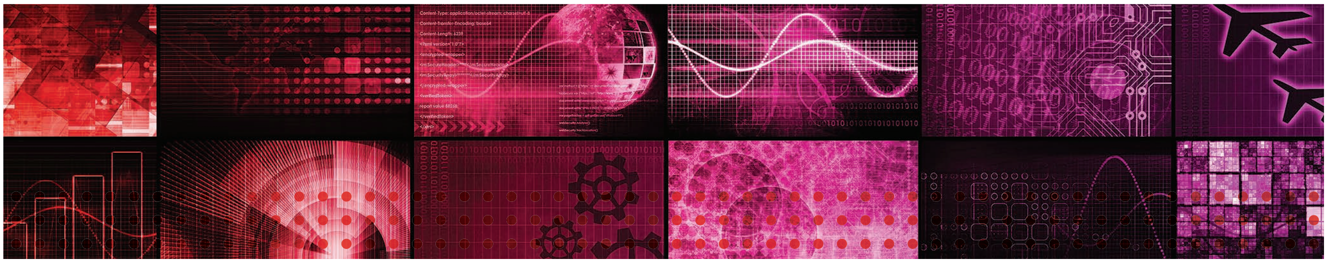

Context-Aware Mobile Cloud Computing and Its Challenges

Atta ur Rehman Khan, COMSATS Institute of Information Technology, Pakistan
Mazliza Othman, University of Malaya
Feng Xia, Dalian University of Technology, China
Abdul Nasir Khan, COMSATS Institute of Information Technology, Pakistan

Context-aware application development models enable effective computation offloading for enhanced performance, energy efficiency, and execution support on mobile devices.

Mobile cloud computing evolved from cloud computing to address the needs of the ever-increasing number of smartphone users and inherent smartphone constraints, such as limited computational power, memory, storage, and energy. Because mobile cloud computing is a comparatively new domain, it has no standard definition and different researchers provide varying definitions. For example,

> Mobile cloud computing is an integration of cloud computing technology with mobile devices to make the mobile devices resource-full in terms of computational power, memory, storage, energy, and context awareness.[1]

In mobile cloud computing, "cloud" can refer to both real and virtual clouds. Real cloud refers to the traditional cloud infrastructure that provides virtually unlimited resources, such as Amazon Elastic Compute Cloud (EC2), Microsoft Azure, and Google App Engine. Real cloud service models include software as a service (SaaS), platform as a service (PaaS), and infrastructure as a service (IaaS). Real cloud also includes multiple cloud deployment models, such as private, community, public, and hybrid. Virtual cloud, on the other hand, refers to a nearby infrastructure, such as servers and personal computers, providing services to the mobile devices.[1]

Mobile cloud computing uses two types of architectures. In an *infrastructure-based* system, the cloud hardware infrastructure remains stationary and provides services to mobile users, via Wi-Fi or cellular-network-based Internet connections. In an *ad hoc*

2325-6095/15/$31.00 © 2015 IEEE

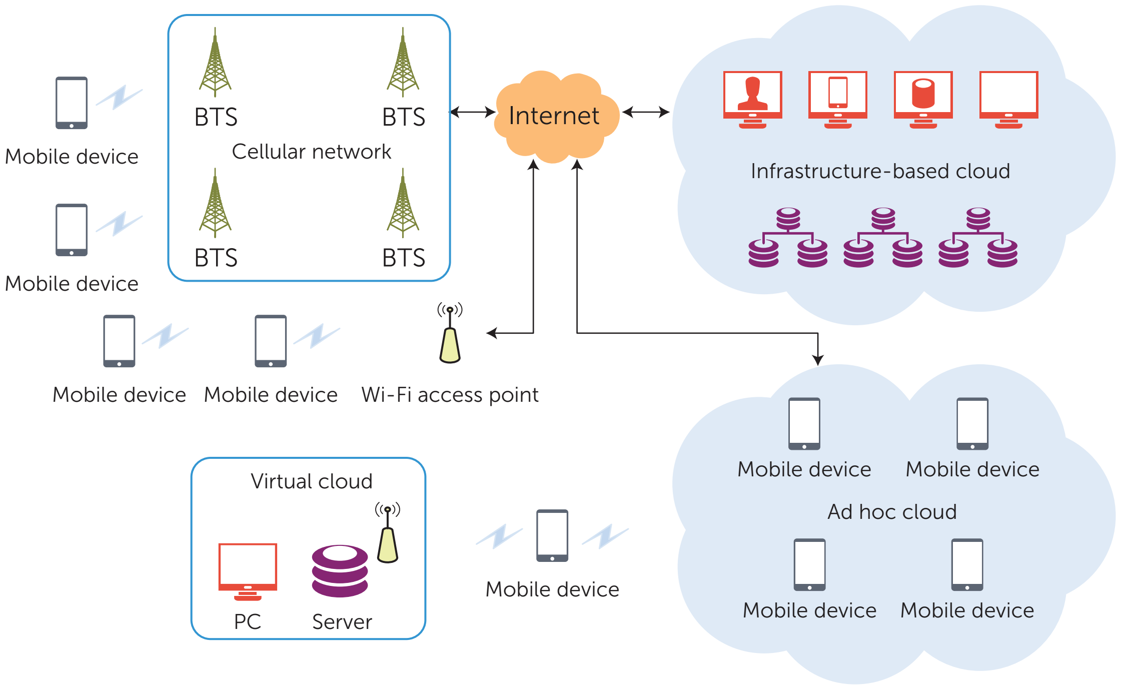

FIGURE 1. A mobile cloud architecture in which mobile devices offload computations to an infrastructure-based cloud, virtual cloud, and ad hoc cloud via cellular or Wi-Fi communication link (BTS: base transceiver station).

system, multiple mobile devices form a group that acts as a cloud and offers services to other mobile devices.[2] These cloud services can be virtual (group based) or real, with requests sent to the real cloud (see Figure 1). Given space limitations, we restrict our discussion to infrastructure-based architectures.

Whereas the primary objective of cloud computing is to provide IT resources to businesses in a cost-effective manner, mobile cloud computing focuses on overcoming smartphone constraints and enhancing mobile users' experience. Recent research has identified three main benefits of mobile cloud technology: it enhances smartphone applications' performance by utilizing the computational power of the resource-rich cloud, makes smartphone applications energy efficient by reducing computational overhead on the devices using computation offloading, and enables smartphones to execute resource-intensive applications that are unsupported in a resource-constrained environment.[1]

Because the two technologies' objectives differ, so do their challenges. For instance, in mobile cloud computing, a mobile device's limited energy is an issue, whereas in cloud computing, the supply of energy is unlimited. Similarly, mobility is an important parameter in mobile cloud computing but less important in cloud computing. Security, however, is equally important in both technologies.[3,4]

Mobile cloud computing uses *computation offloading* to migrate resource-intensive computational tasks from a smartphone to the cloud. Computation offloading is missing in traditional mobile application development models, because smartphone applications are designed to execute on the smartphone only. Therefore, mobile cloud computing requires specialized mobile cloud application development models that support computation offloading and execution of smartphone applications in two environments—smartphone and cloud[5] (see the "Mobile Cloud Application Development Models" sidebar for further discussion). Mobile cloud application models offload computational tasks to the cloud through a process, component (module), application, or smartphone clone image[6] that resides in the cloud and facilitates executing the smartphone's computational requests. The application models need to be context aware because computation offloading isn't always beneficial and can cause performance degradation or energy wastage. This article highlights context-awareness aspects of mobile cloud application development models, and presents various challenges to achieving the required context awareness.

Context-Aware Application Models

There are two perspectives on context awareness in mobile cloud application models: context-aware

MOBILE CLOUD APPLICATION DEVELOPMENT MODELS

Most mobile cloud applications are developed for Android, Windows Mobile, iOS, and BlackBerry platforms using technologies such as Hadoop, HTML5, Internet Suspend/Resume, R-OSGi, Stack-On-Demand (SOD), and Representational State-Transfer (REST). The applications are either powered by real cloud instances (Amazon Elastic Compute Cloud, Microsoft Azure, or Google App Engine) or virtual cloud service/virtual machine (VM) instances powered by VMWare Workstation or Oracle VM VirtualBox.

Currently, mobile cloud applications are tested in a real environment because there's no specialized simulator for this technology. Among other areas, mobile cloud computing research groups and active projects are exploring mobile cloud service models, for example,

- the Mobile Multimedia Cloud Computing Project at RWTH Aachen University (http://dbis.rwth-aachen.de/cms/projects/i5cloud);
- mobile cloud middleware and application migration, such as the Reuse and Migration of Legacy Applications to Interoperable Cloud Services (Remics) project at the University of Tartu (http://mc.cs.ut.ee/mcsite/projects);
- mobile cloud-based context-aware applications for smart cities, healthcare, and productivity, such as the Mobile Cloud Computing Group at University College Cork, Ireland (www.ucc.ie/en/mccg/projects); and
- mobile cloud networks, services, and architectures, such as the MONICA project (http://cordis.europa.eu/project/rcn/101690_en.html) at the Community Research and Development Information Service (CORDIS).

Mobile cloud applications include mathematical tools, file indexing, image processing tools, games, download tools, antivirus tools, rendering and streaming, context-aware health monitoring, smart cities, and big data analysis. Unfortunately, these applications are implemented as a proof of concept (for testing purposes only). Applications currently available in the market have limited support of cloud computing, where the service is hosted solely in the cloud with no support of code/application migration. Mobile cloud applications are expected to be launched in the market soon. Table A summarizes recent well-known application models, which are discussed in detail elsewhere.[1–9]

References

1. A.R. Khan et al., "A Survey of Mobile Cloud Computing Application Models," *IEEE Comm. Surveys & Tutorials*, vol. 16, no. 1, 2014, pp. 393–413.
2. B.-G. Chun et al., "CloneCloud: Elastic Execution between Mobile Device and Cloud," *Proc. Int'l Conf. Computer Systems*, 2011, pp. 301–314.
3. X. Zhang et al., "Towards an Elastic Application Model for Augmenting Computing Capabilities of Mobile Platforms," *Mobile Wireless Middleware, Op-*

application partitioning and context-aware computation offloading.

Context-Aware Application Partitioning

Offloading an entire application to the cloud generally isn't a good solution, because the application might require smartphone hardware support, such as GPS and sensors, that isn't available in the cloud. Moreover, offloading an application can require a high amount of communication that might increase the offloading delay and energy consumption. To overcome this issue, researchers recommend offloading only resource-intensive parts of an application to the cloud, which can enhance performance, increase energy efficiency, or support execution.

To enable this selective offloading, a smartphone application is partitioned into components to be distributed between the smartphone and cloud for execution. The partitioning can be static (predefined during development) or dynamic (based on runtime conditions).[7] However, to gain the benefits of mobile

Table A. Current mobile cloud application development models.

Applications model	Description
CloneCloud[2]	Offloads resource-intensive processes from a mobile device to a mobile device clone that's maintained on the nearby infrastructure (personal computers or servers) or cloud.
Xinwen Zhang and colleagues' model[3]	Partitions an application into multiple components (weblets), which are executed locally or offloaded to the cloud, depending on the available local resources and user preference.
μCloud[4]	Focuses application development using heterogeneous components (presented as a directed graph) that can execute on a smartphone, cloud, or both.
Mahadev Satyanarayanan and colleagues' model[5]	Uses an augmented execution technique in which smartphone computations are offloaded to a VM on a resource-rich computer or group of computers (a cloudlet).
Ioana Giurgiu and colleagues' model[6]	Distributes functional layers among the smartphone and cloud/nearby infrastructure, and deploys an application's functional components (bundles) dynamically, based on optimal deployment analysis.
eXCloud[7]	Bases computation offloading on VM-instance level and performs on-demand code/data migration to the cloud.
MAUI[8]	Uses dynamic application partitioning and supports method-level offloading to the cloud/nearby infrastructure.
ThinkAir[9]	Uses an energy model to make context-aware offloading decisions and supports method-level offloading to a smartphone clone in the cloud.

erating Systems, and Applications, Springer, 2010, pp. 161–174.

4. V. March et al., "μCloud: Towards a New Paradigm of Rich Mobile Applications," *Procedia Computer Science*, 2011, vol. 5, pp. 618–624
5. M. Satyanarayanan et al., "The Case for VM-Based Cloudlets in Mobile Computing," *IEEE Pervasive Computing*, vol. 8, no. 4, 2009, pp. 14–23.
6. I. Giurgiu et al., "Calling the Cloud: Enabling Mobile Phones as Interfaces to Cloud Applications," *Proc. ACM/IFIP/USENIX 10th Int'l Conf. Middleware* (Middleware 09), 2009, pp. 83–102.
7. R.K. Ma, K.T. Lam, and C.-L. Wang, "eXCloud: Transparent Runtime Support for Scaling Mobile Applications in Cloud," *Proc. Int'l Conf. Cloud and Service Computing* (CSC), 2011, pp. 103–110.
8. E. Cuervo et al., "MAUI: Making Smartphones Last Longer with Code Offload," *Proc. Int'l Conf. Mobile Systems, Applications, and Services*, 2010, pp. 49–62.
9. S. Kosta et al., "ThinkAir: Dynamic Resource Allocation and Parallel Execution on the Cloud for Mobile Code Offloading," *Proc. IEEE INFOCOM*, 2012, pp. 945–953.

cloud computing, applications are partitioned in a context-aware fashion that considers user interaction frequency, local resource access (for example, GPS, input module, or sensors), resource requirements, computational intensity, and bandwidth consumption.[1]

Context-Aware Computation Offloading

In context-aware computation offloading, decisions are made in favor of performance enhancement, energy efficiency, and application execution. Context awareness is necessary because computation offloading isn't always beneficial. To prove this, we developed three Android-based applications:

- app-1 finds the sum of 0.1 million numbers,
- app-2 cracks a five-character password, and
- app-3 multiplies a 750 × 750 matrix.

All applications were capable of offloading computations to the Google App Engine (F1 instance) using HTTP requests via Wi-Fi and 3G connections. The applications were executed on a Sony Xperia S

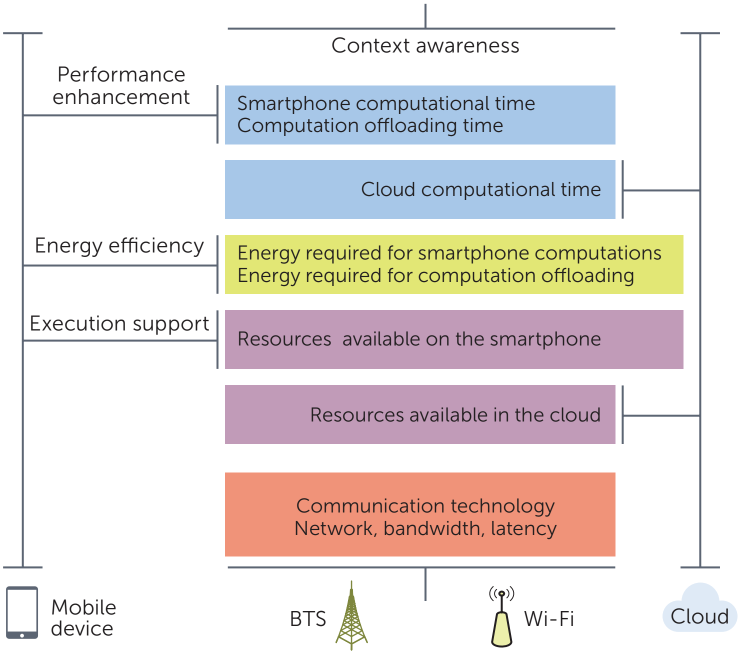
Context awareness
Performance enhancement
Smartphone computational time
Computation offloading time
Cloud computational time
Energy efficiency
Energy required for smartphone computations
Energy required for computation offloading
Execution support
Resources available on the smartphone
Resources available in the cloud
Communication technology
Network, bandwidth, latency
Mobile device
BTS
Wi-Fi
Cloud

FIGURE 2. Entities involved in context-aware computation offloading. These entities include smartphone resources, communication technology (Wi-Fi or cellular), network bandwidth, data size and location, available cloud resources, and application model/computation offloading technique (BTS: base transceiver station).

smartphone to achieve performance and energy efficiency. All the applications had different computational complexities and required a variable amount of data.

When app-1 was executed, it took more time in terms of communications (offloading) and overconsumed the smartphone energy than a local execution, proving computation offloading to be unfavorable in terms of energy efficiency and performance enhancement. When app-2 and app-3 were executed, the applications took less time and consumed less energy than a local execution. Consequently, for app-2 and app-3, the computation offloading is favorable in terms of energy efficiency and performance enhancement. However, the performance and energy gain ratio of app-2 and app-3 vary because of the computational complexity and the amount of data offloaded to the cloud. Hence, to achieve the required level of performance or energy efficiency, the offloading decisions must be context aware or offloading might not provide any benefit.

Context-aware offloading decisions involve various entities, as Figure 2 illustrates. The important context-awareness aspects of mobile cloud computing are objective awareness, performance awareness, energy awareness, and resource awareness.

Objective awareness. Because the three main features of mobile cloud computing—performance enhancement, energy efficiency, and execution support—are interlinked, achieving one objective could adversely affect the others. For instance, in some cases, computation offloading is favorable for application execution but unfavorable in terms of energy efficiency or performance enhancement. This phenomenon depends mainly on the nature of the application and its model type. Consequently, many models focus on a particular feature.[1] However, a few models support multiple features (performance enhancement, energy efficiency, and execution support) at the same time, where the enhancement ratio of performance to energy efficiency is set dynamically based on the runtime conditions or according to the user's preferences.

Objective awareness is important for computation offloading because it not only helps to prioritize a user's preference for mobile cloud computing features but also guides an application model to partition an application accordingly.

Performance awareness. In general, computation offloading is favorable in terms of performance enhancement when an application's computational time on a smartphone is high compared to the sum of computation offloading time and cloud computational time. Therefore, to make favorable offloading decisions, the application models need to be aware of the task's local (smartphone-based) and cloud-based computational time. This information is provided by the profilers, which are responsible for monitoring the local and cloud-based executions of a computational task. Alternatively, in some scenarios, the offloading decisions are based on the difference between the available resources of the smartphone and cloud.

Energy awareness. The execution of a resource-intensive computational task on a smartphone consumes a considerable amount of energy. Likewise, offloading a computational task to the cloud consumes a smartphone's energy in terms of computational request preparation, communications for offloading, and result integration. Computation offloading is favorable for energy efficiency when the energy required for smartphone-based computations is high compared to the energy required for computation offloading.

Therefore, favorable offloading decisions in this context depend on the energy awareness of the application models (that is, the energy required for local execution versus computation offloading). The required energy information is estimated by the smartphone energy consumption models.[8] For computation offloading to be energy efficient, the decision is made based on an application's energy profile and an estimate of the amount of energy required for communications per size of data.

Resource awareness. Computation offloading for application execution is performed to execute an application in a resource-constrained environment. This occurs when the smartphone resources are insufficient for execution or the available resources are overloaded. Supporting computation offloading in such scenarios requires resource awareness about the smartphone and cloud, particularly when offloading to a virtual cloud.

For example, consider a scenario where a quad-core smartphone offloads to a virtual cloud with limited resources that are shared among multiple users. To make an optimum offloading decision for the application execution, the application models use information about available resources at the smartphone and cloud.

Challenges in Context-Aware Mobile Cloud Computing

Mobile cloud computing faces many challenges in performing context-aware computation offloading. Here, we identify and discuss six of the most important challenges that hamper mobile cloud application models from making context-aware offloading decisions.

Application Partitioning

As discussed in the previous sections, application partitioning is critical for computation offloading. However, identifying resource-intensive components is a challenge because there's no hard rule for defining a component's intensity. For instance, an application component might be resource intensive (in terms of computational time) for a low processing power smartphone but not for a high processing power one.

Even if intensity is defined in terms of computational complexity, application partitioning is still an issue. For instance, static partitioning and decisions regarding the components' execution location is not a foolproof solution and might fail in a number of scenarios.[1] Even though dynamic context-aware partitioning has an edge over static partitioning, it requires timely repartitioning of applications to accommodate changes caused by the mobile environment and inconsistently available resources (on a virtual cloud).

Computational Time

A task's computational time varies based on available smartphone/cloud resources, the task's nature, and the input data's size. Therefore, to enhance performance using mobile cloud computing, the application must be aware of the task's computational time on the smartphone and cloud. The challenge in doing this is that smartphones have different hardware specifications and architectures (single core, dual core, and quad core). Therefore, there's no predefined computational time for any application or its components.

We can estimate an application's worst-case execution time, which is a correct solution to some extent. However, as discussed previously, the applications can be partitioned dynamically based on the runtime condition, which can further change depending on its current environment. Therefore, estimating the worst-case execution time for every possible partitioning pattern isn't an optimal solution because it might incur a large computational overhead on the smartphone.

Moreover, the computational time of an application (or its components) in the cloud can vary based on available resources. For instance, in a virtual cloud environment, a single server/personal computer is shared among multiple users, and its load varies from time to time, which ultimately affects execution time. Furthermore, data input size and type of code instructions (integer or floating point) are important factors that can affect a task's computational time.

Given these factors, estimating the computational time of an application or its components is a complex problem. Although this can be partially resolved by profiling smartphone and cloud-based executions,[9] the overhead of profiling every execution of a component against variable size data input is worth consideration.

Computational Energy

To make smartphone applications energy efficient through mobile cloud computing, the applications must be aware of the amount of energy required for smartphone-based application execution and computation offloading. Otherwise, incorrect offloading decisions can overconsume smartphones' energy because of a high amount of communication between the smartphone and the cloud about offloading.

The challenge is that the amount of energy consumed by applications depends on the smartphone model (hardware type and specifications). For example, two smartphones with different types of processors (single core versus quad core) will consume different amounts of energy. Moreover, an application's energy consumption can vary depending on the CPU frequency and utilization level.

The problem is compounded by the fact that smartphones don't provide low-level energy information for computations and communications. For instance, in Android OS, developers can only access information about smartphone battery level (total battery remaining) and the percentage of smartphone energy that's consumed by a particular application (see http://developer.android.com/training/monitoring-device-state/battery-monitoring.html), which is insufficient for making energy-aware offloading decisions.

As proof, we can execute a resource-intensive application on a smartphone for 1 minute and compare the battery level readings before and after the execution. Most of the time, there's no change in the readings. Consequently, monitoring energy consumption of multiple components of an application in terms of computations becomes a challenging task.

As mentioned earlier, the application models use energy-consumption models to overcome this issue. However, the energy models are developed by executing predefined computational and communicational tasks on a smartphone, and the energy consumption is measured using external hardware (power meter). Further, energy consumption coefficients are defined based on the monitored readings. Consequently, the energy models are valid only for the monitored smartphone and might provide inaccurate readings on unknown (new model) smartphones.[8]

Offloading Time

Some people might compare smartphone and cloud computational times to check if the computation offloading enhances performance. However, this comparison doesn't guarantee the required performance gain, because computation offloading takes a considerable amount of time in terms of communication between the smartphone and the cloud. Apart from this, in some application models, computation offloading incurs considerable delay on the smartphone (request preparation time and result integration time) and cloud (request marshalling time and result preparation time). Therefore, context-aware application models must utilize most of the highlighted parameters to make optimum offloading decisions.

The challenge in doing so is that the aforementioned delays on the smartphone and cloud can vary with time.[10] Moreover, the communication link quality of mobile networks is never consistent, and offloading time varies depending on signal strength, network bandwidth, latency, mobility, cell size (in cellular networks), and network load. Therefore, communication link quality estimation and prediction of associated delays are challenging issues.

To address these challenges, some researchers use profilers to monitor the required information, which is later used in decision making, whereas other researchers use runtime link monitoring by sending a small amount of data to the cloud to estimate the communication time. Alternatively, some researchers use the most recent communication history information to estimate the communication time. However, these techniques incur computational and communicational overhead on the smartphone and might fail because of mobile network fluctuating characteristics.

Offloading Energy

As with offloading time, estimating the amount of energy required for computational offloading is a challenging task, because the offloading process incurs a variable computational overhead on the smartphone (as discussed previously). Moreover, energy consumption varies based on the communication technology, bandwidth, transmission power/radio state, amount of data, cell size (in cellular networks), and most importantly, communication pattern (packet size and the interval between packet sending).

We estimate offloading energy using techniques similar to those discussed in the previous section. It has similar pros and cons.

Application Support

Smartphones support a wide range of applications, each with variable characteristics and resource demands. For instance, a mathematical tool might take small data input and perform large computations, whereas an antivirus application might require large data input to perform large computations. Consequently, an application model that can improve performance or energy efficiency by using runtime data offloading might do well for one type of application and fail for others. This variability exists because existing application models use a single offloading technique to achieve a particular objective for a predefined application type.

Therefore, new application models are required that can support different types of applications by using multiple offloading techniques in a single model. However, making an application model intelligent enough to characterize the applications properly and apply optimal offloading technique is challenging.

In light of current developments and challenges in context-aware mobile cloud computing, we propose several actions. First, benchmarking the available smartphone processors against desktop system processors and common cloud instances will highlight the differences between their computational powers and expected performance enhancement (irrespective of the communication time). In addition, smartphone vendors must release energy information datasheets for their smartphones so that precise energy consumption coefficients of different computational and communicational entities can be known. Smartphone operating systems must also evolve in terms of information provision so that detailed energy consumption information of a particular task in terms of computations and communications is available to both applications and developers. Finally, mobile cloud computing service models must be standardized so the application models can use common offloading techniques.

Cloud storage and application-specific services, such as Apples' iCloud and Siri, already appear on smartphones. Before long, cloud-based computing applications for smartphones will appear in the market that will enhance mobile users' experience in terms of performance, energy efficiency, and execution support. However, this demands context awareness in multiple aspects. Efforts are being made to achieve the desired level of context awareness, but existing solutions aren't up to the mark. Therefore, more efforts are required to make this technology grow. Cloud computing might not be on the horizon, but its powered technologies, such as mobile cloud computing, are still emerging, and with the wide range of open issues, this technology won't be going anywhere soon. ●●●

References

1. A.R. Khan et al., "A Survey of Mobile Cloud Computing Application Models," *IEEE Comm. Surveys & Tutorials*, vol. 16, no. 1, 2014, pp. 393–413.
2. T. Xing et al., "MobiCloud: A Geo-Distributed Mobile Cloud Computing Platform," *Proc. Int'l Conf. Network and Service Management* (CNSM 12), 2012, pp. 164–168.
3. R. Lacuesta et al., "Spontaneous Ad Hoc Mobile Cloud Computing Network," *Scientific World J.*, vol. 2014, 2014, article 232419; doi: 10.1155/2014/232419.
4. H. Modares et al., "Security in Mobile Cloud Computing," *Mobile Networks and Cloud Computing Convergence for Progressive Services and Applications*, J.J.P.C. Rodrigues and J. Lloret, eds., 2013, pp. 79–91.
5. A.R. Khan et al., "Pirax: Framework for Application Piracy Control in Mobile Cloud Environment," *J. Super Computing*, vol. 68, no. 2, 2014, pp. 753–776.
6. M. Satyanarayanan et al., "The Case for VM-Based Cloudlets in Mobile Computing," *IEEE Pervasive Computing*, vol. 8, no. 4, 2009, pp. 14–23.
7. J. Niu et al., "Bandwidth-Adaptive Application Partitioning for Execution Time and Energy Optimization," *Proc. IEEE Int'l Conf. Communications* (ICC 13), 2013, pp. 3660–3665.
8. L. Zhang et al., "Accurate Online Power Estimation and Automatic Battery Behavior Based Power Model Generation for Smartphones," *Proc. Int'l Conf. Hardware/Software Codesign and System Synthesis*, 2010, pp. 105–114.
9. A.R. Khan et al., "MobiByte: An Application Development Model for Mobile Cloud Computing," *J. Grid Computing*, Apr. 2015, pp. 1–24; doi: 10.1007/s10723-015-9335-x.
10. B.-G. Chun et al., "CloneCloud: Elastic Execution between Mobile Device and Cloud," *Proc. Int'l Conf. Computer Systems*, 2011, pp. 301–314.

ATTA UR REHMAN KHAN *is an assistant professor in the Department of Computer Science at COMSATS Institute of Information Technology (CIIT), Pakistan, and a freelance ICT consultant. His research interests include mobile computing, cloud computing, ad hoc networks, distributed systems, and security. Khan has a PhD in mobile cloud computing from the University of Malaya. Contact him at dr@attaurrehman.com.*

MAZLIZA OTHMAN *is an associate professor with the Faculty of Computer Science and IT at the University of Malaya. Her research interests include pervasive computing and self-organizing networks. Othman has a PhD in mobile computing from the University of London. She is the author of* Principles of Mobile Computing and Communications *(Auerbach Publications, 2007). Contact her at mazliza@um.edu.my.*

FENG XIA *is a full professor in the School of Software, Dalian University of Technology, China. His research interests include social computing, mobile computing, and cyber-physical systems. Xia has a PhD in control science and engineering from Zhejiang University, Hangzhou, China. He's a senior member of IEEE, IEEE Computer Society, IEEE SMC Society, ACM, and ACM SIGWEB. Xia is the corresponding author. Contact him at f.xia@ieee.org.*

ABDUL NASIR KHAN *is an assistant professor in the Department of Computer Science, COMSATS Institute of Information Technology. His research interests include various aspects of network security and distributed computing. Khan has a PhD in mobile cloud security from the University of Malaya. Contact him at anasir@ciit.net.pk.*