distributed nodes at the same time in order to speed up execution for resource-limited mobile devices. However, techniques like GPU computing have

[6]Download available at http://www.nvidia.com/object/cuda_home_new.html.

limitations on the types of applications that can be run as well as vendor lock-in issues, which makes parallel execution restricted.

3.4 Opportunistic Mobile Collaboration

The mobile cloud augmentation techniques discussed so far focus on utilizing remote computing resources via wireless networks like WiFi. However, most of these techniques rely on adequate wireless network bandwidth, which can be intermittent and cost-unfriendly. In addition, since mobile devices are usually on the move and may lose Internet connections in areas lacking signal strength, communication between mobile devices and cloud services may be disrupted.

In recent years, the short-range wireless communication technologies on mobile devices such as Bluetooth, WiFi-direct, and Zigbee have been significantly improved regarding bandwidth and energy efficiency. With the development of both mobile device computing capacity and wireless networks, the MANET has become a potential mobile cloud infrastructure for augmentation. Many proposed works consider MANET as a **mobile device cloud**, which is a network of mobile devices connected wirelessly in either a centralized or distributed manner (Toh 2001). The opportunistic mobile collaboration network provides a solution for mobile devices to dynamically utilize other peer devices in the vicinity to overcome the issue of unstable wireless network connections.

Many research works (Penner et al. 2014; Mao et al. 2016; Ravi and Peddoju 2014) have been carried out on applying MANET to enhance mobile device performance. Penner et al. (2014) proposed a collaborative computing platform, "Transient Clouds," to enable mobile devices in the vicinity to share their own mobile application services with other devices in the manner of ad-hoc networks. The prototype was developed on Android using the built-in WiFi-direct feature. The tasks are stored in the auto-generated .dex file and later distributed to other devices by the decision engine. However, due to the limit of WiFi-direct, the ad-hoc network implemented is a one-hop network, which is not a complete implementation of the proposed framework; its feasibility needs further evaluation. Mao et al. (2016) utilized mobile edge device computing to offload computation in order to reduce network latency and congestion, and they proposed an online Lyapunov optimization-based dynamic offloading algorithm to minimize execution cost by adopting DVFS. Zhou et al. (2015b) proposed a multi-tier mobile cloud service framework that includes a mobile device cloud. The mobile device cloud is used for code offloading to obtain timely task execution and lower energy consumption for mobile cognitive applications like OCR, especially in the case of unavailable cloud services. The above-mentioned mobile device implementations all have only one type of wireless medium for communication, which leaves them vulnerable to network disruptions and also will not be scalable to devices with different wireless interfaces. Therefore, mobile device clouds that can utilize devices with various wireless mediums need to be investigated.

Discussion. The mobility and dynamic nature of MANETs carry reliability issues that need the support of fault-tolerant policies for stabler performance. Approaches addressing fault tolerance issues in mobile cloud augmentation systems are discussed in Section 3.8. In addition, MANET performance can be affected by network congestion as well as by synchronizing the up-to-date information of the mobile devices.

3.5 Task Offloading Decision-Making

To provide efficient mobile task offloading services, decision-making techniques are required to decide whether, when, and where to offload. In Section 3.5 and Section 3.6, existing decision-making techniques regarding task offloading and task allocation and scheduling are discussed respectively. Figure 8 shows a detailed taxonomy of decisionmaking techniques, including offloading decisions (whether to offloading) and task allocation and scheduling decisions (where to offload). For

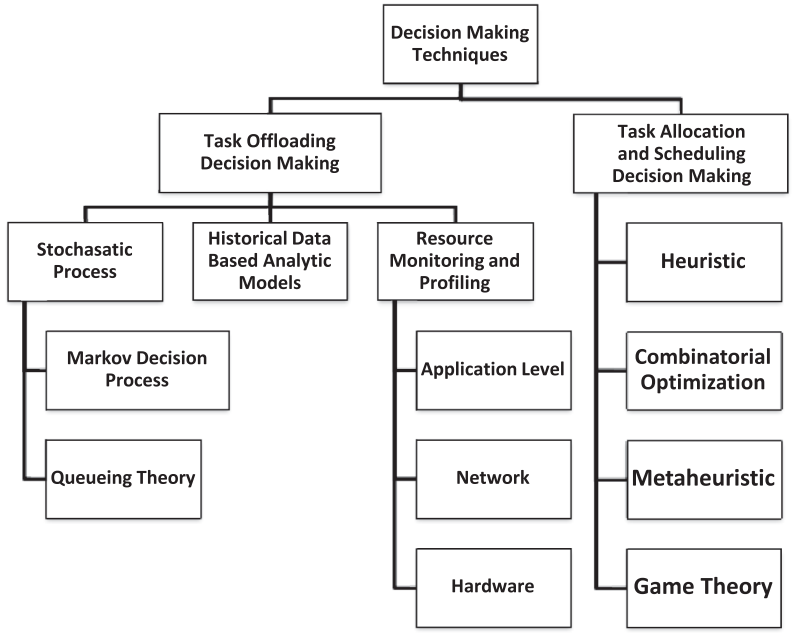

Fig. 8. Taxonomy of decision-making techniques.

making offloading decisions, three main categories are discussed: stochastic process, analytic model, and resource monitoring and profiling. For making task scheduling decisions, the techniques are classified into four categories: heuristic, combinatorial optimization, metaheuristic, and game theory.

In this section, the task offloading decision-making techniques are classified into (i) stochastic processes, which abstract systems with random variables and statistically devise offloading decisions as systems evolve into stable states with certain probabilities; (ii) historical data-based analytic models, which aim to abstract systems with parametric models based on historical data and devise real-time offloading decisions (unlike the stochastic processes); and (iii) resource monitoring and profiling, which provides real-time monitoring and profiling of the system and running applications and utilizes real-time data to devise offloading decisions without analysis on parametric models.

For both parametric model-based techniques and real-time monitoring data-based techniques, different factors of a mobile cloud system need to be captured in order to perform valid analysis on task offloading decisions. The factors can be classified as mobile application contents, contexts of the mobile cloud environment, characteristics of the hardware, and user preferences.

Mobile application contents. The contents refer to characteristics of applications, including application type, code granularity, data size and type, and user interaction requirements, as well as computation intensity. These application contents have important impacts on the results of augmentation. For instance, applications with computationally intensive jobs are more suitable for cloud servers if outsourced. Data streaming applications such as video streaming and optical character recognition require short response delay, while data analytic applications such as sentinel surveillance in the medical field are more delay-tolerant. Hence, it is important to consider the differences in application contents when making augmentation decisions.

Context of mobile cloud environment. The mobile cloud environment is a resource sharing environment that consists of mobile devices, remote servers, and various wireless communication mediums. For example, wireless communication is one of the key components when outsourcing computation, and bandwidth and congestion will affect the time taken to transfer data, which will hinder augmentation performance. Moreover, the mobility of mobile devices adds dynamics to the environment because the available resources are changing rapidly. Thus, decision-making schemes should be agile to support dynamic context changes.

Hardware specifications. Hardware refers to both mobile devices and remote servers, which have heterogeneous CPUs, memory, storage, and communication modules. The challenge for decision-making schemes is to distribute tasks among machines to either balance between time and energy saved by augmentation or to fulfil user preferences, all based on hardware as well as other factors discussed. Therefore, monitoring schemes are required to profile the hardware information to support the decision-making schemes.

User preferences. Last, but not the least, user preferences, which are priority objectives for decision-making schemes, must be taken into consideration. User preferences can be very different depending on circumstances. Some users want to use the augmentation service to save battery life, while other users wish to outsource mobile tasks to get a shorter processing time regardless of how much battery it will consume.

3.5.1 Stochastic Process. The stochastic process is a random process evolving with time (Cinlar 2013). It aims to abstract the evolution of a system that changes based on random variations, and it predicts possible outcomes weighted by probability. Some well-known stochastic processes include Markov processes, the Poisson process, and queueing theory. In mobile cloud augmentation systems, a stochastic process is applied to model mobile applications as well as to devices statistically to evaluate execution conditions to help devise optimal augmentation schedules.

Chen et al. (2013) proposed a Semi-Markovian Decision Process (SMDP)-based method to solve the problem of optimal offloading tasks dispatch and transmission for mobile cloud augmentation systems. Its objective is to obtain an optimal tradeoff between computation time and energy consumption. The SMDP-based optimization is developed to abstract mobile device status, decide the probability of tasks to offload, and, at each stable state, which DVFS level of CPU and data rate of the transmitter to be set. The SMDP is solved with linear programming. One drawback is that many mobile devices have no access for users to change DVFS and wireless interface status.

Instead of focusing on a single device, Terefe et al. (2016) proposed an energy-efficient multisite offloading policy for mobile clouds using a Markov decision process (MDP). MDP is used to formulate application partitioning and scheduling to a multi-cloud as a delay-constrained, least-cost shortest path problem with the goal of minimizing energy consumption on wireless channels. Similarly, Wang et al. (2017) applied a Markov model in dynamic scheduling for mobile cloud health telemonitoring. As the mobile cloud infrastructure in health monitoring faces ever-changing clinical priorities and personal demands, the MDP-based dynamic offloading scheduling algorithm proposed produces joint optimization of processing latency, energy efficiency, and diagnostic accuracy. Stochastic processes such as MDP can provide optimal solutions to multiobjective problems. Nonetheless, the processing overhead of solving the problem and the sensitivity of the solution to new changes may impose a burden on resource-limited mobile devices.

3.5.2 Historical Data-Based Analytic Models. Unlike stochastic processes, this approach uses parametric models to abstract multiple attributes (parameters) of the mobile cloud augmentation environment and analyzes the benefits of augmentation in terms of time and energy consumption based on historical data on task execution. The attributes range from CPU speed of mobile devices and remote servers and network conditions (e.g., bandwidth, congestion, and latency) to the load

of remote resources. However, the disadvantage of analytic models is that the types of attributes used in the models have to be consistent with the model's objective. Both too many attributes without proper assumptions and too few attributes without accurate abstraction may downgrade model performance. Many previous works applied this method to evaluate the benefits of mobile cloud computing augmentation.

Ali et al. (2016) focused on modeling the power consumption of mobile devices to enhance the energy efficiency of mobile cloud augmentation systems. They presented detailed energy consumption models for CPU, display, wireless communication interfaces, and memory on the mobile devices. For example, the energy model for a CPU is based on CPU frequency and utilization:

$$Power_{cpu} = \beta^{freq} \times U + \beta_{base}^{freq}, \tag{1}$$

where β^{freq} and β_{base}^{freq} denote frequency-dependent coefficients when the CPU is in busy and idle states, respectively. U is the CPU utilization. All the coefficients in the energy models are derived from a linear regression on the measurement results of smartphones. However, this regression approach is device-dependent and may not be accurate when applied to other mobile devices.

Rahimi et al. (2013) took into consideration device mobility when making augmentation decisions by modeling mobile applications as *location-time workflows*. The mobile device is assumed to move in a partition of a 2D area. The mobility-sensitive workflow models are derived by integrating the locations (i.e., coordinates) of the mobile device and time duration of the device on that location into the workflow models. It is represented as follows:

$$W(u_k)_T^L \triangleq \left(w(u_k)_{t_1}^{l_1}, w(u_k)_{t_2}^{l_2}, \ldots, w(u_k)_{t_n}^{l_n}\right), \tag{2}$$

where u_k is the k^{th} mobile user and $w(u_k)_{t_n}^{l_n}$ is the user workflow in location l_n for time t_n. Such models can assist mobile cloud augmentation services to meet QoS requirements considering mobile device movements. However, the accuracy of the proposed models needs to be further studied and improved since the mobile user trajectory is obtained from some conventional mobility models such as Random Point Walk and Manhattan models (Camp et al. 2002).

Chen (2015) proposed a set of analytic models for processors and wireless network interfaces in terms of both computation and communication. For the computation models, the author proposed execution time models and energy consumption models which are both related to the computation size of tasks:

$$T_n^l = \frac{D_n}{F_n^l}, \; E_n^l = \nu_n D_n, \tag{3}$$

where D_n is the computation amount, ν_n is the coefficient of the energy consuming rate per CPU cycle, and F_n^l is the computation capability. Similarly, for the communication, the models are proposed based on the size of input data B_n, the network speed R_n, and wireless network power consumption rate P_n of user n.

3.5.3 Resource Monitoring and Profiling. To overcome the issues of context adaptability in an analytic model approach, resource monitoring and profiling are adopted in mobile cloud computing. This approach involves the implementation of multiple profilers and monitors on mobile devices to constantly monitor the context changes and behavior of the mobile device. Then, the obtained data are put through the decision logic to evaluate execution benefits and make augmentation decisions based on the objective and user preferences. Many works have been proposed using this approach. We discuss a few representative ones.

Benkhelifa et al. (2016) proposed a genetic algorithm-based resource augmentation for mobile device cloud to minimize energy consumption. A social cloud resource profiling and negotiating system is implemented, including a logger module which focuses on profiling the energy usage

of mobile device applications under different circumstances, such as using different wireless interfaces, executing tandemly with other applications, and executing at different times of the day. However, the process of gathering a large amount of data on storage-limited mobile devices is not presented.

On the other hand, history-based resource profiling with a compressed data storage model can solve this issue. Kaya et al. (2016) implemented a set of profilers to monitor application usage such as running time and CPU usage on mobile devices and cloud VMs. The history-based profiles are then constructed in a call graph model that axes redundant information to reduce the storage. In case the application has never been executed before, the profiling mode is invoked first and run several times with all possible use cases. However, this process requires unnecessary extra time for application users, which affects user experiences.

ThinkAir (Kosta et al. 2012) implements a set of profilers on Android that observe a more comprehensive range of system parameters to assist its *Execution Controller* for augmentation, including hardware profilers, application profilers, and network profilers. All the profilers are implemented using the system APIs provided by Android.

Discussion. In this section, we discussed three major types of offloading decision-making techniques. The stochastic process provides a statistical approach to devise offloading decisions based on the states of the system and the transition probabilities obtained from different distributions. The accuracy of statistic distributions abstracting the mobile cloud system can have a significant impact on the offloading decision. The analytic model approach has a similar component of system abstracts as the stochastic processes, but instead it devises offloading decisions based on historical execution data of the system rather than statistic distributions. Unlike these two approaches that are not agile to context change in the system environment, the resource monitoring and profiling approach makes offloading decisions based on real-time observations of task executions in mobile cloud systems. The main difference among the three approaches is the execution data that they use to make offloading decisions. Note that these three approaches can be used in combination to provide more accurate results.

3.6 Task Allocation and Scheduling Decision-Making

In an HMC environment, each mobile device can outsource tasks via augmentation techniques to other resources using wireless networks. After the execution monitoring on the context parameters is completed, mobile tasks submitted from mobile devices need to be allocated to execute either locally or be offloaded to other computing resources. Moreover, these tasks also need to be scheduled among the shared resources pool to achieve the objective of the system, such as optimal task completion time or minimum energy consumption. Therefore, task allocation and scheduling algorithms are required to distribute mobile tasks among shared resources. Although there have been many task-scheduling algorithms developed for distributed computing environments, such as grid and cloud computing, the unique characteristics of HMCs (such as intermittent wireless networks, unstable devices, and human-related behaviors) make it difficult for conventional scheduling algorithms to be applied. Hence, new algorithms and mechanisms are needed. Many task-scheduling algorithms have been adapted to solve task allocation and scheduling problems in HMCs. They can be classified into four categories: **heuristics, combinatorial optimization, metaheuristics, and game theory**, based on their approach to solving the allocation and scheduling problem, as shown in Figure 8.

Heuristics are search algorithms that rank and select solution subjects at each step of the searching process based on the most current information. They are often considered a faster, less complex type of algorithms for solving optimization problems (Pearl 1984). This is suitable for execution on mobile devices that require lightweight processes. Greedy algorithms, listing heuristics, and

Min-Min algorithm are among many of the most used scheduling algorithms proposed for mobile cloud augmentation. Zhou et al. (2015a) adopted the Min-Min algorithm to select the machine with the least response time and energy consumption as the offloading destination for independent task offloading. Li et al. (2015) proposed six online and batch scheduling-based heuristics for scheduling independent tasks onto mobile nodes in a mobile device cloud to reduce energy consumption. Trneberg et al. (2017) proposed an iterative local search algorithm to find near-optimal solutions for application placement on mobile cloud resources. However, in exchange for processing efficiency, heuristics are only able to provide local optimal solutions because they have no knowledge of the future events.

By contrast, combinational optimization can devise global optimal solutions for task-scheduling optimization problems in mobile cloud augmentation. It searches the entire space of the feasible solution candidates and finds the best one as the optimal solution. The computation efficiency of this method can be exhaustive if the search space is significantly large. Therefore, it usually deploys on a more powerful tier of computing resources in HMCs. Methods such as (integer) linear programming, dynamic programming, and A* search algorithm have been applied in mobile cloud augmentation. Gai et al. (2016) presented a dynamic programming-based algorithm to minimize the energy consumption of wireless communication in mobile cloud systems. Yang et al. (2016) solved a joint optimization of task placement and load balancing with linear programming and further proposed a greedy heuristic with low complexity. As we can observe, heuristics are possible alternatives when combinational optimization is infeasible to solve in real time. Another alternative is an approximation algorithm, which aims to reduce the original problem to a similar, conventional problem and is solved with approximation algorithms to the reduced problem. Some recent works have proposed solutions with this method (Sardellitti et al. 2015; Xu et al. 2016; Kao et al. 2017).

Another alternative is a metaheuristic, which is a high-level algorithm to select and guide a heuristic to generate near-optimal solutions. It often refers to nature-inspired algorithms, such as the genetic algorithm, ant colony algorithm, and simulated annealing. Metaheuristics usually adapt randomization and stochastic processes in each of their evolutionary searches to reduce the search space and generate solutions in a reasonable amount of time. Many recent works have been proposed to solve task-scheduling problems in HMC augmentation using metaheuristics such as the genetic algorithm (Goudarzi et al. 2016; Benkhelifa et al. 2016; Gai et al. 2017), swarm optimization (Goudarzi et al. 2017; Qi et al. 2016), and ant colony optimization (Rashidi and Sharifian 2017; Wei et al. 2016).

Game theory-based methods can also be leveraged to solve optimization problems, especially for mobile cloud augmentation where mobile users can be considered as players in task outsourcing games. These methods can abstract the HMC in a more realistic approach in which the mobile device users can apply some strategies to maximize their interest. The game theoretic scheduling algorithm aims to find the optimal task augmentation schedules that can achieve a Nash equilibrium so that no strategy change from one player can change the results of optimal solutions. A few recent works have presented game theory-based algorithms for task offloading and scheduling in mobile cloud systems (Chen 2015; Chen et al. 2016; Tang et al. 2017).

Discussion. The task allocation and scheduling algorithms proposed in mobile cloud augmentation systems aim to provide optimal mobile task execution solutions to achieve system objectives, such as optimal execution time and minimized mobile device energy consumption. As discussed, the performance of these algorithms depends on the types of mobile applications, user preferences, SLAs, and the context of mobile cloud augmentation systems. Therefore, algorithms need to be designed while considering all the factors discussed.

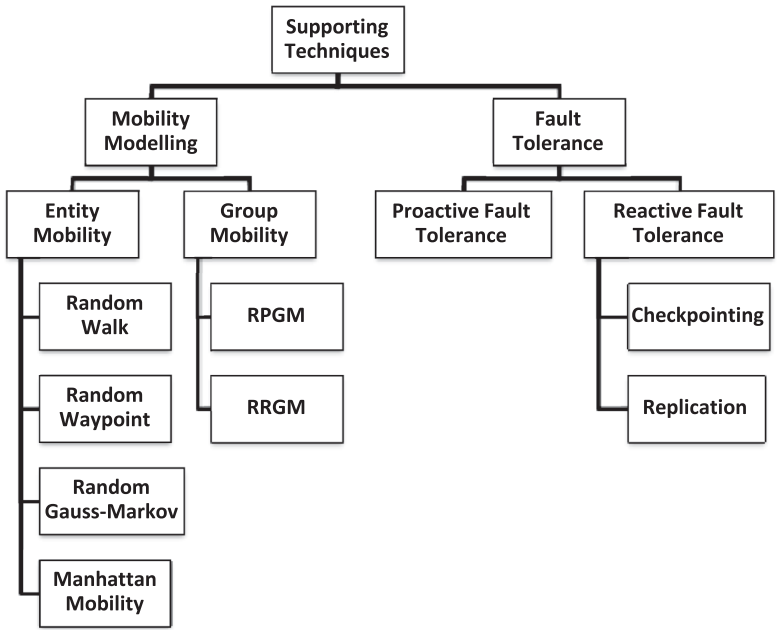

Fig. 9. Taxonomy of supporting techniques for computation augmentation.

3.7 Mobility Model for Mobile Device Cloud

In the following two sections, we discuss the mobility impact of mobile devices and fault tolerance techniques caused by a device's mobility. These are the supporting techniques for HMC augmentation. A detailed taxonomy of the supporting techniques are presented in Figure 9. Mobility modeling is important for mobile cloud computing, especially for the mobile device cloud which consists of an ad-hoc network of mobile devices. The decision-making logics in a mobile cloud augmentation system can utilize the information obtained from mobility models to estimate the availability and reliability of mobile devices in the network and further make augmentation decisions. There are two types of mobility models commonly used: trace-based models and synthetic models. Trace-based models refer to those models built on real-world moving traces, which contain accurate information of mobility behaviors. However, real-world traces require long observation periods and a large number of participants. Therefore they are only useful case by case. On the other hand, synthetic models try to capture mobility patterns without the assistance of traces. Synthetic mobility models are categorized as entity mobility models and group mobility models.

3.7.1 Entity Mobility Models. Entity models aim to mimic the moving patterns of a single mobile node without interaction with other nodes in the proximity. We present four most commonly used entity models: the Random Walk model, Random Waypoint model, Random Gauss-Markov (RGM) model, and Manhattan mobility model. A few available simulators such as ns2, ns3 can be used to simulate node mobility.

The Random Walk model was first introduced by Pearson (1905) in 1905. It describes a walk path that consists of a series of random steps on a single- or multiple-dimension space. In the mobile cloud scenario, it is usually a 2D space. At each stepping point, the mobile node chooses a random direction from $[0, 2\pi]$ and a random speed from a range set by the model, so that the patterns can be limited to a certain area. The Random Waypoint model is very similar to Random

Walk. The only difference is that Random Waypoint adds a randomly chosen pause time to every stepping point. These two mobility models have been widely applied in mobile cloud computing (Taleb et al. 2016; Li and Li 2014; Mavromoustakis et al. 2015). However, from the descriptions, we can observe that Random Walk model is a stateless random process that does not remember information from previous steps. This property makes Random Walk generate unrealistic walking patterns since there may be sudden stops or sharp turns.

To solve the sudden stops and turns issue, the RGM model can be applied in mobility modelling for mobile cloud augmentation. RGM moves a mobile node in time intervals. At each time interval, the next location d_{next} and speed s_{next} are calculated based on its current location d_{pre} and speed s_{pre}:

$$s_{next} = \alpha s_{pre} + (1-\alpha)\overline{s} + \sqrt{(1-\alpha^2)s_{ran}} \tag{4}$$

$$d_{next} = \alpha d_{pre} + (1-\alpha)\overline{d} + \sqrt{(1-\alpha^2)d_{ran}} \tag{5}$$

where α is the tuning parameter to vary the randomness. $\overline{s}, \overline{d}$ represent the mean values, and s_{ran}, d_{ran} are two random variables from a Gaussian distribution. RGM avoids the sudden change issue by letting past states influence future states. A few works have implemented RGM (Qiao et al. 2017; Antonescu et al. 2013; Miyake and Kami 2015).

For some special mobile devices, like those on vehicles, nodes move on an urban grid which includes only horizontal and vertical movements. The Manhattan mobility model captures such movement patterns. In this model, each mobile node at each intersection is allowed to choose to go straight with a 0.5 probability or to turn left or right a with 0.25 probability. This model is implemented in Rahimi et al. (2013) and Jeong et al. (2016).

3.7.2 Group Mobility Models. Different from entity mobility models, group mobility models consider the influence that mobile devices have on each other that may affect their movement. In mobile cloud systems, especially mobile device cloud, nodes tend to move together, such as a tourist group or a group of soldiers. Two commonly applied models are the Reference Point Group Mobility (RPGM) model and the Reference Region Group Mobility (RRGM) model.

RPGM (Hong et al. 1999) is one of the most used group-based mobility models. Each group will have a central node as the leading node which follows an entity mobility model and sets the speed and direction for the entire group. The other mobile nodes in the network will be paired with a reference point that follows the leader point's movement in same direction and speed. When the reference points move to the new location, its associated points will move to a random location within a circle of radius R around the reference point. A few RPGM-based mobility models have been proposed for mobile cloud augmentation in Huang et al. (2010) and Ammari (2006). However, the RPGM model should avoid using Random Walk-based models that could generate sudden stops.

RRGM (Ng and Zhang 2005) further extends RPGM to use a real-time determined sequence of regions to lead the group to some destination. The reference region is determined dynamically by user-defined node density and the size of the group. Members linked to the reference region will move to a random location in the region and then follow a random waypoint model to wait until all members arrive in the region. The group will eventually move to the target destination following the reference regions. The advantage of RRGM is that it can mimic the real scenario where a large group can be divided into subgroups and merge back together after movement.

3.8 Fault Tolerance

As a computing environment with heterogeneous mobile devices, remote servers, and wireless network interfaces, maintaining continuous service to avoid and recover from service interruptions

caused by failures is a difficult challenge. Existing fault tolerance strategies proposed in distributed computing can be applied to the mobile cloud augmentation environment, with adaptations to its unique challenges such as device mobility and loss of wireless connections.

Mobile devices play a significant part in mobile cloud systems because they are not only the consumer of the systems, but also can be resource providers (i.e., mobile device cloud). As resource providers, mobile devices can enter and leave the system environment unpredictably if it moves out of the wireless network range or the network connection is disrupted. Therefore, faults considered in the context of mobile clouds are related to the mobility and availability of the mobile devices and remote servers as well as to the stability of wireless network conditions. There are two main types of fault tolerance strategies: proactive fault tolerance and reactive fault tolerance.

3.8.1 Proactive Fault Tolerance. Proactive fault tolerance aims to prevent system faults by monitoring the system to predict potential faults and prevent them from taking place. When a node failure is indicated, the fault tolerance mechanism preemptively migrates parts of the mobile application from the nodes about to fail. The basic form of proactive fault tolerance applies feedback-loop control over the distributed nodes in the system (i.e., mobile devices and cloud servers). The feedback loop consists of continuous node monitoring and reallocation of application partitions. However, proactive fault tolerance policies predict only before the task is dispatched, with no further action if failure happens during task execution.

Hummel and Jelleschitz (2007) proposed a proactive and reactive combined fault tolerance mechanism for the ad-hoc mobile grid. The proactive fault tolerance policy uses two mechanisms: *redundant execution* and *super-peer access*. Super peers refer to nodes that are constantly available and able to perform critical tasks. The unstable nodes use super-peer access to perform the execution and usually apply redundant execution on more than one node. Park et al. (2011) proposed a Markov process-based approach to analyze and predict resource states to improve the system's resistance to fault problems related to the mobility of mobile devices. Ravi and Peddoju (2014) presented a handoff strategy for offloaded mobile tasks that proactively monitors the execution conditions of the mobile cloud augmentation system. The proactive failure evaluation adopts a fuzzy logic-based multicriteria decision-making algorithm. The handoff strategy is decided by monitoring energy consumption produced by using services from other resources and the remaining available time of mobile devices.

3.8.2 Reactive Fault Tolerance. On the other hand, reactive fault tolerance policies aim to reduce the effect of system faults that already happened. The most applied reactive fault tolerant mechanisms are *checkpointing* and *replication*.

Checkpointing allows applications to restart at the most recent checkpoint when a failure is detected. Sonara (Chen et al. 2012) is a platform that aims to provide continuous mobile cloud augmentation service. Sonara's execution engine enables a fault-tolerant distributed runtime that performs a checkpointing-based partial rollback recovery scheme. It adopted a snapshot protocol (Chandy and Lamport 1985) that periodically carries out global checkpointing throughout the system.

Replication is another well-known fault tolerance mechanism. In order to keep the service operating continuously after system faults occur, multiple replicas of the task are distributed and run on different resources until the task is completed. However, replication brings the challenge of added redundancy into the system, as well as synchronization problems. Choi et al. (2014) proposed a fault tolerance scheduling algorithm for a Content Addressable Network (CAN)-based mobile cloud augmentation system. This is done using cloud service replication. When the cloud server receives a request for cloud service, the server returns with two or more proper resources from other mobile devices that can meet the QoS and operate the service on all the resources.

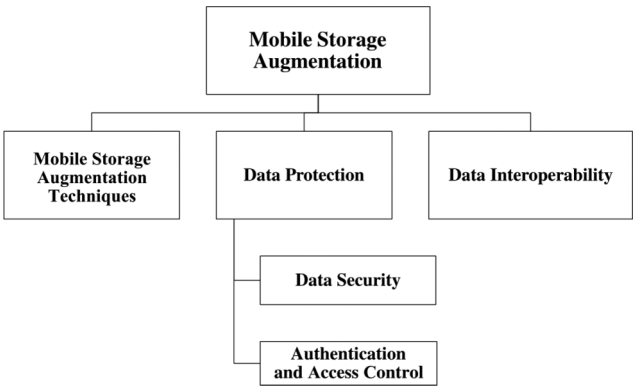

Fig. 10. Taxonomy of mobile storage augmentation.

A summary of the existing frameworks implemented in mobile cloud computation augmentation techniques is presented in Table 1.

4 TAXONOMY OF MOBILE STORAGE AUGMENTATION

Mobile devices are equipped with limited storage that cannot match mobile applications' growing need for larger data storage. On the other hand, the public cloud also provides Data-as-a-Service (DaaS). DaaS enables users to offload a large amount of data to cloud storage services that mobile device storage cannot provide. For example, Amazon web service provides Simple Storage Service for object storage and charges based on usage. Therefore, data storage augmentation by storing mobile data on cloud servers is of interest for mobile cloud augmentation systems. Transmission of sensitive data, such as location coordinates and healthcare information, via wireless communication media and data storage on other computing resources bring security challenges into mobile cloud computing. We present a taxonomy of mobile storage augmentation in Figure 10.

4.1 Mobile Storage Offloading

Despite the fact that mobile devices have improved regarding hardware, the lack of substantial storage and computing capacity are still hindering them from better user experiences. The cloud has become a primary resource for enhancing mobile devices in terms of computation and storage. Many approaches and techniques have been adopted in mobile cloud storage augmentation to solve problems with storage augmentation, data distribution, data access, and data synchronization.

Most commonly used mobile storage augmentation solutions involve public cloud storage services. To extend mobile storage, the solutions propose middlewares between cloud services and mobile devices to provide data offloading and data management functions. Zheng et al. (2010) presented a novel cloud storage augmentation framework called *SmartBox*. It introduces a concept called "shadow storage" services to extend the storage on mobile devices. The shadow storage service will automatically back up the data stored on mobile devices to the cloud when the device is connected to SmartBox and share data between different mobile devices. Hung et al. (2015) proposed a mobile storage augmentation system based on Trusted Architecture for Securely Shared Service (TAS3) (Kirkham et al. 2011). It is a secure network for user-centric storage that uses a rule-based policy framework to let service users store and process their private data in distributed applications on both mobile devices and cloud servers. However, mobile storage augmentation using public cloud services heavily relies on wireless networks, which is disadvantageous in case of wireless network outages.

Table 1. Comparison of Mobile Cloud Computation Augmentation Frameworks

Framework	Cloud Resource Type	Code Offloading	SOA Task Delegation	Parallel Execution	Opportunistic Collaboration	Context-aware Execution Monitoring	Task Allocation & Scheduling	Mobility Management	Fault Tolerance
MobileFBP	Nearby servers	Flow-based programming	–	–	–	Resource profiling	Single user, heuristic	–	–
MAUI	Nearby servers	.Net common language runtime	–	–	–	Resource profiling	Single user, minimum makespan and energy, ILP	–	Checkpoint
ThinkAir	Public clouds	Java reflection	–	–	–	Resource profiling	Single user, heuristic	–	Memory error detection
COMET	Public clouds	Distributed shared memory	–	–	–	Thread execution profiling	Single user, heuristic	–	Checkpoint
Cloudlet	Nearby servers	Live VM migration	–	–	–	–	Single user, heuristic	–	–
Clonecloud	Public clouds	Live VM migration	–	–	–	Analytic models	Single user, minimum makespan and energy, ILP	–	–
JADE	Public clouds	Mobile agent(Java)	–	–	–	Analytic models	Single user, heuristic	–	–
Scavenger	Nearby servers	Mobile agent(Python)	–	–	–	Resource profiling	Multi users, heuristic	–	–
Spectra	Nearby servers	–	Remote procedure call(RPC)	–	–	–	Single user, heuristic	–	Proactive prediction on usage demand
Cuckoo	Nearby servers	–	Android interface defined language	–	–	Not required	Single user, heuristic	–	–
SAMI	Public clouds	–	SOA-based middleware	–	–	Not required	Multi users, heuristic	–	–
MCM	Public clouds	–	SOA-based middleware	–	–	Not required	Multi users, heuristic	–	–
AIOLOS	Public clouds	–	OSGi	–	–	Not required	Multi users, heuristic	–	Checkpoint
AlfredO	Mobile device cloud	–	OSGi	–	–	Not required	Multi users, heuristic	–	Redundant execution
MOCHA	Mobile device cloud and public clouds	–	–	GPU computing with CUDA	–	Analytic models	Multi users, heuristic	–	–
Hyrax	Mobile device cloud	–	–	MapReduce	–	–	Single users, heuristic	–	Redundant execution
(Shih et al. 2015)	Mobile device cloud and public cloud	–	–	OpenCL	–	Analytic models	Multi users, heuristic	–	Imprecise computation model
(Eom et al. 2013)	HPC clusters	–	–	OpenCL	–	Resource profiling and discovery	Single user, heuristic	–	Monitoring error code implemented in each method
mCloud	Mobile device cloud, Cloudlet, public cloud	Live VM migration	–	–	Ad-hoc network	Resource profiling and discovery	Single user, heuristic	–	Heartbeat, checkpoint
Transient Clouds	Mobile device cloud	–	–	–	Ad-hoc network	Resource profiling and discovery	Multi users, heuristic	–	–
Handoff scheme	Mobile device cloud and public cloud	–	–	–	Ad-hoc network	Resource profiling and discovery	Multi users, heuristic	–	Fuzzy multi-criteria decision making based handover strategy
Follow-Me Cloud	Public clouds	–	SOA-based middleware	–	–	Resource profiling	Single user, minimum makespan, MDP	Random Walk Model	–
(Qiao et al. 2017)	Mobile device cloud	–	–	–	Ad-hoc network, edge computing	–	Multi-user, MDP	Gauss-Markov model	–
MobiCloud	Mobile device cloud	–	SOA-based middleware	–	Ad-hoc network	Resource profiling	–	RPGM	–

To avoid this wireless network bottleneck, some approaches applied mobile device clouds as storage augmentation resources. Phoenix (Panta et al. 2013) is a protocol designed to make opportunistic use of mobile devices in the MANET to provide a short-term storage service to clients in the proximity of each other. The distributed and asynchronous protocol breaks content into blocks and stores copies of blocks on multiple mobile devices so that it can ensure some degree of storage redundancy despite hardware failures, device mobility, and wireless communication failures. Additionally, Phoenix implements an advertisement model for maintaining and managing the blocks among the distributed mobile device storage. Similarly, Chen et al. (2015) proposed a fault-tolerant data storage management algorithm for the mobile device cloud that only contains mobile devices. The algorithm introduces a "k-out-of-n" strategy, which is a well-studied reliability control strategy (Coit and Liu 2000). The strategy distributes multiple copies among the number of n mobile devices and ensures that the system operates correctly as long as task copies are available.

Instead of using a mobile device cloud to solve the network bottleneck, Cui et al. investigated wireless network bandwidth and data sync traffic directly. They proposed a system called *QuickSync* to improve the synchronization inefficiency problem in mobile cloud storage augmentation services. Three key components are implemented: namely, Network-aware Chunker (**NC**), Redundancy Eliminator (**RE**), and Batched Syncer (**BS**) to reduce the redundant data generated by sync traffic in the network.

4.2 Data Protection

For most mobile applications, such as healthcare and OCR applications, that benefit from mobile cloud augmentation services, the transmitted data often contain sensitive and private information and therefore needs to be protected. We discuss three major related issues: namely, data security, accessibility, and authentication.

4.2.1 Data Security. The three important attributes of data security are confidentiality, integrity, and availability.

Confidentiality guarantees that sensitive data is only exposed to users with proper authority. One commonly used approach for confidentiality is data encryption. By encrypting data with private information (e.g., public/private key pair), only authenticated users have access to the data. Examples include SSL/TLS and HTTPS, which are communication protocols for secure information exchange.

Data integrity maintains the consistency and accuracy of data. The most widely applied methods for protecting data integrity include data hashing technologies, such as MD5 and SHA that capture the signature of the original data with certain hashing methods and compare it with the received data.

Data availability refers to the property that authorized entities can exclusively access the information on demand. A common solution for improving data availability is doing regular offline data backups, which are used to restore data and services when the original data are damaged or under attack, as in a DDoS attack. Many works have been proposed for mobile storage augmentation to ensure the above-mentioned three attributes of data security. Common topics include encryption, access control, authentication, data synchronization, and privacy.

Zhou and Huang (2013) present a new Privacy Preserving Cipher Policy Attribute-Based Encryption (PP-CP-ABE) to ensure the confidentiality of data. The encryption algorithm is based on CP-ABE (Bethencourt et al. 2007), which is used to simplify the process of key pairs generation and access control. However, CP-ABE requires intensive computation that is not suitable for mobile devices. PP-CP-ABE provides a solution to this problem by outsourcing encryption and decryption operations to the cloud; it preserves privacy by performing the last step of decryption at the decryptors.

Yuan and Yu (2014) investigated data integrity checking techniques for cloud data sharing with multiuser data modification. They proposed an integrity checking scheme with constant computational cost by using polynomial-based authentication tags that allow aggregation of tags of different data segments. Therefore, the scheme can tag files from different users in batches on the cloud and ensure constant performance with scalability. Wang et al. (2013) proposed a similar approach but without the need of an entire data file for integrity checking. This is enabled by utilizing re-signatures for file blocks on a public verifier.

Li et al. (2013) adapted Chase and Chow's Attribute-Based Encryption (ABE) scheme (Chase and Chow 2009) for mobile cloud computing with substantial communication and computation overhead reduction to provide a low-complexity multi-authority ABE scheme (MA-ABE) for mobile devices sharing storage. The overhead reduction is made by introducing a cloud server-based semi-trusted authority where the computation of keys for encryption and decryption takes place.

Differing from the preceding approaches, Khan et al. (2014) consider the limited resource on mobile devices when adopting data encryption for mobile cloud computing. They proposed an increment-based proxy re-encryption scheme that improves file modification operations. A trusted entity was introduced so that the re-encryption services are offloaded to it instead of being performed on the mobile device itself.

4.2.2 Authentication and Access Control. In mobile cloud storage augmentation systems, mobile users usually share files among multiple computing resources as well as with other mobile users. To support the data protection requirements, mechanisms for access control and authentication need to be provided to ensure that only verified user groups have access to certain files.

Zhou and Huang (2013) proposed a set of access control schemes called Attribute Based Data Storage (ABDS) for energy-efficient and secure storage augmentation services in mobile cloud computing. ABDS access control is managed by an access policy tree that consists of leaf and branch nodes. The leaf nodes represent parameters that carry the information of the access request, and each branch node is a logical gate, such as "AND" and "OR." The mobile users requesting the MSA services will be determined by each access policy tree defined for different user groups.

Lomotey and Deters (2013) designed a framework called *ALILI* to solve the group file-sharing problems in mobile cloud systems. ALILI aims at ensuring data synchronization and user authentication among mobile device users as well as cloud service platforms. The authentication is based on the OAuth 2.0 mechanism to provide users with secure tokens and basic information retrieved from social media credential servers. It also makes the proposed authentication approach simplifies the process to locate the user's shared files.

Wang et al. (2015) proposed a key distribution mechanism for mobile devices in the Internet of Things by considering real-time data collection and monitoring. The proposed mechanism secures real-time key distribution in batch for the parallel mobile services while keeping the communication cost consistent with the number of mobile clients.

4.3 Data Interoperability

The heterogeneity of mobile cloud augmentation systems brings the problem of data interoperability. Since mobile devices may have different operating systems and hardware settings, the application data and APIs for communicating with each other may vary in data formats. Issues such as data interoperability between various service APIs on cloud and mobile devices, data portability among different types of data warehouse facilities, and data migration from mobile devices to cloud services or across different cloud service vendors need to be studied.

One solution is to apply standardized service frameworks and message exchanging techniques such as SOA, REST, XML, and JSON to mobile augmentation systems. Abolfazli et al. (2012)

proposed a SOA-based mobile device cloud called *MOMCC*. Since SOA defines standard service APIs for mobile devices and cloud services, and services apply SOAP to exchange messages, mobile applications only need a small amount of modification to provide services across heterogeneous platforms.

Mobile Social Networks (MSN) occur when mobile device users having common interests connect and interact with each other through their mobile devices. In MSNs, the heterogeneity of software platforms on mobile devices and intrinsic user data and content raise the need to develop a uniform mobile application platform. Toninelli et al. (2011) proposed a middleware called Yarta for mobile social systems. To achieve this, a representative model based on the Resource Description Framework[7] is presented. Mobile social applications developed with the representative model can share and reuse their respective data interoperably.

Doukas et al. (2010) presented a mobile healthcare application framework for Android OS that provides trusted healthcare information online storage, retrieval, and update using cloud services. The data management of healthcare data in mobile cloud augmentation systems involves problems such as data privacy and interoperability. They presented a data management system called *HealthCloud* that implements a series of REST APIs for utilizing storage services on the cloud. To ensure data security, all transferred data are compressed and sent via SSL-enabled links.

5 GAP ANALYSIS AND FUTURE DIRECTIONS

Although many aspects of mobile cloud augmentation systems have been studied in previous works, technique gaps still exist between the proposed solutions and comprehensive mobile cloud services. In this section, we identify the technique gaps and related challenges, and we present future directions in mobile cloud augmentation ecosystems.

5.1 Heterogeneous Mobile Cloud Service

As mobile devices become prevalent, various types of smart mobile devices such as wearable devices, smart home appliance, mobile phones, and tablets have significantly affected the ecosystem of mobile cloud augmentation. Different types of mobile devices have different capabilities. For example, wearable devices are regarded as sensors that provide user-related information without further processing and analysis, while smartphones have the computing capacity to execute more complicated mobile tasks. Therefore, a multi-tier Mobile Augmentation Service (MAS) framework that utilizes resources from all types of available mobile devices, as well as from cloud resources, is of interest. The framework needs to be general enough to adapt mobile devices and machines that run different operating systems. Moreover, the heterogeneity of mobile devices and servers in terms of system and data format and the problem of data interoperability must be studied.

5.2 Context-Awareness

Within MAS systems, context information such as user mobility, social information among users, network conditions, and device information can provide additional assistance for decision modules to devise more comprehensive mobile augmentation solutions based on different objectives. The multi-tier heterogeneous MAS frameworks make it possible to enable MAS providers to process the context of MAS to improve the quality of their services and allow opportunities for mobile cloud social-aware applications. However, the rapidly changing execution environment hinders the efficiency of continuous context monitoring and analysis that produces considerable overhead on mobile devices. Therefore, designing resource- and energy-efficient task allocation and scheduling mechanisms for multi-tier MAS systems is necessary.

[7] https://www.w3.org/TR/rdf-primer/.

5.3 Quality of Service Management

The QoS in mobile augmentation services refers to criteria such as service response time (delay), constant wireless communication, availability and scalability of services, the fair use of services, and mobility management. For mobile device-based systems, the fundamental problem for improving QoS is mobility management. Hence, an efficient mobility management scheme to estimate a mobile device's available time within the MAS system is crucial. With the support of mobility management, we can further improve service response time by utilizing computing resources such as other mobile devices and Cloudlets in the vicinity based on the mobile task requirements. On the other hand, it is challenging to design a service model that can manage a large number of mobile clients and a wireless communication system while ensuring the availability of services. Therefore, efficient and continuous provisioning of resources and services in mobile augmentation is a research perspective that needs investigation.

5.4 Reliability of Mobile Cloud Augmentation Systems

The HMC system is a dynamic computing environment in that the availability of computing resources and network conditions may change constantly. To provide mobile device users with reliable and seamless mobile cloud services, fault tolerance mechanisms are required to reduce the impact of service outages. However, only a few fault tolerance schemes are proposed targeting the specific type of mobile cloud environment (i.e., mobile ad-hoc network). Therefore, new reliability mechanisms adopting both proactive and reactive fault tolerance schemes need to be investigated for the nested computing environment of mobile clouds.

Containers have drawn increasing attention in public cloud services and businesses. They provide a light virtualization approach as an alternative to VMs. Containers can run on multiple operating systems and have high portability. Future research can investigate the possibility of engaging container technology with mobile cloud computing to improve the reliability and scalability of mobile cloud services.

5.5 Security and Privacy

Due to data transmission between mobile devices and other computing resources like clouds and mobile devices in the vicinity, data safety and privacy are important concerns. Despite the enormous amount of research on security issues in the cloud, it is still one of the major gaps in mobile cloud-based systems. First, the security and privacy mechanism needs to be lightweight to reduce overhead on mobile devices. Second, due to the large amount of data transmission over wireless networks, data integrity and confidentiality in must be considered. Last, but not least, a trustworthy distributed computing model must cope with computation taking place on the remote server and mobile devices and prevent unauthorized access and potential data leakage.

5.6 Incentives for Service Users

To build a fair-use mobile augmentation service, an incentive mechanism is required to convince mobile device users to opt-in to the system. The incentive mechanism needs to provide proper motivation for mobile device users to add their resources to the shared resource pool within the mobile cloud augmentation system. However, finding efficient motivation mechanisms has been difficult because of individual concerns about information security and privacy, limited battery life, and differing demands on the mobile augmentation service. Therefore, designing an appropriate incentive mechanism for mobile cloud augmentation systems is of interest for future research.

5.7 Virtual Reality, Augmented Reality, and Artificial Intelligence on Mobile Devices

The larger displays and more powerful hardware on mobile devices make it widely popular to build Augmented Reality (AR) and Virtual Reality (VR) mobile applications. These types of mobile applications require constant data streaming of the captured frames from cameras and an always-on display. In addition, Artificial Intelligence (AI) technologies are used to enable devices to use cognitive recognition (e.g., of human beings) to discover useful information from the captured data. Recent research has seen development in this field. FlashBack (Boos et al. 2016) is a system proposed for VR head-mounted devices to precompute and cache all possible encountered images to reduce the computational burden on the device GPU. Furion (Lai et al. 2017) is a mobile VR framework that enables QoS-focused application development on mobile devices using cloud offloading. MobileDeepPill (Zeng et al. 2017) is a mobile AR system for smartphones to help identify unknown prescription pills captured by the phone's camera through the use of a proposed deep learning image recognition algorithm. The constant data streaming and high computational requirements of AI from these types of application create challenges for mobile devices regarding energy efficiency, data streaming management, prestored data management for reuse, and network throughput optimization. Designing efficient mobile cloud offloading systems for these applications can provide possible solutions.

6 SUMMARY

With the prevalence of mobile devices and the development of improved hardware capability, especially in smartphones, mobile device users demand a more comprehensive and advanced user experience from mobile applications that mobile devices are not able to provide in terms of computing capacity and storage. Augmenting the computing capabilities and storage of mobile devices is a promising solution to fill the gaps. The ultimate goal of mobile augmentation solutions is to provide scalable, continuous, and secure PC-like services for mobile users regardless of underlying limited mobile device resources.

This article presents a comprehensive taxonomy and survey on the augmentation techniques applied in mobile clouds. First, the terms and definitions used throughout the survey were explained, including cloud computing and mobile cloud computing concepts. Then, current issues and challenges in the literature are presented. Second, existing mobile cloud augmentation techniques for both computation and storage were discussed in two detailed taxonomies. The taxonomy of computing capacity augmentation discussed the approaches and techniques applied in mobile cloud augmentation to merge hybrid cloud resources into a shared resource pool for mobile devices to provide reliable and energy-efficient computation outsourcing through a mobile-cloud-as-a-service. The advantages and disadvantages of each type of technique are compared, and future directions to tackle the disadvantages are discussed at the end of each section. Regarding mobile storage augmentation, the taxonomy discussed the data-oriented architecture for storing data on distant clouds as well as the mobile device cloud. Moreover, it covered the most critical issues in mobile cloud augmentation systems, namely data protection and data interoperability. As mobile computing technologies evolve rapidly, new applications and techniques are released to mobile users, which can create technical gaps between current mobile cloud services and new user demands. The survey analyzed seven major technical gaps for further study, including service heterogeneity to incorporate new mobile devices, such as wearable devices and IoT devices; service context-awareness and QoS management to agilely adapt the change of device context; reliability and security management of mobile cloud systems, because modern smartphone operating systems such as Android and iOs can be vulnerable; incentive mechanisms; and AI-empowered VR/AR mobile applications using mobile cloud services.

ACKNOWLEDGMENTS

We thank Rodrigo N. Calheiros, Amir Vahid Dastjerdi, and Satish Narayana Srirama for their comments on improving this article.

REFERENCES

Saeid Abolfazli, Zohreh Sanaei, Erfan Ahmed, Abdullah Gani, and Rajkumar Buyya. 2014. Cloud-based augmentation for mobile devices: Motivation, taxonomies, and open challenges. *IEEE Communications Surveys & Tutorials* 16, 1 (2014), 337–368.

Saeid Abolfazli, Zohreh Sanaei, Muhammad Shiraz, and Abdullah Gani. 2012. MOMCC: Market-oriented architecture for mobile cloud computing based on service oriented architecture. In *Proceedings of 2012 1st IEEE International Conference on Communications in China Workshops (ICCC)*. IEEE, 8–13.

Farhan Azmat Ali, Pieter Simoens, Tim Verbelen, Piet Demeester, and Bart Dhoedt. 2016. Mobile device power models for energy efficient dynamic offloading at runtime. *Journal of Systems and Software* 113 (2016), 173–187.

Mushtaq Ali, Jasni Mohamed Zain, Mohammad Fadli Zolkipli, and Gran Badshah. 2014. Mobile cloud computing & mobile battery augmentation techniques: A survey. In *Proceedings of 2014 IEEE Student Conference on Research and Development*. IEEE, 1–6.

OSGi Alliance. 2009. OSGi-the dynamic module system for Java. Accessed May 25, 2009.

Habib M. Ammari. 2006. Using group mobility and multihomed mobile gateways to connect mobile ad hoc networks to the global IP Internet. *International Journal of Communication Systems* 19, 10 (2006), 1137–1165.

Pelin Angin and Bharat K. Bhargava. 2013. An agent-based optimization framework for mobile-cloud computing. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications* 4, 2 (2013), 1–17.

Alexandru-Florian Antonescu, Andre Gomes, Philip Robinson, and Torsten Braun. 2013. SLA-driven predictive orchestration for distributed cloud-based mobile services. In *Proceedings of 2013 IEEE International Conference on Communications Workshops (ICC)*. IEEE, 738–743.

Elhadj Benkhelifa, Thomas Welsh, Loai Tawalbeh, Abdallah Khreishah, Yaser Jararweh, and Mahmoud Al-Ayyoub. 2016. GA-based resource augmentation negotation for energy-optimised mobile ad-hoc cloud. In *Proceedings of 2016 4th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*. IEEE, 110–116.

John Bethencourt, Amit Sahai, and Brent Waters. 2007. Ciphertext-policy attribute-based encryption. In *Proceedings of the 2007 IEEE Symposium on Security and Privacy (SP'07)*. IEEE, 321–334.

Kevin Boos, David Chu, and Eduardo Cuervo. 2016. FlashBack: Immersive virtual reality on mobile devices via rendering memoization. In *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*. ACM, New York, 291–304.

Alex Bourd. 2016. The OpenCL Specification. (March 2016). https://www.khronos.org/registry/cl/specs/opencl-2.2.pdf.

Don Box and Ted Pattison. 2002. *Essential .NET: The Common Language Runtime*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Tracy Camp, Jeff Boleng, and Vanessa Davies. 2002. A survey of mobility models for ad-hoc network research. *Wireless Communications and Mobile Computing* 2, 5 (2002), 483–502.

K. Mani Chandy and Leslie Lamport. 1985. Distributed snapshots: Determining global states of distributed systems. *ACM Transactions on Computer Systems* 3, 1 (Feb. 1985), 63–75.

Chii Chang, Satish Narayana Srirama, and Sea Ling. 2012. *An Adaptive Mediation Framework for Mobile P2P Social Content Sharing*. Springer Berlin, 374–388.

Melissa Chase and Sherman S. M. Chow. 2009. Improving privacy and security in multi-authority attribute-based encryption. In *Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS'09)*. ACM, New York,121–130.

Chien-An Chen, Myounggyu Won, Radu Stoleru, and Geoffrey G. Xie. 2015. Energy-efficient fault-tolerant data storage and processing in mobile cloud. *IEEE Transactions on Cloud Computing* 3, 1 (Jan 2015), 28–41.

Shuang Chen, Yanzhi Wang, and Massoud Pedram. 2013. A semi-markovian decision process based control method for offloading tasks from mobile devices to the cloud. In *Proceedings of 2013 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2885–2890.

Xu Chen. 2015. Decentralized computation offloading game for mobile cloud computing. *IEEE Transactions on Parallel and Distributed Systems* 26, 4 (April 2015), 974–983.

Xiuwei Chen, Ivan Beschastnikh, Li Zhuang, Fan Yang, Zhengping Qian, Lidong Zhou, Guobin Shen, and Jacky Shen. 2012. *Sonora: A Platform for Continuous Mobile-cloud Computing*. Technical Report.

Xu Chen, Lei Jiao, Wenzhong Li, and Xiaoming Fu. 2016. Efficient multi-user computation offloading for mobile-edge cloud computing. *IEEE/ACM Transactions on Networking* 24, 5 (October 2016), 2795–2808.

David Chess, Colin Harrison, and Aaron Kershenbaum. 1997. *Mobile Agents: Are They a Good Idea?* Springer Berlin, 25–45.

SookKyong Choi, KwangSik Chung, and Heonchang Yu. 2014. Fault tolerance and QoS scheduling using CAN in mobile social cloud computing. *Cluster Computing* 17, 3 (2014), 911–926.

Byung-Gon Chun, Sunghwan Ihm, Petros Maniatis, Mayur Naik, and Ashwin Patti. 2011. Clonecloud: Elastic execution between mobile device and cloud. In *Proceedings of the 6th Conference on Computer Systems*. ACM, 301–314.

Erhan Cinlar. 2013. *Introduction to Stochastic Processes*. Courier Corporation.

Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hansen, Eric Jul, Christian Limpach, Ian Pratt, and Andrew Warfield. 2005. Live migration of virtual machines. In *Proceedings of the 2nd Conference on Symposium on Networked Systems Design & Implementation - Volume 2 (NSDI'05)*. USENIX Association, Berkeley, CA, 273–286.

David W. Coit and Jia Chen Liu. 2000. System reliability optimization with k-out-of-n subsystems. *International Journal of Reliability, Quality and Safety Engineering* 07, 02 (2000), 129–142. arXiv:http://www.worldscientific.com/doi/pdf/10.1142/S0218539300000110.

Marco Conti and Silvia Giordano. 2014. Mobile ad hoc networking: Milestones, challenges, and new research directions. *IEEE Communications Magazine* 52, 1 (January 2014), 85–96.

Eduardo Cuervo, Aruna Balasubramanian, Dae-ki Cho, Alec Wolman, Stefan Saroiu, Ranveer Chandra, and Paramvir Bahl. 2010. MAUI: Making smartphones last longer with code offload. In *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*. ACM, 49–62.

Jeffrey Dean and Sanjay Ghemawat. 2008. MapReduce: Simplified data processing on large clusters. *ACM Communication* 51, 1 (Jan. 2008), 107–113.

Adam Dou, Vana Kalogeraki, Dimitrios Gunopulos, Taneli Mielikainen, and Ville H. Tuulos. 2010. Misco: A MapReduce framework for mobile systems. In *Proceedings of the 3rd International Conference on Pervasive Technologies Related to Assistive Environments (PETRA'10)*. ACM, New York, Article 32, 8 pages.

Charalampos Doukas, Thomas Pliakas, and Ilias Maglogiannis. 2010. Mobile healthcare information management utilizing cloud computing and android OS. In *Proceedings of 2010 Annual International Conference of the IEEE Engineering in Medicine and Biology*. IEEE, 1037–1040.

David Ehringer. 2010. The dalvik virtual machine architecture. *Technical Report* 4 (2010), 8.

Heungsik Eom, Pierre St. Juste, Renato Figueiredo, Omesh Tickoo, Ramesh Illikkal, and Ravishankar Iyer. 2013. OpenCL-based remote offloading framework for trusted mobile cloud computing. In *Proceedings of 2013 International Conference on Parallel and Distributed Systems (ICPADS)*. IEEE, Seoul, 240–248.

Huber Flores and Satish Narayana Srirama. 2014. Mobile cloud middleware. *Journal of Systems and Software* 92 (2014), 82–94.

Keke Gai, Meikang Qiu, and Hui Zhao. 2017. Cost-aware multimedia data allocation for heterogeneous memory using genetic algorithm in cloud computing. *IEEE Transactions on Cloud Computing* (2017). DOI: 10.1109/TCC.2016.2594172

Keke Gai, Meikang Qiu, Hui Zhao, Lixin Tao, and Ziliang Zong. 2016. Dynamic energy-aware cloudlet-based mobile cloud computing model for green computing. *Journal of Network and Computer Applications* 59 (2016), 46–54.

Mark S. Gordon, David Ke Hong, Peter M. Chen, Jason Flinn, Scott Mahlke, and Zhuoqing Morley Mao. 2015. Accelerating mobile applications through flip-flop replication. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys'15)*. ACM, New York, 137–150.

Mark S. Gordon, D. Anoushe Jamshidi, Scott Mahlke, Z. Morley Mao, and Xu Chen. 2012. COMET: Code offload by migrating execution transparently. In *Proceedings of the 10th USENIXc Conference on Operating Systems Design and Implementation*. USENIX, Hollywood, CA.

Mohammad Goudarzi, Mehran Zamani, and Abolfazl Toroghi Haghighat. 2017. A fast hybrid multi-site computation offloading for mobile cloud computing. *Journal of Network and Computer Applications* 80 (2017), 219–231.

Mohammad Goudarzi, Mehran Zamani, and Abolfazl Toroghi Haghighat. 2016. A genetic-based decision algorithm for multisite computation offloading in mobile cloud computing. *International Journal of Communication Systems* 30, 10 (2016). DOI: 10.1002/dac.3241

William Gropp, Ewing Lusk, and Anthony Skjellum. 1994. Using MPI: Portable parallel programming with the message-passing interface. MIT Press.

G. Guerrero-Contreras, Jose Luis Garrido, Sara Balderas-Diaz, and Carios Rodriguez-Dominguez. 2017. A context-aware architecture supporting service availability in mobile cloud computing. *IEEE Transactions on Services Computing* 10, 6 (2017), 956–968.

Qassim Bani Hani and Julius P. Dichter. 2017. Energy-efficient service-oriented architecture for mobile cloud handover. *Journal of Cloud Computing* 6, 1 (2017), 9.

Xiaoyan Hong, Mario Gerla, Guangyu Pei, and Ching-Chuan Chiang. 1999. A group mobility model for ad hoc wireless networks. In *Proceedings of the 2nd ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM'99)*. ACM, New York, 53–60.

Dijing Huang, Xinwen Zhang, Myong Kang, and Jim Luo. 2010. MobiCloud: Building secure cloud framework for mobile computing and communication. In *Proceedings of 2010 5th IEEE International Symposium on Service Oriented System Engineering (SOSE)*. IEEE, 27–34.

Karin Anna Hummel and Gerda Jelleschitz. 2007. A robust decentralized job scheduling approach for mobile peers in ad-hoc grids. In *Proceedings of the 7th IEEE International Symposium on Cluster Computing and the Grid (CCGrid'07)*. IEEE, 461–470.

Shih-Hao Hung, Tien-Tzong Tzeng, Gyun-De Wu, and Jeng-Peng Shieh. 2015. A code offloading scheme for big-data processing in android applications. *Software: Practice and Experience* 45, 8 (Aug. 2015), 1087–1101.

IBM. 2016. IBM Netezza Data Warehouse Appliances. (2016). https://www-01.ibm.com/software/data/netezza/.

Jaehoon Jeong, Hohyeon Jeong, Eunseok Lee, Tae Oh, and David Du. 2016. SAINT: Self-adaptive interactive navigation tool for cloud-based vehicular traffic optimization. *IEEE Transactions on Vehicular Technology* 65, 6 (June 2016), IEEE, 4053–4067.

John Jubin and Janet D. Tornow. 1987. The DARPA packet radio network protocols. *Proceedings of the IEEE* 75, 1 (Jan 1987), 21–32.

Warley Junior, Adriano Frana, Kelvin Dias, and Jos N. de Souza. 2017. Supporting mobility-aware computational offloading in mobile cloud environment. *Journal of Network and Computer Applications* 94 (2017), 93–108.

Yi-Hsuan Kao, Bhaskar Krishnamachari, Moo-Ryong Ra, and Fan Bai. 2017. Hermes: Latency optimal task assignment for resource-constrained mobile computing. *IEEE Transactions on Mobile Computing* 16, 11 (2017), 3056–3069.

Mahir Kaya, Altan Koyiit, and P. Erhan Eren. 2016. An adaptive mobile cloud computing framework using a call graph based model. *Journal of Network and Computer Applications* 65 (2016), 12–35.

Abdul Nasir Khan, M. L. Mat Kiah, Sajjad A. Madani, Mazhar Ali, Atta ur Rehman Khan, and Shahaboddin Shamshirband. 2014. Incremental proxy re-encryption scheme for mobile cloud computing environment. *Journal of Supercomputing* 68, 2 (2014), 624–651.

Tom Kirkham, Serge Ravet, Sandra Winfield, and Sampo Kellomki. 2011. A personal data store for an internet of subjects. In *Proceedings of 2011 International Conference on Information Society (i-Society)*. IEEE, London, UK, 92–97.

Sokol Kosta, Andrius Aucinas, Pan Hui, Richard Mortier, and Xinwen Zhang. 2012. Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading. In *Proceedings of 2012 IEEE International Conference on Computer Communications (INFOCOM)*. IEEE, 945–953.

Zeqi Lai, Y. Charlie Hu, Yong Cui, Linhui Sun, and Ningwei Dai. 2017. Furion: Engineering high-quality immersive virtual reality on today's mobile devices. In *Proceedings of the 23rd International Conference on Mobile Computing and Networking (MobiCom'17)*. ACM, Snowbird, Utah, USA.

Grace Lewis and Patricia Lago. 2015. A catalog of architectural tactics for cyber-foraging. In *Proceedings of 2015 11th International ACM SIGSOFT Conference on Quality of Software Architectures (QoSA)*. ACM, 53–62.

Bo Li, Yijian Pei, Hao Wu, and Bin Shen. 2015. Heuristics to allocate high-performance cloudlets for computation offloading in mobile ad hoc clouds. *The Journal of Supercomputing* 71, 8 (01 Aug 2015), 3009–3036.

Chunlin Li and Layuan Li. 2014. Phased scheduling for resource-constrained mobile devices in mobile cloud computing. *Wireless Personal Communications* 77, 4 (2014), 2817–2837.

Fei Li, Yogachandran Rahulamathavan, Muttukrishnan Rajarajan, and Raphael C.-W. Phan. 2013. Low complexity multi-authority attribute based encryption scheme for mobile cloud computing. In *Proceedings of 2013 IEEE 7th International Symposium on Service Oriented System Engineering (SOSE)*. IEEE, 573–577.

W. Li, Y. Zhao, S. Lu, and D. Chen. 2015. Mechanisms and challenges on mobility-augmented service provisioning for mobile cloud computing. *IEEE Communications Magazine* 53, 3 (March 2015), 89–97.

Xing Liu, Chaowei Yuan, Zhen Yang, and Zengping Zhang. 2016. Mobile-agent-based energy-efficient scheduling with dynamic channel acquisition in mobile cloud computing. *Journal of Systems Engineering and Electronics* 27, 3 (June 2016), 712–720.

Richard K. Lomotey and Ralph Deters. 2013. Reliable consumption of web services in a mobile-cloud ecosystem using REST. In *Proceedings of 2013 IEEE 7th International Symposium on Service Oriented System Engineering (SOSE)*. IEEE, 13–24.

C. Matthew MacKenzie and Ken Laskey. 2006. Reference model for service oriented architecture 1.0. (2006). https://www.oasis-open.org/committees/download.php/18486/pr2changes.pdf.

Yuyi Mao, Jun Zhang, and Khaled B. Letaief. 2016. Dynamic computation offloading for mobile-edge computing with energy harvesting devices. *IEEE Journal on Selected Areas in Communications* 34, 12 (Dec 2016), 3590–3605.

Eugene E. Marinelli. 2009. *Hyrax: Cloud Computing on Mobile Devices Using MapReduce*. Ph.D. Dissertation. Carnegie Mellon University.

Constandinos X. Mavromoustakis and others. 2015. A social-oriented mobile cloud scheme for optimal energy conservation. *Resource Management of Mobile Cloud Computing Networks and Environments* (2015), 97–121.

Peter Mell and Tim Grance. 2011. The NIST definition of cloud computing. (2011). https://csrc.nist.gov/publications/detail/sp/800-145/final.

Hiroshi Miyake and Nobuharu Kami. 2015. QoI-based data upload control for mobility-aware cloud services. In *Proceedings of 2015 3rd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering*. IEEE,16–23.

J. Paul Morrison. 2010. *Flow-based Programming: A New Approach to Application Development*. CreateSpace.

Bruce Jay Nelson. 1981. *Remote Procedure Call*. Ph.D. Dissertation. Carnegie Mellon University Pittsburgh, PA, USA.

Jim M. Ng and Yan Zhang. 2005. A mobility model with group partitioning for wireless ad hoc networks. In *Proceedings of the 3rd International Conference on Information Technology and Applications (ICITA'05)*, Vol. 2. IEEE, Sydney, NSW, Australia, 289–294.

Rajesh K. Panta, Rittwik Jana, Fan Cheng, Yih-Farn Robin Chen, and Vinay A. Vaishampayan. 2013. Phoenix: Storage using an autonomous mobile infrastructure. *IEEE Transactions on Parallel and Distributed Systems* 24, 9 (Sept 2013), 1863–1873.

JiSu Park, HeonChang Yu, KwangSik Chung, and EunYoung Lee. 2011. Markov chain based monitoring service for fault tolerance in mobile cloud computing. In *Proceedings of 2011 IEEE Workshops of International Conference on Advanced Information Networking and Applications (WAINA)*. IEEE, 520–525.

Ariel Pashtan. 2005. *Mobile Web Services*. Cambridge University Press.

Judea Pearl. 1984. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley.

Karl Pearson. 1905. The problem of the random walk. *Nature* 72, 1865 (1905), 294.

Terry Penner, Alison Johnson, Brandon Van Slyke, Mina Guirguis, and Qijun Gu. 2014. Transient clouds: Assignment and collaborative execution of tasks on mobile devices. In *Proceedings of 2014 IEEE Global Communications Conference*. IEEE, 2801–2806.

Qi Qi, Jianxin Liao, Jingyu Wang, Qi Li, and Yufei Cao. 2016. Dynamic resource orchestration for multi-task application in heterogeneous mobile cloud computing. In *Proceedings of 2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 221–226.

Yuanyuan Qiao, Yihang Cheng, Jie Yang, Jiajia Liu, and Nei Kato. 2017. A mobility analytical framework for big mobile data in densely populated area. *IEEE Transactions on Vehicular Technology* 66, 2 (Feb 2017), 1443–1455.

M. Reza Rahimi, Nalini Venkatasubramanian, and Athanasios V. Vasilakos. 2013. MuSIC: Mobility-aware optimal service allocation in mobile cloud computing. In *Proceedings of the 6th IEEE International Conference on Cloud Computing*. IEEE, 75–82.

Shima Rashidi and Saeed Sharifian. 2017. A hybrid heuristic queue based algorithm for task assignment in mobile cloud. *Future Generation Computer Systems* 68 (2017), 331–345.

Anuradha Ravi and Sateesh K. Peddoju. 2014. Handoff strategy for improving energy efficiency and cloud service availability for mobile devices. *Wireless Personal Communications* (2014), 1–32.

Stuart Robinson. 2009. Cellphone energy gap: Desperately seeking solutions. *Strategy Analytics* (2009).

Claudio Rossi, M. H. Heyi, and Francesco Scullino. 2017. A service oriented cloud-based architecture for mobile geolocated emergency services. *Concurrency and Computation: Practice and Experience* 29, 11 (2017).

Zohreh Sanaei, Saeid Abolfazli, Abdallah Gani, and Rajkumar Buyya. 2014. Heterogeneity in mobile cloud computing: Taxonomy and open challenges. *IEEE Communications Surveys Tutorials* 16, 1 (First 2014), 369–392.

Stefania Sardellitti, Gesualdo Scutari, and Sergio Barbarossa. 2015. Joint optimization of radio and computational resources for multicell mobile-edge computing. *IEEE Transactions on Signal and Information Processing over Networks* 1, 2 (June 2015), 89–103.

Mahadev Satyanarayanan. 2001. Pervasive computing: Vision and challenges. *IEEE Personal Communications* 8, 4 (Aug 2001), 10–17.

Mahadev Satyanarayanan. 2011. Mobile computing: The next decade. *ACM SIGMOBILE Mobile Computing and Communications Review* 15, 2 (Aug. 2011), 2–10.

Mahadev Satyanarayanan, Paramvir Bahl, Ramon Caceres, and Nigel Davies. 2009. The case for VM-based cloudlets in mobile computing. *IEEE Pervasive Computing* 8, 4 (Oct 2009), 14–23.

Chi-Sheng Shih, Yu-Hsin Wang, and Norman Chang. 2015. Multi-tier elastic computation framework for mobile cloud computing. In *Proceedings of 2015 3rd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*. IEEE, 223–232.

Muhammad Shiraz, Abdullah Gani, Rashid Hafeez Khokhar, and Rajkumar Buyya. 2013. A review on distributed application processing frameworks in smart mobile devices for mobile cloud computing. *IEEE Communications Surveys Tutorials* 15, 3 (Third 2013), 1294–1313.

Junaid Shuja, Abdullah Gani, Muhammad Habibur Rehman, Ejaz Ahmed, Sajjad A. Madani, Muhammad Khurram Khan, and Kwangman Ko. 2016. Towards native code offloading based MCC frameworks for multimedia applications: A survey. *Journal of Network and Computer Applications* 75 (2016), 335–354.

Tolga Soyata, Rajani Muraleedharan, Jonathan Langdon, Colin Funai, Scott Ames, Minseok Kwon, and Wendi Heinzelman. 2012. COMBAT: Mobile-cloud-based cOmpute/coMmunications infrastructure for BATtlefield applications. In *SPIE Defense, Security, and Sensing*, Eric J. Kelmelis (Ed.). International Society for Optics and Photonics.

Satish Narayana Srirama. 2017. Mobile web and cloud services enabling internet of things. *CSI Transactions on ICT* 5, 1 (01 Mar 2017), 109–117.

Satish Narayana Srirama, Matthias Jarke, and Wolfgang Prinz. 2006. Mobile web service provisioning. In *Proceedings of Advanced International Conference on Telecommunications and International Conference on Internet and Web Applications and Services (AICT-ICIW'06)*. 120–120.

Richard W. Stevens. 1990. *Unix Network Programming*. Prentice Hall PTR.

Tarik Taleb, Adlen Ksentini, and Pantelis Frangoudis. 2016. Follow-me cloud: When cloud services follow mobile users. *IEEE Transactions on Cloud Computing* (2016). DOI: 10.1109/TCC.2016.2525987

Ling Tang, Shibo He, and Qianmu Li. 2017. Double-sided bidding mechanism for resource sharing in mobile cloud. *IEEE Transactions on Vehicular Technology* 66, 2 (Feb 2017), 1798–1809.

Mati B. Terefe, Heezin Lee, Nojung Heo, Geoffrey C. Fox, and Sangyoon Oh. 2016. Energy-efficient multisite offloading policy using Markov decision process for mobile cloud computing. *Pervasive and Mobile Computing* 27 (2016), 75–89.

Chai K. Toh. 2001. *Ad Hoc Mobile Wireless Networks: Protocols and Systems*. Pearson Education.

Alessandra Toninelli, Animesh Pathak, and Valérie Issarny. 2011. *Yarta: A Middleware for Managing Mobile Social Ecosystems*. Springer Berlin, 209–220.

William Trneberg, Amardeep Mehta, Eddie Wadbro, Johan Tordsson, Johan Eker, Maria Kihl, and Erik Elmroth. 2017. Dynamic application placement in the mobile cloud network. *Future Generation Computer Systems* 70 (2017), 163–177.

Tim Verbelen, Pieter Simoens, Filip De Turck, and Bart Dhoedt. 2012. AIOLOS: Middleware for improving mobile application performance through cyber foraging. *Journal of Systems and Software* 85, 11 (2012), 2629–2639.

B. Wang, B. Li, and H. Li. 2013. Public auditing for shared data with efficient user revocation in the cloud. In *Proceedings of 2013 IEEE International Conference on Computer Communications (INFOCOM)*. IEEE, 2904–2912.

Wei Wang, Peng Xu, and Laurence Tianruo Yang. 2015. One-pass anonymous key distribution in batch for secure real-time mobile services. In *Proceedings of 2015 IEEE International Conference on Mobile Services*. IEEE, 158–165.

Xiaoliang Wang, Wenyao Xu, and Zhanpeng Jin. 2017. A hidden Markov model based dynamic scheduling approach for mobile cloud telemonitoring. In *Proceedings of 2017 IEEE EMBS International Conference on Biomedical Health Informatics (BHI)*. IEEE, 273–276.

Zhi Wang, Rui-Chun Hou, and Zhi-Ming Zhou. 2016. An Android/OSGi-based mobile gateway for body sensor network. In *Proceedings of 2016 15th International Symposium on Parallel and Distributed Computing (ISPDC)*. IEEE, Fuzhou, China, 135–140.

Xianglin Wei, Jianhua Fan, Tongxiang Wang, and Qiping Wang. 2016. Efficient application scheduling in mobile cloud computing based on MAX–MIN ant system. *Soft Computing* 20, 7 (2016), 2611–2625.

Liyao Xiang, Shiwen Ye, Yuan Feng, Baochun Li, and Bo Li. 2014. Ready, set, go: Coalesced offloading from mobile devices to the cloud. In *Proceedings of 2014 IEEE Conference on Computer Communications*. IEEE, 2373–2381.

Zichuan Xu, Weifa Liang, Wenzheng Xu, Mike Jia, and Song Guo. 2016. Efficient algorithms for capacitated cloudlet placements. *IEEE Transactions on Parallel and Distributed Systems* 27, 10 (Oct 2016), 2866–2880.

Lei Yang, Jiannong Cao, Hui Cheng, and Yusheng Ji. 2015. Multi-user computation partitioning for latency sensitive mobile cloud applications. *IEEE Transactions on Computing* 64, 8 (Aug 2015), 2253–2266.

Lei Yang, Jiannong Cao, Guanqing Liang, and Xu Han. 2016. Cost-aware service placement and load dispatching in mobile cloud systems. *IEEE Transactions on Computing* 65, 5 (May 2016), 1440–1452.

Sen Yang, Xiangshun Bei, Yongbing Zhang, and Yusheng Ji. 2016. Application offloading based on R-OSGi in mobile cloud computing. In *Proceedings of 2016 4th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*. IEEE, 46–52.

Jiawei Yuan and Shucheng Yu. 2014. Efficient public integrity checking for cloud data sharing with multi-user modification. In *Proceedings of 2014 IEEE Conference on Computer Communications*. IEEE, 2121–2129.

Xiao Zeng, Kai Cao, and Mi Zhang. 2017. MobileDeepPill: A small-footprint mobile deep learning system for recognizing unconstrained pill images. In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys'17)*. ACM, New York, 56–67.

Weimin Zheng, Pengzhi Xu, Xiaomeng Huang, and Nuo Wu. 2010. Design a cloud storage platform for pervasive computing environments. *Cluster Computing* 13, 2 (2010), 141–151.

Bowen Zhou, Amir Vahid Dastjerdi, Rodrigo Calheiros, Satish Srirama, and Rajkumar Buyya. 2015b. mCloud: A context-aware offloading framework for heterogeneous mobile cloud. *IEEE Transactions on Services Computing* (2015). DOI: http://dx.doi.org/10.1109/TSC.2015.2511002

Bowen Zhou, Amir Vahid Dastjerdi, Rodrigo N. Calheiros, Satish Narayana Srirama, and Rajkumar Buyya. 2015a. A context sensitive offloading scheme for mobile cloud computing service. In *Proceedings of 2015 IEEE 8th International Conference on Cloud Computing*. IEEE, 869–876.

Liang Zhou, Zhen Yang, Joel JPC Rodrigues, and Mohsen Guizani. 2013. Exploring blind online scheduling for mobile cloud multimedia services. *IEEE Wireless Communications* 20, 3 (2013), 54–61.

Zhibin Zhou and Dijiang Huang. 2013. Efficient and secure cata storage operations for mobile cloud computing. In *Proceedings of the 8th International Conference on Network and Service Management (CNSM'12)*. International Federation for Information Processing, Laxenburg, Austria, Austria, 37–45.

Dimitrios Zissis and Dimitrios Lekkas. 2012. Addressing cloud computing security issues. *Future Generation Computer Systems* 28, 3 (2012), 583–592.

Received January 2017; revised October 2017; accepted October 2017