Seamless Application Execution in Mobile Cloud Computing: Motivation, Taxonomy, and Open Challenges

Ejaz Ahmed[a], Abdullah Gani[a], Muhammad Khurram Khan[b], Rajkumar Buyya[c], Samee U. Khan[d]

[a] *Centre for Mobile Cloud Computing Research (C4MCCR), Faculty of Computer Science and Information Technology, University of Malaya, Kuala Lumpur, Malaysia*
[b] *Center of Excellence in Information Assurance, King Saud University, Riyadh, Saudi Arabia*
[c] *Cloud Computing and Distributed Systems (CLOUDS) Lab, Department of Computing and Information Systems, The University of Melbourne, Australia*
[d] *Department of Electrical and Computer Engineering, North Dakota State University, USA*
Email: {ejazahmed, abdullahgani}@ieee.org, mkhurram@ksu.edu.sa, raj@csse.unimelb.edu.au, samee.khan@ndsu.edu

Abstract

Seamless application execution is vital for the usability of various delay-sensitive mobile cloud applications. However, the resource-intensive migration process and intrinsic limitations of the wireless medium impede the realization of seamless execution in mobile cloud computing (MCC) environment. This work is the first comprehensive survey that studies the state-of-the-art cloud-based mobile application execution frameworks (CMAEFs) in perspective of seamless application execution in MCC and investigates the frameworks suitability for the seamless execution. The seamless execution enabling approaches for the CMAEFs are identified and classified based on the implementation locations. We also investigate the seamless application execution enabling approaches to identify advantages and disadvantages of employing such approaches for attaining the seamless application execution in MCC. The existing frameworks are compared based on the significant parameters derived from the taxonomy of the seamless application execution enabling approaches. The principles for enabling the seamless application execution within the MCC are also highlighted. Finally, open research challenges in realizing the seamless application execution are discussed.

Keywords: Mobile Cloud Computing, Seamless Application Execution, Cloud-Based Mobile Application Execution Frameworks

1. Introduction

The incomparable advantages of mobile cloud computing (MCC) and a diverse range of potential mobile cloud applications are provoking mobile users to leverage on the benefits of emerging mobile applications, such as m-learning [1], m-gaming [2], and m-health [3]. In the MCC, computational migration has introduced as a software level solution of utilizing remote cloud-based resources for

augmenting application processing capabilities of the smart mobile devices. To execute the mobile applications, application execution frameworks usually migrate either part of an application [4, 5] or the entire application to the cloud [6, 7]. On the successful execution, the results are sent back to the mobile device for integration with the rest of the application inside the mobile device.

The run-time migration of an application to the cloud improves the application responsiveness and energy consumption [8]. However, the application migration process in the MCC usually involves human interaction; thereby, impeding the smooth execution of the mobile applications. The intrinsic limitations of the mobile devices and wireless access technologies are other common factors that originate disruption in the execution of the mobile applications and inhibit seamless execution of the application in MCC [9]. The seamless application execution refers the state of unobtrusive application execution with the least possible user involvement, interaction, and distraction aiming to deliver enhanced functionality, higher performance, and improved responsiveness towards a richer user experience. The seamless execution is imperative to meet the application requirements for mobile applications, particularly for delay sensitive applications, such as m-health [3] and augmented reality [10]. However, the realization of seamless application execution in MCC is non-trivial due to several issues, such as dynamic MCC execution environments, complex multi-objective offloading decision functions, incompatible heterogeneous wireless technologies [11–13], intrinsic limitations in the wireless medium, intensive authentication, authorization, and accounting processes.

Table 1: List of Acronyms and Corresponding Full Form

Symbol	**Description**
CMAEF	Cloud-based Mobile Application Execution Framework
CPU	Central Processing Unit
I/O	Input/ Output
MAUI	Mobile Assistance Using Infrastructure
MCC	Mobile Cloud Computing
QoE	Quality of Experience
QoS	Quality of Service
SLA	Service Level Agreement
TSP	Telecommunication Service Provider
VM	Virtual Machine
WAN	Wide Area Networks
WiFi	Wireless Fidelity
WLAN	Wireless Local Area Network
3G	Third Generation

Although several surveys [14–18] have studied various aspects of leveraging the cloud services to augment the computing capabilities of resource-constrained mobile devices, seamless execution

of mobile applications within the MCC, has not been surveyed. Contrary to our prior efforts, this study is the first effort that comprehensively surveys the seamless application execution within the MCC. It describes the seamless application execution in the MCC and comprehensively surveys the state-of-the-art cloud-based mobile application execution frameworks (CMAEFs) to investigate the suitability of the frameworks. Moreover, we identify and describe the important principles and open challenges in realizing the vision of the seamless application execution.

The contributions of the survey are manifolds. Firstly, we comprehensively survey the state-of-the-art CMAEFs to evaluate their features implemented to achieve seamlessness in execution of the mobile applications. Secondly, we classify the seamless application execution enabling approaches employed by the current frameworks to devise a taxonomy. Thirdly, we analyze and synthesize seamless application execution enabling approaches to identify advantages and disadvantages. Fourthly, we compare the state-of-the-art CMAEFs by using the thematic taxonomy of seamless application execution enabling approaches. Fifthly, we identify the principles of designing seamless CMAEFs in the MCC. Finally, we detail open challenges in realizing the seamless application execution in the MCC. This article also enables the mobile cloud application engineers and cloud service providers to leverage on the appropriate features that can mitigate communication and computation latencies when developing applications or providing MCC services to increase the QoS for mobile-cloud users. The identified principles guide the framework designers to incorporate the specific features to efficiently realize the seamless application execution. Similarly, the identified open research challenges highlight the future research directions. For the ease of reading, we list all of the commonly used acronyms in Table 1.

The remainder of this paper is organized as follows. Section 2 introduces the MCC technology and explains the seamless application execution definition. Section 3 discusses the significance and motivation of attaining the seamless application execution in MCC. The research methodology is provided in Section 4 for the readers to give them information that how we have collected research articles for the survey and what is our selection criteria. Section 5 presents the comprehensive survey of the state-of-the-art CMAEFs and enquires the logical implications and critical aspects of the frameworks with respect to seamless application execution in the MCC. The taxonomy of seamless application execution enabling approaches is presented and described in Section 6. This section also discusses the advantages and disadvantages of seamless application execution enabling approaches along with the comparison of CMAEFs based on the taxonomy. Section 7 presents the identified principles for designing seamless CMAEFs in the MCC. Section 8 discusses open research challenges in realizing the vision of seamless application execution in the MCC. Finally, Section 9 draws the conclusions and identifies the directions for future research.

2. Background

This Section provides a brief background on the MCC including definition and motivation behind realizing the seamless application execution in the MCC.

2.1. Mobile Cloud Computing

MCC is an emerging distributed computing paradigm that aims to augment the resources of mobile devices by leveraging the resources and services of remote cloud. The MCC synergistically integrates the mobile computing, the cloud computing, and the wireless technologies to unleash the potentials of these technologies. Therefore, along with the inherited features of the three main technologies, the MCC also inherits issues from all of the three domains. Research in the MCC domain aims to augment storage capability [19] and processing power [20] of the resource-constrained mobile devices by migrating the application to the cloud-based servers [18]. The storage capabilities of mobile devices are enhanced by leveraging the cloud storage resources for mobile cloud applications. The processing capabilities of the mobile devices are enhanced by migrating and processing the compute-intensive parts of the application in the cloud. Because the resource-intensive computational operations are performed in the cloud, the battery life of the mobile device is increased.

The resource-constrained mobile devices are possibly augmented by migrating the entire/partial application to four kinds of cloud-based computing resources: (a) the remote cloud data centers [21], (b) the proximate resource-rich computers [7], (c) group of nearby mobile computing devices [6], and (d) hybrid models in which a multitude of heterogeneous computing resources in various locations are used [4]. The cloud-based augmentation solutions provide the required computing resources in dynamic, on-demand manner, and pay-as-you-use fashion. On demand elasticity enables mobile service consumers to instantaneously scale up and down resources for execution of resource-intensive mobile application anytime anywhere from any mobile device and pay for the exact resources consumed.

2.2. Cloud-based Mobile Application Execution

Mobile application whose execution requires cloud-based resources, are named cloud-based mobile applications. Such applications inherit client-server traits and require a distributed environment for efficient execution. Figure 1 presents the flowchart of the application execution process. The execution starts on the resource constrained mobile device where the user preferences and application requirements are validated for the remote execution. If the user prefers to offload and the offloading process conserves the energy, then optimum remote resources for execution are opted. If the remote execution is infeasible, then a warning message is sent to notify the mobile device. Otherwise, resources are allocated remotely and the application is executed on the remote server. Finally, the results are pushed back to the mobile device, where the data is processed and on the completion of execution the application ends.

2.3. Seamless Application Execution in the MCC

The seamlessness of cloud-based mobile application execution is the most distinguishing feature compared to the normal application execution. Therefore, clear definition of the term seamless, is essential for an effective comprehension of the topic.

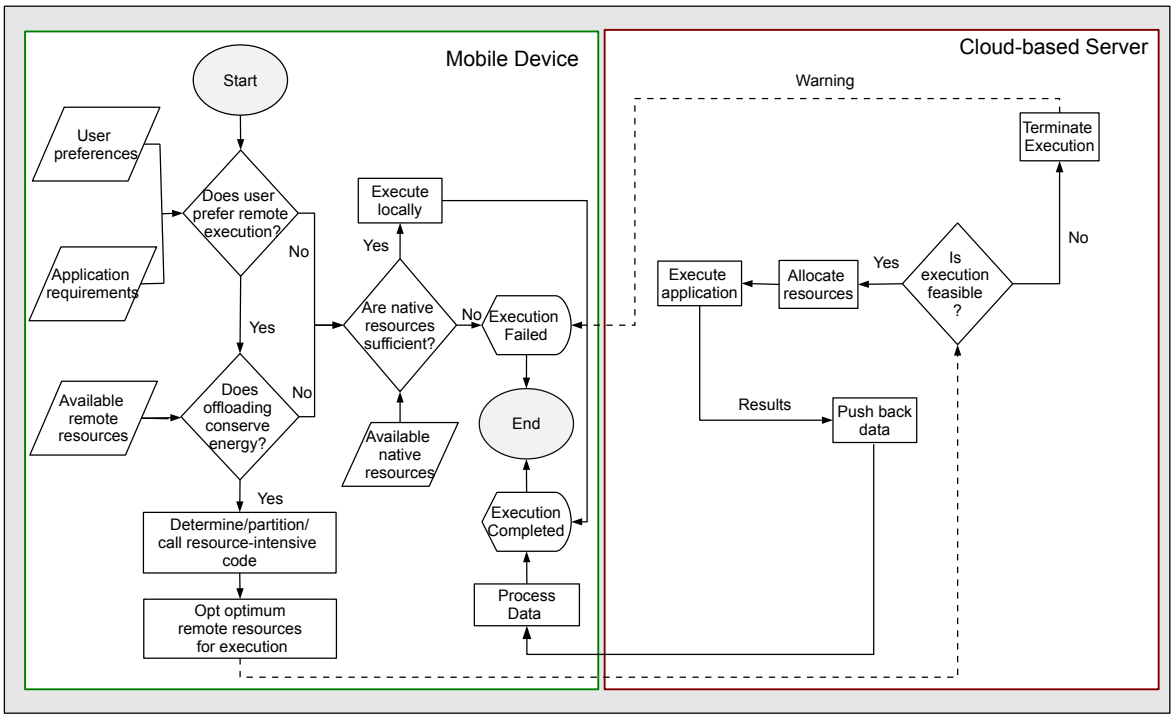

Figure 1: Flowchart of Cloud-based Mobile Application Execution in the MCC

2.3.1. ***Definitions***

The term seamless is used in various research areas, particularly networking and computer applications. Therefore, we present some of the most relevant definitions of the term seamless, followed by our definition and brief description of the concept of seamless application execution in the MCC.

Seamless Transition: The term transition is used when the current state of the system turns into a new state that creates changes in the entire system. According to M. Satyanarayanan *et al.* [22], "a seamless transition is one that involves a potentially disruptive state change, yet hardly distracts the user."

Seamless Mobility: Seamless mobility is "the capability to change the mobile device's point of attachment to an IP-based network, **without losing established connections and without disruptions** in the communication" [23].

Seamless Interface: In software engineering, seamless interface is "an interface that carefully joins two different programs together so that they appear to be a single program with a single user interface" [24]. A seamless interface provides a uniform interface to a user for two or more different programs.

From the above definition, it can be observed that the seamlessness raises when more than

one entity is entertained in the system. Moreover, the main characteristic of the seamlessness is unobtrusiveness. Accordingly, we present our definition of the seamless application execution as follows:

Seamless Application Execution in MCC: Seamless application execution in the MCC is "the state of unobtrusive application execution with the least possible user involvement, interaction, and distraction aiming to deliver enhanced functionality, higher performance, and improved responsiveness towards a richer user experience". The seamless application execution is only applicable to scenarios where a mobile application is executed on more than one node (a mobile device and a cloud-based server). The most important aspects of the seamless application execution are the application response time and the user interaction delay. The following attributes of the application response time with respect to user satisfaction level are highlighted in [25] as: (a) the user productivity is unaffected by the application response time below 150 milliseconds; (b) the user becomes aware of the application response time when the application response time is within the range of 150 milliseconds and one second; and (c) the user gets frustrated when the application response time and the user interaction delay is more than one second.

To realize the seamless application execution in MCC, a lightweight offloading system is required, in which the application initiation overhead is low so that the application faces a minimum execution disruption.

3. Motivation for Seamless Application Execution

With the advent of MCC, mobile users want to run their real-time compute-intensive applications on mobile devices by leveraging the services and resources of cloud. To improve the performance of real-time compute-intensive mobile applications, the applications are migrated from mobile device to the cloud. However, the compute-intensive migration process and intrinsic limitations of mobile devices impede the smooth execution of these real-time compute-intensive applications in MCC. These limitations affect the usability and sustainability of real-time mobile applications, such as m-gaming, m-health, which require the seamless application execution in MCC. Hence, it is vital to design the application execution frameworks that can enable the seamless application execution in MCC.

The motivation for the seamless application execution can be clearly illustrated by the example scenario presented in [26]. "A severe earthquake has just hit California. All of the communication lines, power cables, and highway infrastructures are severely damaged. The Internet infrastructure, including many main data centers, have suffered significant damage. Despite the heroic efforts, disaster relief is slow and inadequate relative to the destruction scale. The maps of the place, even GoogleEarth and GoogleMaps, are no longer usable because of destruction of buildings, bridges, highways, and landmarks. In this context, the searching and rescuing mission is dangerous and difficult because of the obsolete information. The maps must be reconstituted from scratch, and at an acceptable resolution to be effectively used in the rescue missions. In this despairing situation, local citizens support the rescue team by taking hundreds of close-up photos of the disaster strucken

place by using mobile phones cameras. These images are then stitched into a zoomable panorama using compute-intensive vision algorithms. Because the algorithms cannot run on a single smart mobile device, in this situation, a hybrid mobile cloud is formed by using the cloudlet deployed in the emergency vehicle and mobile phones of the local citizens and the rescue teams. The task of stitching hundreds of images is executed on multiple smart mobile devices. The images are transmitted via a low-grade wireless technology. Using the stitched photos and topographical overlays, detailed maps are drawn from the bottom-up. As the maps become available, rescue works in those areas are sped up. Although rescuing people is still dangerous, at least the search teams now have an accurate information." As the time is critical in this scenario, it is vital to enable the seamless application execution for such real life problems. The seamless application execution timely empowers the rescue team by reducing the delay in constructing the maps of the demolished area.

However, there are number of challenges involved in realizing the seamless application execution in this scenario. The device discovery to form mobile ad-hoc cloud is the main challenge to realize the seamless application execution. The application partitioning, task scheduling, offloading of application, authentication, authorization, and ensuring privacy of personal data on mobile device are other common challenges in realizing the seamless application execution.

4. Research Method

We have conducted a systematic review to execute in-depth unbiased literature survey on the cloud-based mobile application execution frameworks to investigate their suitability for the seamless application execution in MCC. The objective of this study is to identify the seamless application execution enabling approaches employed by the application execution frameworks in MCC. Moreover, the purpose of the study is to identify principles for seamless application execution and find the open research challenges in attaining the seamless application execution in MCC.

Table 2: Number of Studies per Article Type

Article Type	**Journals**	**Conferences**	**Magazines**	**Book Chapters**	**Workshop**	**Symposium**	**PhD Thesis**
Number of Studies	4	12	1	3	3	1	1

The criteria used to find and consider the research articles is based on following factors: a) source of study must have web search mechanism, b) complete research articles must be accessible for download using contracts between our university and the digital library, c) importance of sources, and d) relevance of sources. We have used the following digital libraries for searching the articles:

a) ACM digital library, b) IEEE Xplore digital library, c) ScienceDirect, d) Springer, and e) simple browsing through google search engine. We have only considered the research articles that covers the application execution in MCC. We have excluded the following types of articles: a) studies that do not have the full text available, b) repeated studies that are published in more than one source, c) talks, demonstrations, or short papers. Tables 2 and 3 present the number of studies per article type and per source type, respectively.

Table 3: Number of Studies per Source Type

Source Type	**ScienceDirect**	**Springer**	**ACM Digital Library**	**IEEE Xplorer**
Number of Studies	3	5	6	8

5. State-of-the-art Cloud-Based Mobile Application Execution Frameworks

In this Section, we provide a comprehensive survey of the state-of-the-art CMAEFs, analyze them with respect to seamless application execution, and highlight the critical aspects and implications in context of the seamless execution. The state-of-the-art CMAEFs are classified into three main categories based on the deployment platform, namely: cloud-based frameworks, cloudlet-based frameworks, and hybrid.

5.1. Cloud-based Frameworks

The mobile device that uses the remote cloud to augment resources, act as a thin client while connecting through any of the wireless technologies to a remote cloud server [27], as shown in Figure 2. The cloud-based frameworks enjoy a diverse range of available services in the cloud, high computational power, low computation latency, and the on-demand availability of resources [18]. The application that leverages on the cloud are required to exchange the data between two ends, the mobile device and the cloud, using the wireless Internet resources, such as backbone wireless networks. The communication across the Internet is affected because of the high WAN latency, interoperability among the underlying different networking technologies, non-guaranteed bandwidth reservation, bursty losses and excessive delay due to congestion, and non-deterministic traffic load along the path. The increase in the mobile communication traffic in the wireless backbone network may cause congestion that increases the data transfer time from a mobile device to the cloud and vice versa. Besides, in the current mobile network topologies, the backbone links are not always accessible in some scenarios, such as during disasters and rescue missions [28]. The use of congested backbone links and non-guaranteed access to these backbone networks increase the probability of disruption during the access of services provided by the remote cloud.

We discuss some of the credible remote cloud-based execution frameworks that leverage on the remote cloud resources to augment the mobile device resources as follows.

X. Zhang *et al.* propose an *elastic application framework* [27] to enable the use of remote cloud resources in seamless manner. The framework provides optimal elasticity by incorporating multiple

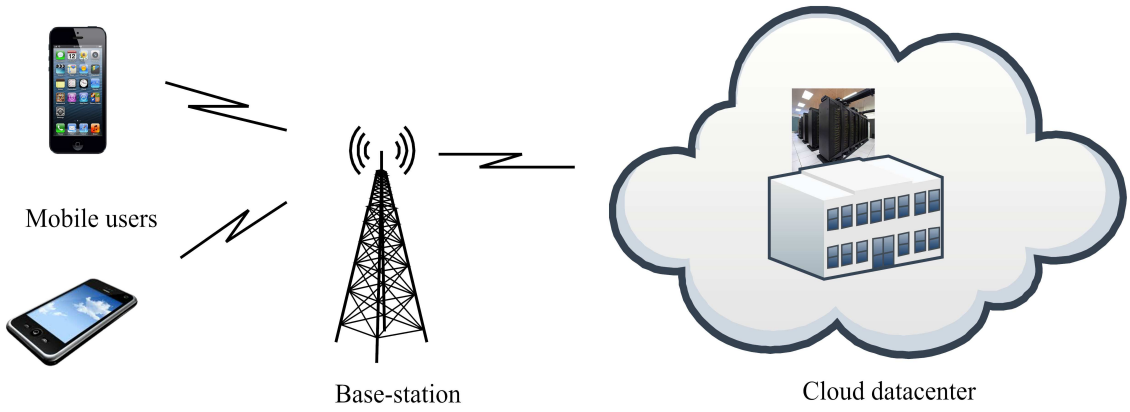

Figure 2: Cloud-based MCC Environment

factors, such as the availability of mobile devices and the cloud resources, user preferences, and the application performance measures. The proposed framework partitions the application into multiple components called weblets. The distinctive attribute of the framework is that the elastic partition components are replicated across multiple cloud servers that enhances availability and reliability. The model supports dynamic partitioning and transparent execution.

L. Yang *et al.* propose a *data stream application partitioning framework* [29] that aims to maximize the speed and throughput. The framework consists of modules running on both (mobile device and the cloud server) sides. Every mobile application in the framework has an application master on the cloud side. The application master performs two responsibilities; one is to find an optimal partition and to make the partitioning adaptive, and the other is the coordination of the distributed execution. The mobile device runs a profiler to measure, and to continuously monitor the device and network parameters. Although the framework supports multi-tenancy and provides scalable, adaptive, and dynamic partitioning, the proposed framework requires synchronization among different master nodes, which is challenging across the Internet and also results in high data exchange; thereby, inducing delay during the execution.

S. Kosta *et al.* propose *ThinkAir* [30], a dynamic and adaptive framework that exploits multiple mobile device virtualization to simultaneously execute multiple offloaded methods to reduce the application execution time. The ThinkAir supports six types of VMs with different CPU, memory, and heap configurations. The default server is called the primary server that always stays online and the rest of the servers are secondary servers. As the user requirements vary according to the type of application, various VMs are allocated to different user demands. The ThinkAir offloading decision is complex due to the intensive profiling mechanism. Moreover, the Android versions are required to be installed in the cloud. Therefore, the complex offloading decision and the platform dependencies deteriorate the seamless application execution.

B-G. Chun *et al.* present *CloneCloud* [31] that supports a flexible application partitioner to empower the unmodified mobile applications to seamlessly offload the compute-intensive partitions to the trusted remote cloud. The system supports dynamic profiling and static analysis to partition the mobile application. The main goal of the partitioning is to optimize the energy usage and the execution time. Unlike its counterparts, the *CloneCloud* partitions the application at thread level. Although the CloneCloud provides the dynamic automatic application partitioning and user-transparent clone migration, *CloneCloud* needs installation of compatible clone VM along with the application on *CloneCloud*. The requirements of compatible clone VM and the application installation obstruct the seamless application execution, if the clone VM and the required application is not already installed.

A cloud-based *mobile application execution framework* is proposed in [32]. For migrating an application, the framework first pauses the application on the mobile device, sends the saved state data files by the application to the cloud, and finally resumes the application execution in cloud server. To avoid the loss of input data, application replay technique is integrated with a state-saving scheme. The framework does not demand an application redesign; however, the application level state saving approach and the data categorizations are added to prioritize the data synchronization. Unlike the traditional VM-based application migration, the proposed framework only requires the transfer of the application saved states instead of the states of the entire VM. Although the framework reduces the amount of data transfer, it still faces the delay induced by the run-time agent program installation, VM creation, and states migration that hinders the realization of seamless application execution for the MCC.

T. verbelen *et al.* propose a *dynamic deployment and quality adaptation framework* [33] that selects the configuration with the best quality considering the current connectivity and available resources. The main unit of deployment in the proposed framework is a bundle. For each bundle, the framework provides multiple configurations with different levels of resources demands and offers various quality level. Although the framework can adapt according to the available resources and has an ability to make decisions at run-time in response to the varying available resources, the framework requires synchronization of data on the mobile device and the cloud server. The synchronization generates significant amount of traffic and requires processing at both ends.

A *replicated application execution framework* for application execution is presented in [34] where the application is replicated either partially or fully in the cloud. The offloading decisioning is performed on the cloud server. The framework frequently exchanges control and context information with the cloud data center to execute the complex offloading decision algorithm. Therefore, the solution is not scalable for highly dynamic workload environments and network conditions as the framework frequently exchanges the context information. Synchronization and data consistency among different nodes are the big challenges for the framework. Moreover, the framework does not focus on optimizing the execution time that is necessary for the seamless application execution in the MCC.

T. Verbelen *et al.* introduce *AIOLOS* [21], an adaptive offloading decision engine that dynamically considers the available resources of the server and varying network conditions in the offloading

decision. The framework selects the execution location of the application partition by identifying the estimated execution time for each of the method call, considering argument size, at both the mobile device and the remote cloud server. Aside from that, the adaptive offloading algorithm and bundle integration are compute- and energy-intensive. All of these limitations demand reconsidering the framework design to support the seamless application execution.

A middleware framework, *Calling the Cloud*, is presented in [5] that automatically and dynamically partitions and offloads various parts of an application to the cloud. The framework aims to minimize the interaction latency between the mobile device and the cloud server, while taking care of the exchanged data overhead. The framework uses two different algorithms, namely: ALL and K-step where the problem is formulated as static and dynamic optimization problem, respectively. In static partitioning, the partitioning is performed in the offline mode without incorporating the network conditions of the mobile device. In the dynamic partitioning, the partitioning is computed in run-time when the connection between the mobile device and the remote cloud server is established.

Coalesced offloading framework is proposed in [35] that saves additional energy by leveraging the property of multiple applications to bundle their code offload requests. By sending code offloading requests in a bundle, the period of time for which the network interface remains in high power state is reduced; thereby, saving mobile device energy. The problem of coalesced offloading is formulated as a joint optimization problem with the objective to minimize the response time and energy cost. Although the middleware framework reduces the interaction latency, other elements, such as consumption graph modeling, optimal cut finding algorithm, two-step dynamic partitioning analysis, and intensive profiling consume computational resources of the mobile device. Therefore, the compute-intensive nature of the framework increases the overall application execution time that impedes the vision of realizing the seamless application execution.

5.2. Cloudlet-based Frameworks

Cloudlet is a local cloud platform that is deployed on a resource-rich device, such as WLAN server, or on a group of mobile devices. The cloudlet mitigates the issues of accessing the remote cloud, such as WAN latency, jitter, packet losses, and failure, by offloading the mobile client workload to nearby available devices using the WLAN technology [7]. The cloudlet-based frameworks aim to mitigate the communication overhead compared to the cellular networking technology that is used to access the remote cloud. However, the work presented in [36] uses the cellular networking technology to utilize the local resources. The cloudlet-based solutions can be classified into two types according to the deployment location: server-based cloudlet and mobile ad-hoc cloudlet. The cloudlet-based frameworks do not require the Internet connectivity to access and to utilize the cloudlet resources. Figure 3(a) and Figure 3(b) show the server-based cloudlets and mobile ad-hoc cloudlets, respectively.

(a) Server-based Cloudlet (b) Mobile Ad-hoc Cloudlet

Figure 3: Cloudlet-based MCC Environment

5.2.1. Server-based cloudlet

In the server-based cloudlet, the cloudlet is deployed on a local server in the WLAN or in the TSP network that can be accessed through the WLAN technologies or the cellular wireless access technologies. Figure 3(a) illustrates the server-based cloudlet. The server performs management, resource monitoring, and task scheduling. Moreover, the server is considered as a secure and trustful resource-rich device. Although the server-based cloudlet framework reduces the response time of the application execution in the MCC by mitigating the WAN latency, the frameworks suffer from the compute-intensive and delay-inducing processes of cloudlet discovery, VM creation and deployment, application partitioning, offloading, authentication, and authorization. The local network service provider also hesitates in deploying the cloudlet and providing access to outside users for accessing the private networks. There is a need to provide incentives, such as charge per-resource-usage, to provoke the local network service providers to allow the outside users for accessing and utilizing the network resources. Security and authorization of local network resources are the big challenges in cloudlet deployment to prevent outside users from accessing the private resources. We review some of the most credible approaches that are proposed based on server-based cloudlets as follows.

A design of a *cloudlet-based network* and an adaptive decision-based service architecture are presented in [37]. The cloudlet-based network consists of the cloudlets server and mobile nodes affiliated with the closer cloudlet. The mobile nodes within the network can communicate with the server and with other mobile nodes in the network using the wireless links. Two new routing algorithms are proposed following the centralized and distributed approaches. In the centralized routing algorithm, each of the cloudlet sends the attached mobile node's ID along with its own ID

to a centralized server. The centralized server computes the route and sends back to the routers. In the distributed routing algorithm, the routing is performed distributively by the cloudlets using the local information. Although the cloudlet-based approach outperforms the cloud-based approach, it still suffers from the initial delays due to service discovery, authentication of mobile users, and joining process.

Mirror server-based framework is proposed in [36] that maintains the VM template for each of the mobile device inside the telecommunication network computing infrastructure. The framework feasibility is tested by designing a protocol, performing scalability test, and developing synchronization methods. In the framework, some workload of the mobile device is shifted to its mirror on the cloud server. Cloud server can host hundreds of mirrors that are implemented as VMs. The framework provides loose synchronization between the mobile device and the mirror server, such that the server replays the input to the mobile device on its mirror in the same order. The framework has high synchronization overhead that consumes network bandwidth. The virtual machine instance creation and mobile device mirror deployment induce the delay before the actual application execution starts.

The *XMPP-based mobile cloud middleware* is presented in [38] that provides dynamic application partitioning and adaptive offloading to the nearby resource-rich devices. The offloading decision is based on a context-aware cost model. The XMPP-based protocol attains a real-time messaging by reducing the network latency. Moreover, the optimization model is supported with a lightweight, efficient, and predictive decision algorithm. The prediction algorithm runs on the device; whereas the optimization model executes on the cloudlet server. Although the approach provides a lightweight offloading mechanism in terms of computation, the middleware must exchange a number of control messages especially when the mobile device and network conditions are highly dynamic.

An *augmented smartphone application* offloading architecture is proposed in [20] that uses loosely synchronized virtualized mobile replicas in the cloud. The architecture reduces the bandwidth consumption by leveraging the incremental checkpointing system replication. Although the framework is lightweight due to smaller size of the transferred data and lesser synchronization overhead, there exist incompatibility issues that arise, if clone VM is not already installed. The mobile clone instance migration arises concerns and issues related to privacy, access control, security, VM deployment, and management.

A fine-grained *dynamic cloudlet* approach is proposed in [39] that deploys a cloudlet on any device in local area network with sufficient available resources. The unit of deployment is a component. The dynamic cloudlet eliminates the following drawbacks of the static cloudlet: (a) service providers require deploying a cloudlet infrastructure in local area network. To alleviate such a constraint, a dynamic cloudlet approach is introduced. All of the devices share resources in a dynamic fashion. (b) As a unit of distribution, the VM-based cloudlet uses a coarse granular approach of VM distribution. Although the dynamic cloudlet overcomes the limitations of static cloudlet, there exist challenges of optimal tasks mapping that are critical for the seamless application execution.

COMET [40] is proposed to transparently migrate the multi-threaded applications to the locally available servers. COMET incorporates the workload of machines in the decision making process

of threads migration. Moreover, COMET leverages on the distributed shared memory techniques, such as field level granularity, to attain the consistency across the end systems; thereby, enabling the migration of any number of threads. Although the COMET transparently migrates the code to the cloud, the framework imposes the synchronization overhead across the endpoints.

5.2.2. Mobile ad-hoc cloudlet

The mobile ad-hoc cloudlet [6] is a group of mobile devices that shares resources with each other in the local vicinity to reduce the resource limitations of individual devices. Figure 3(b) illustrates the mobile ad-hoc cloudlet. The mobile ad-hoc cloudlet is an alternative solution to the server-based cloudlet that mitigates several WAN bottlenecks, such as low throughput and longer delay. The mobile ad-hoc cloudlet provides the resources to mobile devices when either no or weak wireless Internet connections are available to the cloud or the device is unable to find the nearby available resources [6]. In the mobile ad-hoc cloudlet, the mobile devices have to perform management functions, authentication, resource monitoring, and task scheduling by themselves in a distributed manner that consumes mobile device energy and processor cycles. Herein, we review the some of the credible frameworks that are proposed based on the mobile ad-hoc cloudlet.

A *virtual mobile cloud computing* framework for mobile ad-hoc cloud is presented in [6]. The framework enables a mobile device to discover other mobile devices in the local vicinity that are in a stable mode. After discovery of the mobile devices, the target resource provider is changed to a virtual cloud that is formed on the fly. Each of the mobile device is required to detect the neighbours and keep the profile updated. Moreover, the neighbour discovery is a time-intensive process that induces the delay before the actual execution starts.

E. Koukoumidis *et al.* propose a *pocket cloudlet* architecture [41] that leverages the large available memory capacity of the mobile devices to alleviate the latency and energy issues in accessing the distant cloud services. The architecture uses both the community access models and the personal access model to maximize the hit rate of user queries; thereby, minimizing the overall service latency and energy consumption. Considering the storage potential of a mobile device, either partial or full cloud services can be migrated to an actual mobile device; thereby, transforming a mobile device into a pocket cloudlet. The provisioning of a mobile device's storage as a service to other mobile users requires effective authorizing and data integrity mechanisms to protect the mobile data from unauthorized access. However, the pocket cloudlet framework does not address the above mentioned issues.

5.3. Hybrid Frameworks

Some of the existing frameworks (e.g. [4, 7, 8, 42, 43]) leverage on the hybrid platforms available in different clouds. The hybrid frameworks overcome the limitations of both type of clouds. Accessing the remote cloud resources and services, arises a number of issues, such as WAN delays, jitter, congestion, and failures [7]. On the other hand, the cloudlets have limited resources that cannot scale well with the increasing load of the users within the cloudlet network.

The hybrid frameworks leverage on the nearby available cloudlet that is either deployed on a group of mobile devices or on a server within the WLAN/TSP network. The support of hybrid platform provides the flexibility to the CMAEFs to run time-critical components of application on the nearby low-latency cloudlet and delay-tolerant components of an application on the remote high-latency cloud servers. Figure 4 illustrates the execution environment of hybrid CMAEFs. We

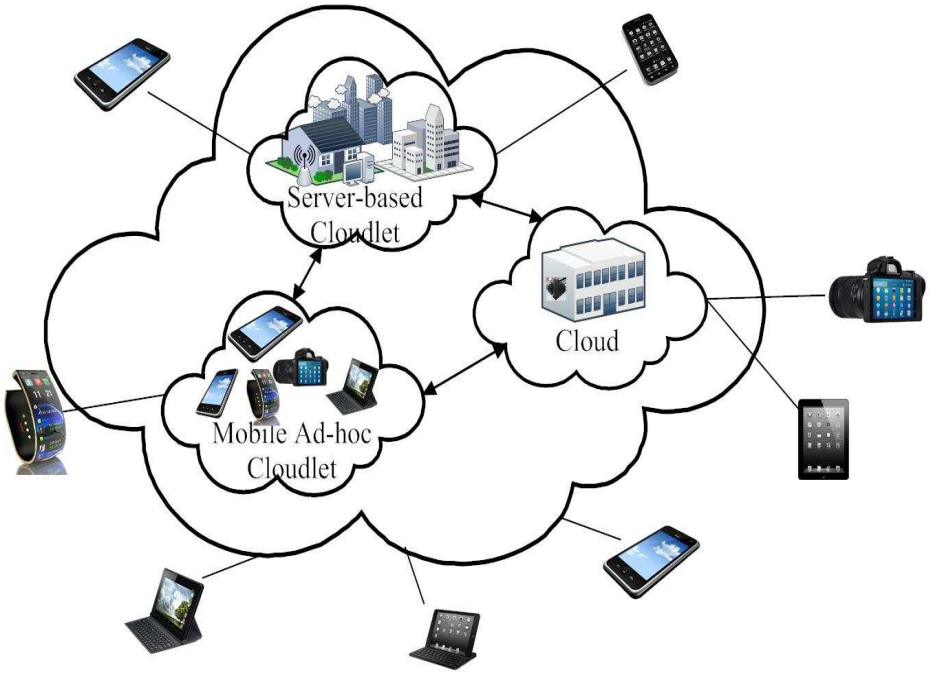

Figure 4: Hybrid MCC Environment

discuss some of the hybrid frameworks that leverages on the multiple available cloud platforms to augment the mobile resources as follows.

M. Satyanarayana *et al.* introduces the concept of *VM-based cloudlet* to leverage on a cloud computing platform that is free of WAN delays, jitter, congestion, and failures [7]. In case of the unavailability of a nearby cloudlet, the mobile applications either run on remote cloud resources or locally on mobile device. The authors employ a dynamic VM synthesis approach for migrating an application. In the approach, a mobile device sends a small VM overlay to the cloudlet that already has a base VM from which the overlay VM was derived. Thereafter, the cloudlet infrastructure derives the launch VM by applying overlay to the base VM. The cloudlet-based approach reduces high WAN latency, transparently executes the application, and generates relatively less amount of traffic. However, the VM synthesis takes a significant amount of time to be initiated and executed that impedes the vision of realizing the seamless application execution. Moreover, the VM migration also raises issues related to security and privacy [7].

D. Kovachev *et al.* propose *Mobile Augmentation Cloud Services (MACS)* [42] an adaptive middleware framework that provides lightweight application partitioning, seamless computational offloading, and resource monitoring. MACS enables mobile applications to take benefit from the seamless computational offloading into a remote or nearby cloud. The developers do not need to change the application model to run an application in the cloud. The partitioning decision is transformed into an optimization problem that is solved by a solver to dynamically partition the application. The MACS requires a strong developer support for structuring the code in a model. Despite lightweight application partitioning and seamless computational offloading, an additional layer of proxy service in MACS adds more delay in the application execution.

A lightweight *cyber foraging framework* is devised in [8] for mobile and embedded resource constrained devices that employs lightweight discovery protocol to discover potential surrogates. The proposed framework follows a client/server architecture that enables the mobile devices to offload the compute-intensive tasks on the surrogate server to utilize the computational resources of the server. Each migrated application runs on isolated virtual server. The cryptographic measures are employed by the framework to ensure the secure communication between mobile device and surrogate server. The framework uses low overhead client side authentication and ciphers. However, due to the time consuming VM mechanisms, the solution is unfavorable for the seamless execution of an application in the MCC.

E. Cuervo *et al.* propose *MAUI* [4] as an energy-aware fine grained, method level mobile application offloading mechanism that supports a semi-dynamic partitioning; wherein, programmers annotate an application with a considerably less effort. The profiler component of the MAUI assesses a method on call time for energy saving; whereas, the solver component takes the decision of migrating the method based on the input from the MAUI profiler. MAUI employs a time-out mechanism to detect the failures in connection with the server. Although MAUI significantly improves the battery life of a mobile device, it does not address scalability, QoS, and transmission latency.

E. E. Marinelli *et al.* design a *Hyrax*-based platform in [43] to provide cloud computing services on a group of mobile devices and servers. The platform employs a mechanism for tolerating node departure. Hyrax is developed by extending the Hadoop [44] framework for Android mobile device. The platform relies on a centralized server and a group of mobile devices. The discovery of a centralized server, access, and connection establishment among the servers is a time-intensive task that induces the delay before executing the actual application.

A *decentralized computation offloading game* [45] is designed to model the offloading decision making problem among mobile users. The game considers both computation and communication cost of MCC. The authors have analyzed the computation offloading game in two types of wireless access scenarios: homogeneous and heterogeneous. In homogeneous scenario, the existence of Nash equilibrium is guaranteed by admitting the beneficial cloud computing group structure. In heterogeneous scenario, the existence of Nash equilibrium is guaranteed by admitting the finite improvement property. Although the solution reduces the controlling and signaling overhead of the cloud, structuring a game such that an equilibrium is always reachable is difficult.

A *semi-markovian decision process-based offloading* framework to solve the optimal application management problem in MCC is presented in [46]. The execution is either performed locally or remotely on a cloud server. The local processing unit can dynamically adjust the processing speed and power consumption of the processing unit by employing dynamic voltage and frequency scaling (DVFS). Moreover, the transmitter can adaptively select the most appropriate modulation scheme and bit rate for offloading request considering the channel capacity, the number of waiting requests, and server congestion levels. A semi-Markov decision process is used to model the mobile device where actions are decision pairs (bit rate, DVFS level) for the transmitter and the local processor in the mobile device. The identification of optimal transmission scheme, offloading rate, and optimal DVFS policy needs continuous monitoring of the running environment.

From the comprehensive survey of state-of-the-art CMAEFs, we have identified a number of seamless application execution enabling approaches employed by the frameworks to improve the response time of the application within the MCC environment. We classify the seamless application execution enabling approaches according to the corresponding implementation locations. The taxonomy of these approaches is presented in next section.

6. Seamless application Execution Enabling Approaches Employed by CMAEFs

This Section presents the thematic taxonomy of seamless application execution enabling approaches employed by the current CMAEFs for the MCC. The thematic taxonomy is beneficial for mobile cloud application developers and cloud service providers to employ the approaches that can reduce delays when developing applications or providing services; thereby, increasing QoS for mobile cloud users. The taxonomy is an outcome of a comprehensive survey of the state-of-the-art CMAEFs presented in Section 5. This section also investigates the advantages and disadvantages of seamless application execution enabling approaches. The strengths and weaknesses of state-of-the-art CMAEFs with respect to seamless application execution are also presented.

6.1. Thematic Taxonomy of Seamless Enabling Approaches and the CMAEFs Comparison

Figure 5 shows the thematic taxonomy of the seamless application execution enabling approaches employed by the CMAEFs for the MCC. The approaches are categorized based on implementation locations into four classes, namely: mobile-centric, cloud-centric, network-centric, and a hybrid approaches.

6.1.1. Cloud-centric Approaches

Cloud-centric approaches are implemented in the cloud to enable the seamless application execution. Cloud-centric approaches are caching [7, 8, 36, 37, 42], mirroring [36], parallel execution [30], pre-installations [8], and deploying cloudlets [6, 7, 37, 39, 43].

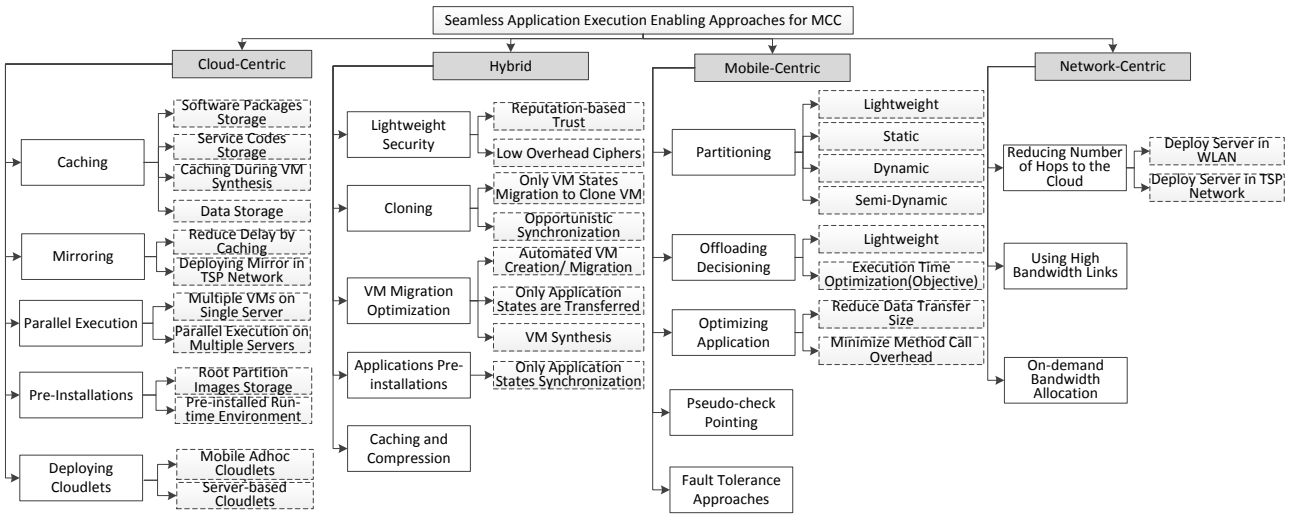

Figure 5: Taxonomy of Seamless Execution Enabling Approaches Employed by the CMAEFs for the MCC

Caching. Caching enables the framework to store the remote data locally to cope with longer WAN delays. Data is cached locally based on the user requests that reduces the access latency for later use. Caching is further classified into four classes according to the contents, namely: software package caching [8], service code caching [42], VM synthesis caching [7], and data caching [37]. Software package caching approach employs caching to store software packages. Such an approach allows the trusted user to install packages directly without facing any download delay. Service code caching approach is used to store the service code (jar files) when client sends data to the cloud for the very first time. The VM synthesis caching approach is used either as a pre-fetching technique to reduce the VM synthesis delay by caching the VM overlay or caching the VM launch for future use. Lastly, the data caching approach temporarily stores the data such as video streams on the local cloudlet. Thereafter, the cloudlet sends the cached data to the subsequent mobile users on their requests. The caching of data on the local cloudlet reduces the latency involved in video streaming.

The caching of contents on local servers reduces the WAN latency; thereby, minimizing data access delay. The caching integration with pre-fetching in the VM synthesis process reduces the VM synthesis delay. Moreover, the caching also lessens the network redundant traffic. In spite of the aforementioned advantages, the caching increases the storage overheads and cost in the cloud. Moreover, the cached data can be outdated for the user.

Mirroring. Mirroring is another cloud-centric approach employed to attain the seamless execution of an application within the MCC. The framework presented in [36] employs the mirroring approach to reduce the delay by caching at the mirror for downloading and uploading. When a mobile device downloads a bulk of data from the Internet, a mirror server first caches the data and then the

mobile device can download the data from the mirror cache. Similarly, to upload data, the mobile device merely sends a command to the mirror server that replays the command with the data. Even for multiple receivers, a mobile device need only to send a command just for a single time. The performance of application can also be improved by deploying a mirror in the nearby TSP network.

The mirror deployment reduces the operational overhead of a mobile device by transmitting only commands of operations from mobile device; actual operations on data are performed on the mirror server. The mirror-based frameworks alleviate the redundant traffic in the network. Moreover, mirror server minimizes the data access latency by caching the data on the server for subsequent users; thereby, optimizing the response time of applications. However, the mirroring approach increases storage cost at cloud. The mirror server requires management for updates, upgrades, new versions, and bug fixes. Furthermore, the mirror-based frameworks require synchronization between the mobile device and the mirror server.

Parallel Execution. Parallel execution is employed to minimize the application execution time in the cloud data center. We classify the parallel execution approach into two classes. One uses multiple VMs on a single cloud server and the other performs parallel execution on multiple servers. Using multiple VMs on a single server reduces the server identification and selection time; whereas, the VM deployment on multiple servers mandates the identification of multiple servers and to send two distinct requests to each server. Using the VM on multiple servers increases the network overhead and operational complexity. A framework proposed in [30] uses the aforementioned approaches to improve the seamless execution of an application. Elastic mobile applications [27] leverage on parallel processing in cloud to improve the response time and load balancing by deploying weblets on multiple servers within the cloud.

The parallel execution reduces application execution time in cloud and facilitates in load balancing. The parallel execution support makes the framework highly scalable. However, the parallel execution consumes more power and requires more hardware. Moreover, the parallel execution mandates the identification of interdependency among tasks to avoid deadlocks.

Pre-installations. Pre-installation reduces the initiation and preparation delay of remote application execution by providing access to the already installed software packages. The pre-installations are classified into two categories. One category facilitates the mobile user by installing root partitioned images with the necessary applications, system boot-up scripts, and essential software packages on the cloud server. The researchers in [8] encourage the usage of already installed applications, system boot-up scripts, and software packages to reduce the pre-execution delay. The other category provides a pre-installed run-time environment [34] that facilitates in the deployment of mobile applications into the cloud to reduce the remote application execution initialization delay.

The pre-installation of applications minimizes run-time transmission of data by eliminating the need of migrating the application from the mobile device to the cloud server. However, the pre-installation increases the resource consumption cost in the cloud if the applications and software packages are not frequently used. The pre-installation also increases the cloud service provider's services maintenance overhead.

Deploying Cloudlets. Deployment of cloudlets enables the seamless application execution by exploiting the capabilities of trusted resource-rich local system either in the WLAN or in the TSP's network. We classify these cloudlets based on the deployment structure into three subclasses: (a) infrastructure-based/ static/ server-based cloudlet, (b) infrastructure-less/ mobile ad-hoc cloudlet, and (c) virtualized infrastructure-based/ elastic cloudlet. M. Satyanarayanan *et al.* [7] present an infrastructure-based cloudlet that is deployed on the local server in the WLAN. Infrastructure-less or mobile ad-hoc cloudlet is proposed in [6] that does not require any local server for the deployment of cloudlet while all of the mobile devices with sufficient idle resources form a mobile ad-hoc cloudlet. An elastic cloudlet is presented in [39], in which a component level application migration is performed instead of a whole VM migration. The approach provides flexible allocation of resources and facilitates in prioritizing the deployment of real-time components of an application in the cloudlet; whereas, delay tolerant components are offloaded in a remote cloud.

The cloudlet deployment closed to user reduces the WAN latency and minimizes the data transport cost. The cloudlet-based frameworks do not need to connect to the Internet. In spite of the aforementioned advantages provided by the cloudlet, the cloudlet requires network isolation for outside users to prevent them from accessing the private network resources. Moreover, the practical realization of the cloudlet deployment requires investigation of incentives for cloudlet service providers to deploy cloudlet. The cloudlet deployment also requires a well-defined charging policy as per usage basis. The cloudlet service providers need more resources in their local networks to provide the on-demand services and resources to the MCC users.

Table 4 shows the comparative summary of the CMAEFs based on the seamless application execution enabling cloud-centric approaches.

Table 4: Comparison of CMAEFs based on seamless application execution enabling cloud-centric approaches

Application Execution Frameworks	**Cloud Usage Overhead**	**Deploys Mirror**	**Pre-execution Delay**	**Caching Support**	**Parallel Execution Support**
MAUI [4]	High	No	High	No	No
Virtual Mobile Cloud Computing [6]	High	No	High	No	No
Secure Cyber Foraging [8]	High	No	Low	Yes	No
VM-based Cloudlets [7]	Low	No	Medium	Yes	No
Augmented Smartphone Application [20]	High	No	High	No	No
AIOLOS [21]	High	No	High	Yes	Yes
ThinkAir [30]	High	No	High	No	Yes
Cloudlets-based Network [37]	High	No	High	Yes	No
Mirror Server-based Framework [36]	High	Yes	Low	Yes	No
Hyrax [43]	High	No	High	No	No
Cloudlet [39]	Low	No	Medium	No	No
COMET [40]	Low	No	Low	Yes	Yes
Pocket Cloudlets [41]	Low	No	Low	Yes	No
MACS [42]	High	No	High	Yes	No

6.1.2. Hybrid Approaches

Hybrid approaches are implemented on both the mobile device and the cloud server for attaining the seamless execution of applications in the MCC. Hybrid approaches include provisioning of lightweight security mechanisms [8], cloning [20, 31], [7, 32], optimizing VM migration [7, 32], pre-installation of applications that require only states to be transferred [27], and lastly, caching and compression of the data and metadata [47].

Lightweight Security. Security is one of the main challenges that are obstructing the major deployment of clouds. Security issues in cloud are investigated in [48–50] that are necessary to be resolved to rapidly increase the growth rate of the clouds. Aside from a number of security issues in the cloud, existing security mechanisms induce delay in the application execution that is obstructing the aim of the seamless application execution. The computational delay of security mechanisms can be reduced by designing and employing lightweight security approaches. Lightweight security is classified into reputation-based trust establishment [7], low overhead ciphers, and authentication mechanisms [8]. The lightweight approaches enable the seamless execution by reducing the delay involved in cipher execution, trust establishment, and authentication. The lightweight security mechanisms reduce the time taken by authentication process and minimize the resources consumption. Therefore, the application response time in MCC can be reduced. However, the security of the data can be compromised because of the employment of the lightweight security mechanisms and relying on reputation-based trust.

Cloning. The cloning improves the response time of an application by deploying the mobile device clone in the cloud. The mobile device clone that need to be deployed in the cloud, require only part of the application execution to be offloaded from mobile device into the cloud. The pre-installed mobile device clone alleviates the overhead involved in initiation and preparation of the application execution. During the application execution process, the states are migrated to the mobile device clone. The research works of [20] and [31] present the CloneCloud frameworks that execute an application on the nearby and remote servers. Moreover, an incremental checkpointing and opportunistic synchronization are used to improve the seamless application execution in MCC. Incremental checkpointing calculates and sends the difference of two checkpoints instead of all of the states that significantly reduces the overhead. The minimization of the amount of the state data transfer reduces the transmission delay that ameliorates the seamless application execution. The opportunistic synchronization approach synchronizes the state related information between the two ends so that the mobile device clones can be updated with low cost. The alleviation of synchronization overhead and reduction in amount of state transferred data reduces the bandwidth consumption of the network, minimizes the transmission delay, and conserves the battery power of the mobile device. The cloning minimizes the application initiation and preparation delay; thereby, improving the application overall response time. However, the mobile device clone deployment increases the maintenance overhead for the cloud service provider. The cloning also increases the resource consumption cost for the cloud service provider if the applications are not frequently used. Moreover, providing the clones of all available mobile devices is not practical solution in the MCC.

Optimizing VM Migration. The VM migration incurs a significant overhead that is investigated in [51]. The VM-based frameworks employ three strategies to optimize the migration process: (a) automation of VM creation and migration [32], (b) transfer of only application specific states instead of the whole VM [32], and (c) the VM synthesis [7]. The optimized migration approaches alleviate the delay involved in the VM creation and migration and reduce the transfer overhead by transferring only application relevant states and by replacing the full VM migration with the VM synthesis that requires low bandwidth and takes less transmission time. However, the VM creation, migration, and deployment in the cloud is time-intensive process. Although the VM synthesis reduces the migration overhead, the VM synthesis requires installation of VM base in the cloud which arises the compatibility issues.

Applications Pre-installation. Elastic mobile application framework presented in [27] employs an approach wherein the weblets are pre-installed on the remote server, and during the execution only states need to be transferred for execution of an application. The pre-installations of weblets reduce data transfer size, which lessens transmission delay, lowers the bandwidth consumption, and conserves the battery power. The application pre-installations reduce the application initiation and preparation time. However, application pre-installation increases the maintenance overhead on cloud server provider. Moreover, the pre-installed applications consume storage even if the applications are not frequently used.

Caching and Compression. The data caching and compression aid in locally caching the data and employing an adaptive compression to reduce the amount of transferred data [47]. The caching and compression approach not only reduces the latency but also the storage and network cost. Moreover, the compression significantly reduces the data transfer size. However, the compression increases processing overhead on resources-constrained mobile device; thereby, consuming a significant amount of battery power.

Table 5 shows the comparative summary of CMAEFs based on the seamless application execution enabling hybrid approaches.

Table 5: Comparison of CMAEFs based on seamless application execution enabling hybrid approaches

Application Execution Frameworks	**Security Overhead**	**Data Transfer Overhead**	**VM Migration Overhead**
COMET [40]	N/A	Low	Low
Elastic Application Framework [27]	High	Low	N/A
CloneCloud [31]	Medium	High	High
Augmented Smartphone Application [20]	Medium	High	High
Mobile Application Execution Framework [32]	High	Medium	Medium
VM-based Cloudlets [7]	Medium	Low	Low
Secure Cyber Foraging [8]	Low	High	N/A

6.1.3. Mobile-centric Approaches

In mobile-centric approaches, the main functionality for attaining the seamless application execution is implemented on the mobile devices. The mobile-centric approaches consist of partitioning, offloading decisioning, optimizing application, pseudo-checkpointing, and fault tolerance approaches.

Table 6: Comparison of the CMAEFs based on the seamless application execution enabling mobile-centric approaches

Application Execution Frameworks	**Partitioning Type**	**Partitioning Overhead**	**Offloading Overhead**	**Method Call Overhead**	**Data Transfer Overhead**	**Fault Tolerance**	**Transmissioin Delay**
COMET [40]	N/A	N/A	Med.	Low	Low	Yes	Low
Decentralized Computation Offloading Game [45]	N/A	N/A	Med.	N/A	Low	No	Low
Semi-Markovian Decision Process-Based Offloading [46]	N/A	N/A	Med.	N/A	Low	No	Low
Coalesced offloading framework [35]	N/A	N/A	Low	Low	Low	No	Low
Replicated Application Execution Framework [34]	DyP	Med.	N/A	High	Med.	No	Med.
XMPP-based Middleware [38]		High	Low	High	High	No	Med.
Elastic Application Framework [27]		High	High	High	High	Yes	High
Data Stream Application Partitioning Framework [29]		Low	High	High	High	No	High
AIOLOS [21]		High	High	Low	High	Yes	Med.
ThinkAir [30]		High	High	High	High	Yes	High
Dynamic Deployment & Quality Adaptation Framework [33]		High	High	High	High	Yes	High
MACS [42]		Low	High	High	Low	No	Med.
CloneCloud [31]	StP	Low	High	High	Low	No	Low
MAUI [4]	SdP	Med.	High	Low	Low	Yes	Low
Calling the Cloud [5]		Med.	High	High	High	No	Med.
Augmented Smartphone Application [20]		Med.	High	High	Low	Yes	Low

[1]Note:Med.: Medium, N/A: Not Applicable, DyP: Dynamic Partitioning, StP: Static Partitioning, SdP: Semi-dynamic Partitioning

Partitioning. Partitioning is performed in three ways, static [31], dynamic [21, 27, 29, 30, 33, 34, 38, 42, 52] and semi-dynamic [4, 5, 20]. The static partitioning requires developer support to annotate the remotely executable methods. Therefore, the static partitioning does not incorporate the conditions of the run-time environment during the partitioning decision. However, in the dynamic partitioning, a dynamic and adaptive decision engine takes the context information as input, computes partitions, and adapts according to the new partitioning results. Although the static partitioning performs lesser run-time computations, the static partitioning approach does not optimally partition the application. The semi-dynamic partitioning takes the advantages of both static and dynamic partitioning. The semi-dynamic partitioning performs static annotation to reduce the

run-time computations and also incorporates the dynamic environment parameters for run-time partitioning. The lightweight application partitioning is vital for seamless application execution in the MCC that alleviates the complexity of migration process [53–55]. The lightweight partitioning approaches employed by the CMAEFs are reported in [42, 52]. The proposed approach in [52] employs a model that predicts the costs of different partitions. Thereafter, a simple approximate optimizer quickly solves the partitioning problem based on predicted costs. The authors transformed an ILP problem into an LP problem to attain speed up over the cost of accuracy. The lightweight application partitioning enhances the seamless execution by reducing the complexity of the problem and performs a quick partitioning. The lightweight partitioning reduces consumption of mobile device battery power. The static partitioning does not require the profiling so it is relatively less complex in nature. On the other hand, the dynamic partitioning incorporates the dynamic conditions of the mobile device and network for partitioning, so performs optimal decision. Moreover, the dynamic partitioning suffers with the complexity of profiling. Similar to static partitioning, semi-dynamic partitioning is less complex, however, the semi-dynamic partitioning also incorporates the dynamic conditions of the environment.

Offloading Decisioning. An offloading decision algorithm is required to be lightweight with an objective function of minimizing the response time of an application in the MCC. The researchers in [38, 42] proposed an offloading middleware that incorporates network latency in the offloading decision with the objective of optimizing the overall application execution time. The XMPP-based middleware [38] reduces the offloading overhead by sharing the cloudlet-provided VM for all of the attached mobile users. The offloading decision algorithms aim to optimize the execution cost [52], to minimize the energy consumption [4], to reduce the execution time [5, 30, 42, 56], to optimize the throughput [29], and to minimize the estimated network latency [21]. Such essential objectives make the offloading mechanism favorable to execute on a mobile device because of the cost-effective lightweight features. The offloading scheme that aims to minimize the execution time or to optimize the estimated network latency is a favorable choice for the seamless application execution.

The incorporation of the communication latency in the offloading objective function minimizes the application migration time and optimizes data transfer cost; thereby, reducing the overall application execution cost and time. However, the run-time application offloading decision algorithms have high computational complexity; thereby, consuming battery power of the mobile device.

Optimizing Application. Optimizing an application focuses on the reduction in the amount of transferred data and method calls overhead. The amount of transferred data is reduced by implementing incremental state deltas and fine-grained code offloading support [4]. The method call overhead is reduced either by reducing the number of method calls [4] or using callback functions [21]. The number of method calls are reduced by implementing a set of similar operations in a single method. The callback function consumes less CPU resources and does not exchange much data. The method call overhead increases the delay especially if the method is called remotely. The huge data size increases the transmission delay; thereby, increasing the overall execution time. Therefore, the application optimization is also an important feature in enabling the seamless application execution

in the MCC. The application optimization minimizes the amount of data sent on network; thereby, reducing the network bandwidth consumption and conserving the mobile device battery. However, optimizing the application design requires modification in application. Moreover, the optimization of the application requires programmer support.

Pseudo check-pointing. The pseudo-checkpointing does not need to explicitly save the states, which reduces the overall states storage and transmission overhead [32]. The reduction in the states storage and transmission overhead enables the fast transmission and execution that aids in realizing the vision of the seamless application execution. Pseudo-checkpointing minimizes the application states storage size and transmission overhead; thereby, optimizing the consumption of network bandwidth and mobile device battery. Moreover, pseudo-checkpointing minimizes the network usage cost. In spite of aforementioned advantages, the pseudo-checkpointing has two limitations: (a) requires modification in applications and (b) needs programmer support.

Fault Tolerance Approaches. Fault tolerance approaches reduce the disruption time because of either the link failures or server failures while an application is executed in the MCC. The frameworks that support fault tolerance can execute the application locally on mobile device, when the cloud server is inaccessible. The fault tolerance approaches alleviate the disruption during execution and provide user transparent failure detection mechanism. The CMAEF supported with fault tolerance approaches gracefully degrades the application performance. However, to provide the fault tolerance support, continuous monitoring mechanism is required to detect the failure. The proactive fault tolerance approaches require more functionality; thereby, inducing high complexity.

Table 6 shows the comparative summary of the CMAEFs based on the seamless application execution enabling mobile-centric approaches.

6.1.4. Network-centric Approaches

The network-centric solutions are employed in a network to alleviate network related issues, such as high WAN latency, jitter, and packet losses, to accomplish the seamless application execution in the MCC. The network-centric solutions employ three types of approaches that include reducing number of hops to the cloud, using high bandwidth links, and on-demand bandwidth allocation.

Reducing Number of Hops. The number of hops to the cloud are reduced by bringing the cloud closer to mobile user either by deploying servers in the WLAN [37] or in the TSP's network [36]. The reduction in the number of hops eliminates the issues of the WAN high latency, jitter, and packet losses that significantly improves application response time. Moreover, reducing the number of hops to the cloud minimizes the data transport network cost. The framework does not require Internet connection to access the cloud server if the cloud server is deployed inside the local network. However, the servers in local networks are not always accessible due to security policies. The cloud deployment in locally available servers also requires isolation from local networks to keep the outside users away from the private resources.

Table 7: Comparison of CMAEFs based on the seamless application execution enabling network-centric approaches

Application Execution Frameworks	Reduction in Number of Hops	Usage of High Bandwidth Links	Guaranteed Bandwidth
COMET [40]	Yes	Yes	No
Cloudlets [39]	Yes	Yes	No
MAUI [4]	Yes	Yes	No
Mirror Server-based Framework [36]	Yes	No	No
AIOLOS [21]	No	Yes	No
ThinkAir [30]	No	Yes	No
Augmented Smartphone Application [20]	Yes	Yes	No
MACS [42]	Yes	Yes	No
Virtual Mobile Cloud Computing [6]	Yes	Yes	No
VM-based Cloudlets [7]	Yes	Yes	No
Cloudlets-based Network [37]	Yes	Yes	No
Secure Cyber Foraging [8]	Yes	Yes	No
Hyrax [43]	Yes	Yes	No
Mobile Application Execution Framework [32]	No	Yes	Yes
Pocket Cloudlets [41]	Yes	Yes	No

Using High Bandwidth Links. The high bandwidth links are deployed by using the WiFi [37] and 3G networks [36] for offloading the application. The high bandwidth links reduce the transmission time during the offloading of an application in the MCC. However, the use of high bandwidth links in the cloudlet-based frameworks is more beneficial than the cloud-based frameworks because the path to the cloudlet has a direct link. Using the high bandwidth wireless access links does not ensure that the same bandwidth will be allocated along the whole path. The path may have bottleneck links to the cloud. The data of cloud-based frameworks may have to go through such bottleneck link across the Internet; thereby, reducing the overall application performance. Using the high bandwidth links provides high data transfer rate across the link. However, use of high bandwidth links increases monetary cost.

On-demand Bandwidth Allocation. The on-demand bandwidth is allocated to the users' traffic to ensure the bandwidth reservation according to the traffic requirements within the network [32]. The on-demand bandwidth reservation approach helps in ensuring the QoS for each application; therefore, it helps in realizing the seamless application execution by providing sufficient bandwidth to each of the flow according to the requirements. However, on-demand bandwidth allocation requires bandwidth negotiation with the network service provider. Moreover, the allocation of bandwidth to the network flows also requires fairness among the flows that requires additional functionality on the network elements. Such additional functionalities increases the overhead on the network elements.

The deployment of server in the WLAN and in TSP's network copes with the WAN issues, such as high latency, jitter, and packet losses that are obstructing the realization of the seamless application execution vision in the MCC environment. The use of high bandwidth links reduces the transmission delay that also assists in achieving the objective of seamless application execution

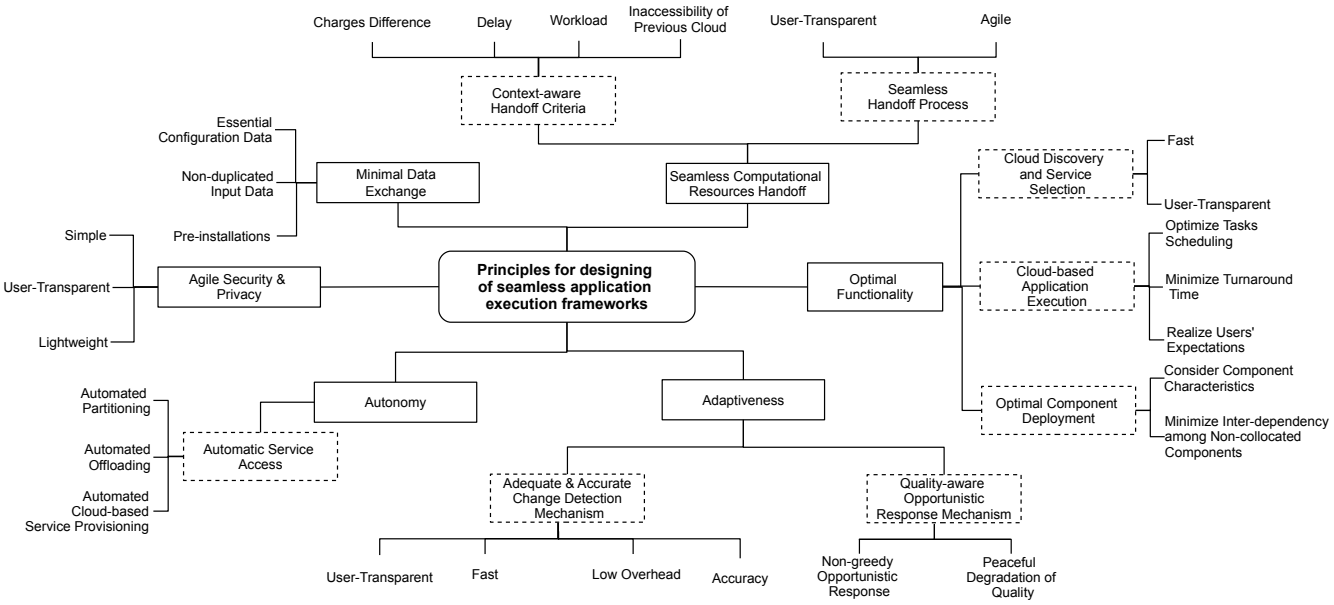

Figure 6: Principles for the Seamless Application Execution in the MCC

in the MCC. The on-demand allocation of bandwidth makes the execution framework scalable for varying demand loads of applications.

Table 7 shows the comparative summary of the CMAEFs based on the seamless application execution enabling network-centric approaches.

7. Seamless Application Execution Principles for MCC

We have identified the seamless application execution principles from the literature review presented in Section 5. This is done so that the framework designers can have straightforward guidelines for designing the frameworks that can satisfy the performance metrics for the seamless application execution in MCC. The principles for seamless application execution are classified into six main categories: optimal functionality, seamless computational resource handoff, adaptiveness, autonomy, minimal data exchange, and agile security and privacy. Herein, we discuss each of the principles in detail. Figure 6 shows the principles for designing of seamless application execution frameworks.

7.1. Optimal Functionality

The optimal functionality feature of the CMAEFs ensures the optimal design and implementation of functional properties of the frameworks. The functional attributes of the application execution can be divided into three categories: (a) seamless cloud discovery and service selection, (b) optimal cloud execution, and (c) optimal component deployment.

7.1.1. Seamless Cloud Discovery and Service Selection

To access and utilize the services of the cloud, a mobile device has to discover the cloud and specify the requirements in a real-time manner before actually accessing the services [8, 57]. The cloud discovery and service selection are required to be seamless to minimize the disruption during the application offloading. The seamless cloud discovery and the service selection can be attained by employing fast and user-transparent mechanisms. However, the effective usage of the cloud services requires publishing of services from the cloud service provider side and discovery and selection of services from the mobile user side. The current ways of publishing the services are based on the stateless Web services that are complex and time-intensive services [58]. Moreover, the mobile user with resources-constrained devices do not have sufficient capabilities for efficient cloud discovery and automatic services' selection. Such problems hinder smooth access and usage of the services. Seamless cloud server discovery is imperative to be attained to reduce the pre-execution delay for realizing the seamless application execution in the MCC.

7.1.2. Optimal Cloud Execution

The application execution in the cloud is also required to be optimized to realize the vision of the seamless application execution in the MCC [30]. The cloud-based application execution requires optimizing the tasks scheduling [59], minimizing the turnaround time by leveraging parallel processing support in the cloud while satisfying the user expectations. The scheduling of tasks can be optimized either by prioritizing the tasks execution according to time-sensitive nature of tasks or by considering tasks heterogeneity.

7.1.3. Optimal Component Deployment

A cloud-based mobile application usually comprises of more than one components and the nature of the components varies based on the functionalities and characteristics. The component of an application can be a user interface that is usually involved in the input so it is required to be deployed on the mobile device [5]. The components of an application that interact with the mobile device hardware are also required to run on the mobile device so that interactivity time is reduced [4]. The real-time compute-intensive components are required to be deployed in the cloudlet; whereas, non real-time compute-intensive components are deployed in the cloud [39]. The components must be deployed in such a manner that the interdependency among non-collocated components is minimized [27]. If a user has security concerns about the offloading of data that is required by a component, then the component that uses the data must run only on the device and must never be offloaded [27]. Such component deployment principles are essential to be followed by the application and the CMAEF designers to realize the vision of the seamless application execution in the MCC.

7.2. Adaptiveness

The CMAEFs are required to be adaptive to continue the application method execution locally on the mobile device, if network connection is lost or the other failure occurs [4, 30]. The frame-

works must implement adequate and accurate change detection mechanism to detect the changes in the operating environment and quality-aware opportunistic response mechanisms to adapt the framework in response to the changes in the operating environment. The adequacy and accuracy can be ensured by employing fast, accurate, user transparent, and low overhead techniques for change detection. On the other hand, the quality-aware opportunistic response mechanism can be implemented by employing non-greedy opportunistic techniques for using the available resources and by implementing peaceful quality degradation of the execution when network connectivity is lost or other failure occurs.

7.3. Autonomy

The seamless application execution also requires minimal human interaction to access the remote services and to offload the application in the cloud. Automated service access is based on providing the automated application partitioning, automated offloading, and cloud-based automated service provisioning. All of the decision algorithms of partitioning, offloading, and service provisioning are also required to be fast and lightweight to enable the seamless application execution in the MCC [34, 38]. The automatic application partitioning and offloading do not require any input from user for partitioning and offloading an application at run-time but takes decision by themselves. The cloud-based automated service provisioning can be implemented into three steps: (a) developing an application performance prediction model to predict the application resource requirements, (b) periodically determining the future demands and resource requirements using the prediction model to fulfil the application requirements, and (c) allocating resources according to the predicted resource requirements. Despite complexities involved in all of the three phases of the automated service provisioning, the automated service access process is essential for attaining the seamless application execution in the MCC.

7.4. Minimal Data Exchange

The amount of transferred data not only affects the cost and mobile battery power consumption but it also influences the time required by a node to transmit the data on a wireless link. The transmission time is directly proportional to data size and inversely proportional to bandwidth of the link [60]. More specifically, delay increases with the increase in the size of data and decreases with the increase in the bandwidth of the link. However, there exists a trade-off between the bandwidth and cost [61]. The transmission time T is computed by $T = S \ / \ B$, where S is the transferred data size and B is the bandwidth of the link. The amount of data exchanged is required to be minimized to attain the seamless application execution in the MCC [32]. The data can be further classified into three categories: (a) configuration data, such as states information, (b) input data, and (c) OS image and application migration data. The amount of the configuration data transferred can be reduced by only moving the essential configuration data. The size of input data can be reduced by sending non-duplicated input data and the amount of OS image and application migration data can be reduced by pre-installations.

7.5. Seamless Computational Resources Handoff

During the execution of an application in the cloud, a mobile user may move from one cloud to another cloud. The mobile user may not be able to access the previous cloudlet from outside due to the security policies implemented in the previously visited WLAN. The delay involved in accessing the application components running in the previously visited clouldet can be relatively higher than that by executing the application components in the newly visited cloudlet. This causes the triggering of the computational resources handoff. The computational resources handoff occurs when mobile user changes its application executing cloud during the execution of an application. The application migration can be either for enabling access from newly visited place or sustaining the application performance. The computational resources handoff involves two factors: (a) handoff criteria and (b) handoff process. The computational resources handoff criteria is required to be context-aware that can incorporate differences in the charging policy, difference in the mobile device-to-cloud delay, workload, or inaccessibility of previous cloud due to security policies. The seamless computational resources handoff can be attained by employing agile and user transparent mechanisms [7].

7.6. Agile Security and Privacy

The security and privacy is an important concern in the MCC that impedes the vast deployment of the clouds. The agile security and privacy mechanisms are essential to enable the seamless application execution in the distributed environment of the MCC. The agility can be attained by employing the simple, lightweight, and user-transparent techniques [7]. The authentication and authorization of resources must involve minimal user interaction to authenticate the user and to provide an access control.

Apart from the above mentioned principles, the pre-execution delay is required to be minimized by pro-actively installing the run-time environment, particularly the Java Run-time Environment (JRE) for attaining the seamless application execution in the MCC [34]. It is also vital to be aware of the communication channel quality and provisioning of the QoS guarantee for the smooth collaboration of mobile devices and cloud servers [32]. Based on our survey, we deduce that the seamless application execution realization is solely based on attaining the non-perceivable disruption. However, the aim of attaining the non-perceivable disruption is impeded by time-intensive cloud server discovery process [62], big data transmissions [63], time- and compute-intensive application partitioning [4, 5, 27, 42] and offloading process [21, 42], complex authentication, authorization and accounting process [64], and deteriorated by performance differences in heterogeneous wireless communication technologies. The disruption duration in application execution can be minimized by opting the presented design principles for designing of the applications and the CMAEFs.

8. Open Challenges for Seamless Application Execution in the MCC

In this Section, we highlight some of the most important challenges in realizing the vision of the seamless application execution in the MCC. The discussion on the research challenges provides

Table 8: Open Challenges and Guidelines to Address Them

Challenges	Guidelines
User-transparent Cloud Discovery	a) Proactive server discovery b) Service discovery algorithms reported in [65, 66] for the peer-to-peer networks can be used in designing of user-transparent cloud discovery
Unobtrusive Application Offloading	By designing an automated, lightweight offloading decision system with the objective to optimize the application response time.
Optimal Live VM Migration	The research efforts made by researchers in similar domain as reported in [67] can be guidelines.
Seamless Computational Resources Handoff	The research efforts in emerging seamless handoff and mobility management solutions in wireless networks, such as one reported in [68] can be used in designing seamless computational resources handoff mechanisms.
Agile Security and Privacy Mechanism	By designing a lightweight automatically reconfigurable security mechanism similar to one as reported for 3G/4G mobile services in [69]

research directions to the domain researchers for further investigations and improvements in the MCC. Table 8 presents a list of open challenges and guidelines for each of the open challenges.

8.1. User-transparent Cloud Discovery

A mobile device must discover the cloud to access and utilize the services provided by the cloud service provider. Although the cloud service discovery is discussed in [62], no one has investigated the problem of the delay involved in discovering and selecting the cloud services. Existing CMAEFs also do not fully address the problem of cloud discovery and service selection. The cloud discovery process is required to be crisp and user-transparent to realize the vision of seamless application execution. The user-transparency of the cloud discovery can be attained by employing automated resource specifying procedures, instantaneous resource discovery mechanisms, and the self resource-selection process. Moreover, constricting cloud resource discovery delay towards crisp response time is still an unresolved issue. The delay involved in discovering the cloud server can be reduced by pro-actively finding the server. However, the proactive server discovery consumes processor cycles and battery power of mobile device and network resources if the service is not used later. The service discovery solutions, such as the ones reported in [65, 66] for the peer-to-peer networks can be used in effective design and implementation of user-transparent cloud discovery in the MCC.

8.2. Unobtrusive Application Offloading

The unobtrusive offloading of application refers to the automated lightweight offloading process with the focus on reducing the offloading time and user inference. The operational environment in the MCC is highly dynamic in nature. The common causes are varying connection status, bandwidth, and attenuation distortion. Because of dynamic environmental conditions, an unobtrusive offloading of application is vital to sustain the usability of real-time applications and to provide the enhanced QoE to the end user.

However, the dynamic environment and run-time delay-inducing partitioning and offloading algorithms make it non-trivial to realize the vision of seamless application execution. The unobtrusive

offloading of applications is still an open research challenge due to the dynamic environment and complexities involved in offloading decision making. The unobtrusiveness in offloading process can be attained by employing automated, lightweight offloading decision systems with the objective to minimize the application response time.

8.3. Optimal Live VM Migration

Optimal live VM migration between cloud-based servers is vital in realizing the vision of seamless application execution in the MCC, considering intermittent wireless network bandwidth and mobility limitations. The problem is more critical when the mobile device is leveraging the local available resources. When a mobile user moves to a place far from the offloaded application, the increase in distance induces the latency and degrades the user's QoE. Therefore, migrating the live VM along with the mobile service consumer without affecting the end-user's service becomes vital to alleviate QoE degradation. The optimal live VM migration can be achieved by employing automated low cost migration procedures with the objective of minimal downtime. However, the live VM migration in a seamless manner is still an open challenge due to intrinsic limitations of wireless technologies and huge overhead of VM migration that cause the significant delay and disruption in execution process. The optimal live VM migration challenges can be addressed by migrating only the live states of VM that reduces the size of data exchange at run-time.

The research efforts already done by researchers in similar domain [67] can be guidelines for optimizing the Live VM migration for CMAEFs in MCC. After successful migration and deployment of VM, it is required to ensure the user-transparent access of migrated VM via initial IP address in the case of VM migration to new physical machine. Further efforts similar to [70] are required to realize the vision of seamless access to migrated VM in MCC.

8.4. Seamless Computational Resources Handoff

Seamless computational resources handoff is an important concern to ensure smooth migration among varied cloud services when mobile service consumer is on the move. The unmanaged mobility in wireless environment causes communication disruption when mobile device moves across two different communication coverage areas [71]. In MCC, if a mobile device moves farther from associated cloud server, then the application performance is surely degraded due to the increase in the communication latency. The problem is more critical when the services are provided by local cloudlets; since the local service provider do not allow the mobile user to access the local resources from outside of the network because of security reasons. This situation results in large number of computation resources handoffs.

The seamless computational resources handoff mechanisms aim to discover the available resources and migrate the application across two clouds/cloudlets in a seamless manner. The seamless computational resources handoff requires user-transparent resources discovery, automated handoff process, and non-perceivable disruption during the execution.

However, the realization of the seamless computational resources handoff is a challenging task because of inherited limitations of wireless technologies and limited resources of the mobile device.

The seamless computational resources handoff is an essential research problem that needs effective solutions for enabling the seamless application execution. The research efforts in emerging seamless handoff and mobility management solutions in wireless networks, such as one reported in [68] can be used in designing effective seamless mechanisms for computational resources handoff in MCC environment.

8.5. Agile Security and Privacy Mechanisms

Security and privacy are important concerns that impede successful deployment of clouds across the Internet [49]. A number of frameworks emphasis the need of security and privacy but very few of them actually implement the security solutions. The application execution has to ship the data from mobile device to cloud server. During the transport, the data is exposed through security breaches to intruders. To ensure the security and privacy, sufficient provisions should be provided. Nevertheless, the complexity of the problem and diversity of the environment make the designing of reliable effective lightweight security solution a challenging research perspective.

To perform the seamless application execution in the MCC, security and privacy mechanisms are required to be agile to mitigate the intolerable delay caused by security provisioning [7]. The agile security and privacy mechanisms can be implemented by employing quick and cost-effective trust establishment, by designing efficient security algorithms, and low-overhead encryption mechanisms.

Several solutions exist to provide security, such as hardware-based secure execution and steganography [72]. The limitations of these existing security mechanisms, such as large encryption key size and dramatic increase in amount of data impede the practical realization of security solutions for seamless application execution in MCC. The research efforts, such as the one reported in [69] can be used to effectively design the agile security mechanisms for the MCC.

9. Conclusions

The dramatically growing cloud-based mobile applications exhibit the dependency of mobile user on the cloud resources to enhance the resource potential of the mobile devices. The process of augmenting the resource potential of the mobile devices by leveraging on the cloud resources is obstructed due to the complexities involved in the CMAEFs, intrinsic limitations of the mobile devices, and wireless technologies. Such limitations make the application execution process non-continuous and obtrusive; thereby, failing to provide the seamless experience of application execution to the end-user. The seamless application execution in the MCC is vitally important for the usability and sustainability of real-time applications, such as disaster recovery systems, m-heath, m-guide, and m-learning. The ultimate goal of attaining the vision of the seamless application execution is to improve the application performance; thereby, providing high QoE to an end user. However, several wireless communication and networking challenges, particularly interference, signal attenuation, mobility, and handover impede the efforts in realizing the vision of seamless application execution.

In this paper, we have defined the seamless application execution. We surveyed the state-of-the-art CMAEFs and identified the seamless application execution enabling approaches employed

by the CMAEFs. We also developed a taxonomy of the seamless application execution enabling approaches. The thematic taxonomy of seamless enabling approaches may help mobile cloud application developers and cloud service providers to leverage on the appropriate features that mitigate communication and computation latencies when developing applications or providing services to increase the QoS for mobile-cloud users. Moreover, we investigated advantages and disadvantages of the seamless application execution enabling approaches and highlighted the strengths and weaknesses of the current CMAEFs by using thematic taxonomy of seamless application execution enabling approaches. Furthermore, we identified the principles for designing the seamless application execution frameworks from the surveyed state-of-the-art CMAEFs. The identified principles guide the framework designers to incorporate the specific features for enabling the seamless application execution. We also discussed open research challenges in realizing the vision of seamless application execution in MCC for future research.

10. Acknowledgements

This work is supported in part by the Malaysian Ministry of Higher Education, as the University of Malaya High Impact Research Grant (UM.C/625/1/HIR/MOE/FCSIT/03) and by the Bright Spark Unit, University of Malaya, Malaysia.

[1] S. Chen, M. Lin, H. Zhang, Research of mobile learning system based on cloud computing, in: Proceedings of 2011 International Conference on e-Education, Entertainment and e-Management (ICEEE'2011), Bali, Indonesia, IEEE, 2011, pp. 121–123.

[2] W. Cai, V. C. L. Leung, M. Chen, Next generation mobile cloud gaming, in: Proceedings of IEEE 7th International Symposium on Service Oriented System Engineering (SOSE'13), San Francisco Bay, USA, IEEE, 2013, pp. 551–560. doi:10.1109/SOSE.2013.30.

[3] A. Bourouis, A. Zerdazi, M. Feham, A. Bouchachia, M-health: Skin disease analysis system using smartphone's camera, Procedia Computer Science 19 (2013) 1116–1120.

[4] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, P. Bahl, MAUI: making smartphones last longer with code offload, in: Proceedings of the 8th international conference on Mobile systems, applications, and services (MobiSys'10), San Francisco, CA, USA., ACM, 2010, pp. 49–62.

[5] I. Giurgiu, O. Riva, D. Juric, I. Krivulev, G. Alonso, Calling the cloud: Enabling mobile phones as interfaces to cloud applications, in: Proceedings of the 10th ACM/IFIP/USENIX International Conference on Middleware (Middleware'09), Champaign, IL, USA, Springer, 2009, pp. 1–20.

[6] G. Huerta-Canepa, D. Lee, A virtual cloud computing provider for mobile devices, in: Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond, (MCS'10), San Francisco, CA, USA, ACM, 2010, pp. 1–5.

[7] M. Satyanarayanan, P. Bahl, R. Caceres, N. Davies, The case for VM-based cloudlets in mobile computing, Pervasive Computing, IEEE 8 (4) (2009) 14–23.

[8] S. Goyal, J. Carter, A lightweight secure cyber foraging infrastructure for resource-constrained devices, in: Proceedings of 6th IEEE Workshop on Mobile Computing Systems and Applications, (WMCSA'04), English Lake District, UK, IEEE, 2004, pp. 186–195.

[9] E. Ahmed, S. Khan, I. Yaqoob, A. Gani, F. Salim, Multi-objective optimization model for seamless application execution in mobile cloud computing, in: 5th International Conference on Information Communication Technologies (ICICT'13), 2013, pp. 1–6. doi:10.1109/ICICT.2013.6732790.

[10] M.-K. Tsai, Y.-C. Lee, C.-H. Lu, M.-H. Chen, T.-Y. Chou, N.-J. Yau, Integrating geographical information and augmented reality techniques for mobile escape guidelines on nuclear accident sites, Journal of environmental radioactivity 109 (2012) 36–44.

[11] E. Ahmed, A. Gani, S. Abolfezlei, L. Yao, S. Khan, Channel assignment algorithms in cognitive radio networks: Taxonomy, open issues, and challenges, IEEE Communications Surveys Tutorials, in press, 2014,doi:10.1109/COMST.2014.2363082.

[12] E. Ahmed, J. Qadir, A. Baig, High-throughput transmission-quality-aware broadcast routing in cognitive radio networks, Wireless Networks, in press, 2014.

[13] E. Ahmed, M. Shiraz, A. Gani, Spectrum-aware distributed channel assignment for cognitive radio wireless mesh networks, Malaysian Journal of Computer Science 26 (3) (2013) 232–250.

[14] N. Fernando, S. W. Loke, W. Rahayu, Mobile cloud computing: A survey, Future Generation Computer Systems 29 (1) (2013) 84–106.

[15] H. T. Dinh, C. Lee, D. Niyato, P. Wang, A survey of mobile cloud computing: architecture, applications, and approaches, Wireless Communications and Mobile Computing, in press 13 (18) (2013) 1587–1611.

[16] K. Kumar, J. Liu, Y.-H. Lu, B. Bhargava, A survey of computation offloading for mobile systems, Mobile Networks and Applications 18 (1) (2013) 129–140.

[17] A. u. R. Khan, M. Othman, S. A. Madani, S. U. Khan, A survey of mobile cloud computing application models, IEEE Communications Surveys & Tutorials 16 (1) (2014) 393–413. doi:10.1109/SURV.2013.062613.00160.

[18] S. Abolfezle, Z. Sinaei, E. Ahmed, A. Gani, R. Buyya, Cloud-based augmentation for mobile devices: Motivation, taxonomies, and open challenges, IEEE Communications Surveys & Tutorials 16 (1) (2014) 337–368. doi:10.1109/SURV.2013.070813.00285.

[19] J. Shin, Y. Kim, W. Park, C. Park, DFCloud: A TPM-based secure data access control method of cloud storage in mobile devices, in: Proceedings of IEEE 4th International Conference on Cloud Computing Technology and Science, (CloudCom'12), Taipei, Taiwan, IEEE, 2012, pp. 551–556.

[20] B.-G. Chun, P. Maniatis, Augmented smartphone applications through clone cloud execution, in: Proceedings of the 8th Workshop on Hot Topics in Operating Systems (HotOS'09), Monte Verit, Switzerland, 2009, pp. 8–14.

[21] T. Verbelen, P. Simoens, F. De Turck, B. Dhoedt, AIOLOS: Middleware for improving mobile application performance through cyber foraging, Journal of Systems and Software 85 (11) (2012) 2629–2639.

[22] M. Satyanarayanan, M. A. Kozuch, C. J. Helfrich, D. R. O'Hallaron, Towards seamless mobility on pervasive hardware, Pervasive and Mobile Computing 1 (2) (2005) 157–189.

[23] P. Vidales, Seamless mobility in 4G systems, Ph.D. thesis, Computer Science Cambridge: Cambrigde University (2005).

[24] C. Janssen, Seamless interface (Accessed on: 25th August 2013).
URL `http://www.techopedia.com/definition/16444/seamless-interface`

[25] N. Tolia, D. G. Andersen, M. Satyanarayanan, Quantifying interactive user experience on thin clients, Computer 39 (3) (2006) 46–52. doi:10.1109/MC.2006.101.

[26] M. Satyanarayanan, Mobile computing: the next decade, ACM SIGMOBILE Mobile Computing and Communications Review 15 (2) (2011) 2–10.

[27] X. Zhang, S. Jeong, A. Kunjithapatham, S. Gibbs, Towards an elastic application model for augmenting computing capabilities of mobile platforms, Mobile wireless middleware, operating systems, and applications 48 (2010) 161–174.

[28] H. Mehendale, A. Paranjpe, S. Vempala, Lifenet: a flexible ad hoc networking solution for transient environments, in: Proceedings of the ACM SIGCOMM conference (SIGCOMM'11), Toronto, On, Canada, ACM, 2011, pp. 446–447.

[29] L. Yang, J. Cao, Y. Yuan, T. Li, A. Han, A. Chan, A framework for partitioning and execution of data stream applications in mobile cloud computing, in: Proceedings of IEEE 5th International Conference on Cloud Computing, (CLOUD'12), Honolulu, Hawaii, USA, IEEE, 2012, pp. 794–802.

[30] S. Kosta, A. Aucinas, P. Hui, R. Mortier, X. Zhang, Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading, in: Proceedings of 31st IEEE

International Conference on Computer Communications (INFOCOM'12), Orlando, Florida, USA, IEEE, 2012, pp. 945–953.

[31] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, A. Patti, Clonecloud: elastic execution between mobile device and cloud, in: Proceedings of the 6th EuroSys conference on Computer systems (EUROSYS'11), Salzburg, Austria, 2011, pp. 301–314.

[32] S.-H. Hung, C.-S. Shih, J.-P. Shieh, C.-P. Lee, Y.-H. Huang, Executing mobile applications on the cloud: framework and issues, Computers and Mathematics with Applications 63 (2) (2011) 573–587.

[33] T. Verbelen, T. Stevens, P. Simoens, F. De Turck, B. Dhoedt, Dynamic deployment and quality adaptation for mobile augmented reality applications, Journal of Systems and Software 84 (11) (2011) 1871–1882.

[34] B.-D. Lee, A framework for seamless execution of mobile applications in the cloud, Recent Advances in Computer Science and Information Engineering 126 (2012) 145–153.

[35] L. Xiang, S. Ye, Y. Feng, B. Li, B. Li, Ready, set, go: Coalesced offloading from mobile devices to the cloud, in: Proc. of IEEE INFOCOM, 2014, 2014, pp. 2373–2381. doi:10.1109/INFOCOM.2014.6848182.

[36] B. Zhao, Z. Xu, C. Chi, S. Zhu, G. Cao, Mirroring smartphones for good: A feasibility study, Mobile and Ubiquitous Systems: Computing, Networking, and Services 73 (2012) 26–38.

[37] D. Fesehaye, Y. Gao, K. Nahrstedt, G. Wang, Impact of cloudlets on interactive mobile cloud applications, in: Proceedings of 16th International Enterprise Distributed Object Computing Conference (EDOC'12), Beijing, China, IEEE, 2012, pp. 123–132.

[38] D. Kovachev, Y. Cao, R. Klamma, Augmenting pervasive environments with an XMPP-based mobile cloud middleware, in: Proceedings of Fourth International Conference on Mobile Computing, Applications and Services (MobiCASE'12), Seattle, Washington, United States, Springer, 2012, pp. 361–372.

[39] T. Verbelen, P. Simoens, F. De Turck, B. Dhoedt, Cloudlets: Bringing the cloud to the mobile user, in: Proceedings of the third ACM workshop on Mobile cloud computing and services, (MCS'12), New York, USA, ACM, 2012, pp. 29–36.

[40] M. S. Gordon, D. A. Jamshidi, S. Mahlke, Z. M. Mao, X. Chen, Comet: code offload by migrating execution transparently, in: Proceedings of the 10th USENIX conference on Operating Systems Design and Implementation, (OSDI'12), Hollywood, CA., Vol. 12, 2012, pp. 93–106.

[41] E. Koukoumidis, D. Lymberopoulos, K. Strauss, J. Liu, D. Burger, Pocket cloudlets, ACM SIGPLAN Notices 47 (4) (2012) 171–184.

[42] D. Kovachev, T. Yu, R. Klamma, Adaptive computation offloading from mobile devices into the cloud, in: Proceedings of 10th International Symposium on Parallel and Distributed Processing with Applications (ISPA'12), Madrid, Spain, IEEE, 2012, pp. 784–791.

[43] E. E. Marinelli, Hyrax: Cloud computing on mobile devices using mapreduce, Tech. rep., DTIC Document (2009).

[44] T. White, Hadoop: the definitive guide, O'Reilly, 2012.

[45] X. Chen, Decentralized computation offloading game for mobile cloud computing (2014). doi:10.1109/TPDS.2014.2316834.

[46] S. Chen, Y. Wang, M. Pedram, A semi-markovian decision process based control method for offloading tasks from mobile devices to the cloud, in: GLOBECOM, 2013, pp. 2885–2890.

[47] H. Mao, N. Xiao, W. Shi, Y. Lu, Wukong: A cloud-oriented file service for mobile internet devices, Journal of Parallel and Distributed Computing 72 (2011) 171–184.

[48] Z. Xiao, Y. Xiao, Security and privacy in cloud computing, IEEE Communications Surveys & Tutorials 15 (2) (2013) 843–859. doi:10.1109/SURV.2012.060912.00182.

[49] A. N. Khan, M. Mat Kiah, S. U. Khan, S. A. Madani, Towards secure mobile cloud computing: A survey, Future Generation Computer Systems 29 (5) (2012) 1278–1299.

[50] M. Sookhak, H. Talebian, E. Ahmed, A. Gani, M. K. Khan, A review on remote data auditing in single cloud server: Taxonomy and open issues, Journal of Network and Computer Applications 43 (2014) 121–141.

[51] M. O. Spata, S. Rinaudo, Virtual machine migration through an intelligent mobile agents system for a cloud grid, Journal of Convergence Information Technology, 6(6), (2011).

[52] B.-G. Chun, P. Maniatis, Dynamically partitioning applications between weak devices and clouds, in: Proc. of the 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond, (MCS'10), San Francisco, USA, ACM, 2010, pp. 1–5.

[53] E. Ahmed, A. Akhunzada, M. Whaiduzzaman, A. Gani, S. H. Ab Hamid, R. Buyya, Network-centric performance analysis of runtime application migration in mobile cloud computing, Simulation Modelling Practice and Theory, 50, 2015, 42–56.

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

[55] M. Shiraz, E. Ahmed, A. Gani, Q. Han, Investigation on runtime partitioning of elastic mobile applications for mobile cloud computing, The Journal of Supercomputing, in press (2013) 1–20.

[56] S. Ou, K. Yang, J. Zhang, An effective offloading middleware for pervasive services on mobile devices, Pervasive and Mobile Computing 3 (4) (2007) 362–385.

[57] S. Simanta, K. Ha, G. Lewis, E. Morris, M. Satyanarayanan, A reference architecture for mobile code offload in hostile environments, in: Mobile Computing, Applications, and Services, Vol. 110, Springer, 2013, pp. 274–293.

[58] S. Wang, Z. Zheng, Q. Sun, H. Zou, F. Yang, Cloud model for service selection, in: Proceedings of IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS'11), 2011, pp. 666–671. doi:10.1109/INFCOMW.2011.5928896.

[59] E. N. Alkhanak, S. P. Lee, S. U. R. Khan, Cost-aware challenges for workflow scheduling approaches in cloud computing environments: Taxonomy and opportunities, Future Generation Computer Systems, in press, 2015.

[60] M.-F. Tsai, N. Chilamkurti, J. Park, C.-K. Shieh, Multi-path transmission control scheme combining bandwidth aggregation and packet scheduling for real-time streaming in multi-path environment, IET Communications 4 (8) (2010) 937–945.

[61] S. Akhlaghi, A. Kiani, M. R. Ghanavati, Cost-bandwidth tradeoff in distributed storage systems, Computer Communications 33 (17) (2010) 2105–2115.

[62] A. Goscinski, M. Brock, Toward dynamic and attribute based publication, discovery and selection for cloud computing, Future Generation Computer Systems 26 (7) (2010) 947–970.

[63] A. ODriscolla, J. Daugelaite, R. D. Sleator, Big data, hadoop and cloud computing in genomics, Journal of Biomedical Informatics, 46(5), 2013.

[64] F. Tusa, A. Celesti, R. Mikkilineni, AAA in a cloud-based virtual dime network architecture (dna), in: Proceedings of 20th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'11) , Paris, France, IEEE, 2011, pp. 110–115.

[65] G. Di Modica, O. Tomarchio, L. Vita, Resource and service discovery in SOAs: A P2P oriented semantic approach, International Journal of Applied Mathematics and Computer Science 21 (2) (2011) 285–294.

[66] C. Bertolli, D. Buono, G. Mencagli, M. Torquati, M. Vanneschi, M. Mordacchini, F. M. Nardini, Resource discovery support for time-critical adaptive applications, in: Proceedings of the 6th International Wireless Communications and Mobile Computing Conference (IWCMC '10), Caen, France, ACM, 2010, pp. 504–508.

[67] K. Takahashi, K. Sasada, T. Hirofuchi, A fast virtual machine storage migration technique using data deduplication, in: Proceedings of The Third International Conference on Cloud Computing, GRIDs, and Virtualization (CLOUD COMPUTING'12), Nice, France, 2012, pp. 57–64.

[68] S. Mitra, Seamless mobility management and QoS support for multihomed mobile node in heterogeneous wireless networks, in: Proc. of International Conference on Industrial and Information Systems (ICIIS'10), Dalian, China, 2010, pp. 145–150. doi:10.1109/ICIINFS.2010.5578717.

[69] J. Al-Muhtadi, D. Mickunas, R. Campbell, A lightweight reconfigurable security mechanism for 3G/4G mobile devices, IEEE Wireless Communications 9 (2) (2002) 60–65. doi:10.1109/MWC.2002.998526.

[70] P. Raad, G. Colombo, D. P. Chi, S. Secci, A. Cianfrani, P. Gallard, G. Pujolle, Achieving sub-second downtimes in internet-wide virtual machine live migrations in LISP networks, in: Proceedings of IFIP/IEEE International Symposium on Integrated Network Management (IM'13), Ghent, Belgium, 2013, pp. 286–293.

[71] M. Zekri, B. Jouaber, D. Zeghlache, A review on mobility management and vertical handover solutions over heterogeneous wireless networks, Computer Communications 35 (17) (2012) 2055–2068.

[72] J. Liu, K. Kumar, Y.-H. Lu, Tradeoff between energy savings and privacy protection in computation offloading, in: Proceedings of ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED'10), Austin, TX, USA, 2010, pp. 213–218.