# Mobile Cloud Computing:

## Resource Discovery, Session Connectivity and Other Open Issues

Markus Schüring
University of Twente
Enschede, the Netherlands
m.schuring@student.utwente.nl

Georgios Karagiannis
University of Twente
Enschede, the Netherlands
g.karagiannis@utwente.nl

***Abstract*—Cloud computing can be considered as a model that provides network access to a shared pool of resources, such as storage and computing power, which can be rapidly provisioned and released with minimal management effort.**

**This paper describes a research activity in the area of mobile cloud computing. It highlights different open issues which are associated with the mobile usage of cloud computing. By establishing a list of criteria for those issues, different solutions are compared against each other. The solutions discussed in this paper focus on different aspects of cloud computing in association with mobile usage. Each of the presented solutions offers at least one satisfactory approach for one of the open issues that are associated with the mobile usage of cloud computing resources. By combining the different existing approaches it would be possible to generate a solution that covers most of the issues currently identified.**

***Keywords-cloud computing; mobile devices; smart phones;***

## I. INTRODUCTION

Smart phones and other mobile devices are heavily used in today's world and still get even more important since the usage of mobile internet. The growth of the number of applications available for those devices in the last years has shown that there is a high demand for mobile applications [2]. However one common problem that all those devices share, still needs to be addressed: the limited capabilities of the devices regarding available resources like processor power, available memory and especially energy consumption.

A technology recently emerged in the IT industry offers an opportunity to solve those problems: Cloud computing (CC) gives its users the possibility to host and deliver services over the internet by dynamically providing computing resources [7].

Cloud computing eliminates the requirement for users to plan ahead for acquiring different resources, such as storage and computing power, and therefore, is attractive to business owners. Moreover, enterprises can provide resources depending on service demand. In particular, resources can be dynamically added and released depending on service demand and with minimal management effort.

The availability of cloud computing services in a mobile environment, also called mobile cloud computing, might thus be a possible solution for the earlier mentioned lack of resources of mobile devices. However research still needs to be done in order to solve several open issues like discovery of cloud computing resources, session connectivity, as well as possible frameworks to support cloud computing on mobile devices.

This paper delivers an insight into how cloud computing techniques can be used to support mobile devices and which open issues are associated with it. Moreover, this paper focuses on two of these open issues, i.e. mobility and resource discovery as well as mobility and session connectivity and shows how these open issues could effectively be solved.

The main research question of this paper is:

***Main research question:*** *Which open issues and possible solutions exist on using mobile cloud computing techniques to support smart phones and other resource-starved devices?*

To answer this question, it has been divided into three separate sub-questions, which will be answered throughout the research:

***RQ1:*** *Which cloud computing services can be used by smart phones and other resource-starved devices?*

***RQ2:*** *What are the open issues associated with mobile cloud computing techniques to support smart phones and other resource-starved devices?*

***RQ3:*** *Which solutions are addressing and efficiently solving the resource discovery and session connectivity open issues that are associated with mobile cloud computing techniques?*

Most of these research questions will be answered using a literature study. The third research question (RQ3) uses a combination of literature study, qualitative comparison and specification of new features to be integrated into mobility resource discovery and session connectivity procedures.

The remainder of the paper is organized as follows. Section II briefly describes the available cloud computing. Section III discusses several open issues that are associated with mobile cloud computing and their solutions. The discussion and comparison of the different solutions is presented in section IV. Section V concludes and provides recommendations for future activities.

## II. CLOUD COMPUTING SERVICES

The National Institute of Standards and Technology (NIST) defines cloud computing as "a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (…) that can be rapidly provisioned and released with minimal management effort or service provider interaction", from [4]. According to NIST, essential characteristics of cloud computing are:

- On-demand self-service
- Broad network access
- Resource pooling
- Rapid elasticity
- Measured service

On-demand self-service means that resources can be requested and released without human interaction at the service provider's side. Furthermore, those resources are accessible via standardized network protocols and are pooled at the service provider's site. The resources are dynamically assigned to the different users that want to make use of them, with the side effect, that the user does not exactly know where the resources he/she currently uses are located. The dynamic assignment of resources can be done manually or automatically, whereas the last offers the possibility to quickly react on resource demands and scale in or out accordingly. The usage of resources can be monitored to control and optimize it respectively. Besides those characteristics, cloud computing services can be classified into three different service models:

- Infrastructure as a service (IaaS)
- Platform as a service (PaaS)
- Software as a service (SaaS)

The models will be discussed in detail in the following sections.

### *A. Infrastructure as a service (IaaS)*

The first service model, which is called 'infrastructure as a service', is based on the provisioning of computing resources which are more hardware oriented. According to NIST the provisioning of "processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications", from [4] fall under this category. With infrastructure as a service, the user is able to run and manage own operating systems including applications by using virtualization technologies. Furthermore he can make use of storage systems and/or network devices like e.g. firewalls. Examples for this type of service model are Amazon EC2 for computation power and Amazon S3 for storage provisioning. With a view to mobile environments, this service model does not seem to be appropriate for mobile usage of cloud services, as it is highly focused on the provision of hardware based services with a low level of abstraction. It might only be interesting in the case of storage provisioning for mobile devices.

### *B. Platform as a service (PaaS)*

The second service model, which is called 'platform as a service', gives users the opportunity to run applications on the infrastructure offered by the service provider. However, it requires that the applications are created with programming languages or tools that are supported by the service provider. Examples for this type of service model are Google App Engine, Force.com and Microsoft Windows Azure. Looking at mobile usage of cloud computing services, this service model seems to be of interest, because it gives users the possibility to outsource applications or parts of them to the cloud. As a result, users can make use of the benefits a cloud computing system can offer them, including scalable and fast computation resources, which in the end could save time and energy.

### *C. Software as a service (SaaS)*

The third service model, named 'software as a service' focuses on the provisioning of applications. Examples for this type of service model are Google Docs, Microsoft Office Web Apps and Apple iWork.com. With regard to mobile usage, this service model might also be of interest, although it fully depends on a working network connection between mobile devices and the cloud system. However, the benefits of cloud systems, which might lead to savings in time and energy consumption, also apply here.

## III. OPEN ISSUES AND SOLUTIONS

The mobile usage of cloud computing services is still in the early stages of development and several open issues need to be addressed. With the mobility of users and their devices, several problems arise that need to be taken into account, when making use of cloud computing services on mobile devices.

**Mobility and resource discovery**

The first being, that cloud computing resources are widely spread around the globe and offer a lot of different services to their users. Mobile devices that want to make use of those resources should be able to automatically discover cloud computing resources that preferably are nearby their current location.

**Mobility and cloud session connectivity**

Another problem to face is the fact that mobile devices are not always connected to a network because of dead spots or other influences, resulting in unreachable cloud computing resources [2]. Related to that is the fact that connectivity with remote cloud computing resources can suddenly disappear [2]. Furthermore, network addresses of mobile devices can change over time, due to different regional address assignments, resulting in a need for topology-agnostic identification of connections.

**Program-technical characteristics**

Offloading of computation to remote resources is a technique that requires additional effort and skills of application developers, as applications possibly need to be adjusted accordingly [2]. To make full use of remote resources, it might for example be necessary that parts designed for remote execution need to be different from their local

complements, as they need to make use of parallelization, which also demands additional skills from developers.

**Overhead due to use of cloud**

Along with the remote execution of application parts comes the problem that those parts need to be transferred to the cloud resource first, before an execution can take place. The overhead produced by this transfer also needs to be taken into account, when dealing with computation offloading and possibly related time and energy savings.

**Reliability**

The ability of the cloud computing system to perform and maintain the provision of its resources under unexpected failures, of e.g., storage, network connectivity and computing power, for a predefined amount of time. This ability can be supported by e.g., (1) supporting replication of objects and services, (2) using redundant communication (more than one communication paths used for the dissemination of the same information), (3) using redundant processing (more than one processing entities used to process the same action).

**Scalability**

The ability of the cloud computing system to maintain its good system performance while supporting (1) an increasing amount of mobile users, (2) expanding the amount of resources and services to satisfy rapid increases in service demand. This ability can be satisfied by e.g., (1) support for massive sharing of content, (2) flexible, fault-tolerant and distributed data bases, (3) fast and consistent content replication support.

**High availability**

The ability of the cloud computing system to provide and support a large amount of different resources that are easily accessible and that are operating in optimal performance conditions for a predefined agreed amount of time.

**Security and privacy**

The ability of the cloud computing system to protect itself and its provided resources from security and privacy attacks. Different security and privacy aspects need to be considered when running foreign code on remote resources that maybe also used by several users at the same time [2]. The main security solutions are e.g., related to (1) data integrity, where the unauthorized modification of information incoming and outgoing the cloud should be detected, (2) confidentiality to secure the data access and transfer. The main privacy solutions should ensure that the identity of the cloud computing clients should not be revealed to unauthorized entities.

Due to paper page limitations, only the open issues subset that is the most relevant for mobile cloud computing is chosen for further investigation:

1. Mobility and resource discovery
2. Mobility and cloud session connectivity

## A. *Criteria*

The following criteria are associated with the open issues mentioned above.

### *1) Mobility and resource discovery*

The criteria associated with the open issue “mobility and resource discovery” are defined as follows:

**Naming and addressing of resources (RNA)**

The resources are using a naming/addressing method and/or structure. Three grades are used for rating this criterion: (1) Good: Supported; (2) Moderate: Supported but problematic; (3) Fair: Not supported.

**Dynamic discovery of cloud resources (RDD)**

The resources can be discovered using a dynamic method. Three grades are used for rating this criterion: (1) Good: Dynamic discovery; (2) Moderate: Manual discovery; (3) Fair: Static discovery.

**Latency of the resource discovery process (RDL)**

The amount of time consumed for the discovery of available cloud computing resources. Two grades are used for rating this criterion: (1) Good: Time experienced by an automated discovery process; (2) Fair: Time experienced by a manual discovery process.

### *2) Mobility and cloud session connectivity*

The criteria associated with the open issue “mobility and cloud session connectivity” are defined as follows:

**Handover support between resources (HSR)**

The ability to migrate from one cloud computing resource to another one. Two grades are used for rating this criterion: (1) Good: Supported; (2) Fair: Not supported.

**Latency of handover between resources (HLR)**

The amount of time used for the handover process from one resource to another. Three grades are used for rating this criterion: (1) Good: Seamless handover; (2) Moderate: Short interruption; (3) Fair: Long interruption.

**Data loss probability during handover (HDLP)**

The probability that data loss will occur during the handover from one resource to another. Two grades are used for rating this criterion: (1) Good: No data loss at all; (2) Fair: High chance for data loss.

**Roaming support for mobile devices (RS)**

The ability to support roaming of mobile clients through different wireless network technologies. The solution is considered to be efficient when:

(a) The identification of connections is topology-agnostic

(b) The solution supports a seamless handover procedure

Three grades are used for rating this criterion: (1) Good: (a) and (b) are supported; (2) Moderate: Either (a) or (b) is supported; (3) Fair: Neither of the two is supported.

## B. *Evaluation of solutions*

The evaluation of solutions and the degree in which the different solutions match the criteria is based on the documentation and resources available for the particular

solution. Table I gives an overview over the symbols used in the scoring tables and their meaning.

TABLE I. LEGEND FOR EVALUATION OF SOLUTIONS AND CRITERIA.

| Symbol | Description |
|---|---|
| + | Good |
| ø | Moderate |
| – | Fair |
| n/a | No information available |

The exact meanings of the ratings for each criterion have been defined in the preceding section.

For this evaluation, three main mobile cloud computing solutions are identified. These solutions are described in the following subsections.

*1) Cuckoo*

Cuckoo is a computation offloading framework for smartphones which, at the moment of writing this document, has been implemented for use with the Android operating system only [2].

Cuckoo has been designed with the fact in mind that cloud computing resources are not always available when used by mobile devices. Therefore its programming model supports both local and remote execution of application methods to keep applications working when cloud resources are not available. It supports different implementations of application methods for local and remote sites to establish the usage of features that may only be available in cloud resources (e.g. parallelization). Cuckoo has been implemented as a standard server/client model. The server can run on any resource, which has a Java virtual machine installed. Services (parts of an application) that are available on the client can be uploaded to the server and executed remotely afterwards. When a service has been initially uploaded to the server, it can be used afterwards without the need to upload it again.

The discovery of resources is done manually in the current prototype. As soon as a server is running, a QR code (short for "Quick Response", a two-dimensional barcode) is displayed, which can be scanned in by a client with its integrated camera. The resource (or more precisely its address) is then stored in the "Resource Manager", a part of the Cuckoo framework, which is responsible for the connection with resources. The resource can then be used to offload and execute parts of an application.

The process of deciding whether a part of an application should be executed local or remotely, is done by making use of heuristics, context information and history. At the moment of writing, these heuristics rather are very simple: remote execution is preferred over local execution, if remote resources are available.

Communication between server and clients is realized with the Ibis communication middleware, which abstracts and supports different networks like WiFi, Cellular and Bluetooth. The Ibis communication middleware is a portable, high performance, Java based library.

Due to the characteristics of mobile environments, connections to resources can be lost over time. Cuckoo handles these disconnects by switching to different resources and continuing execution on a new available resource. As a last fallback, execution of application parts can always be done locally on the mobile device.

The current resource discovery process of Cuckoo can be considered inefficient, because of the fact that for each new resource, a manual scanning of the resource's QR code is needed, see Table II. This takes up a large amount of time and is not a very scalable process. Cuckoo supports migration from one resource to another and has a local fallback that enables applications to remain working, when no cloud resources are available. The process of migration itself however is not described, so there is no evidence of correct working or information about the latency of this process. Due to the use of the Ibis communication middleware, it is possible to establish communication between mobile devices and cloud services over different networks.

*2) VOLARE*

VOLARE is a middleware-based solution, which provides context-aware adaptive cloud service discovery for mobile systems. It does this by monitoring resources and context of devices and dynamically adapts cloud service requests accordingly [6].

Operation takes place at two levels: service discovery time and runtime.

At service discovery time, VOLARE intercepts service requests from applications executed on a mobile device. According to the current context of the device, which may include hardware resources (e.g. battery consumption, CPU, memory usage), environmental variables (e.g. network bandwidth) and user preferences (e.g. low cost binding, low power operation) it then starts to process the request.

At runtime, VOLARE continuously monitors existing cloud bindings and the context of a device. If the context of the device or the provided service level of the cloud service changes, VOLARE reacts on this by searching for a service that matches the new requirements and initiates a rebinding.

The architecture of VOLARE consists of several independent modules. If an application requests a service, the request is intercepted by the service request module and forwarded to the adaption module. At the same time, the context of the device is continuously monitored by the context monitoring module and information about the current context is forwarded to the adaption module. The adaption module handles the service requests according to the current context of the device. If changes in the context occur, a re-evaluation of currently active services is triggered, possibly resulting in a rebinding according to the new context and QoS (Quality of Service) level. The fulfillment of QoS levels is monitored by the QoS monitoring module. If deviations are asserted, the service request module gets alerted, resulting in the initiation of a new discovery cycle. The service discovery and rebinding is done by the service binding module. It forwards the adapted

service request to a broker, which then chooses the best matching service provider.

According to [6], the architecture enables the use of VOLARE without modifications to the application itself. The definition of QoS levels takes place by providing an "adaption policy file", which is written in an own two level policy specification language, together with the application.

VOLARE uses brokers to access services in the cloud. The broker accepts a service request with a corresponding service level and tries to bind to the best matching service provider. VOLARE supports handover from one resource to another. According to [6], the "rebinding lasts an average of 0.963 seconds", from [6], which is considered a very low latency, see Table II. However, it should be mentioned that due to the fact that VOLARE has not focused on remote execution of applications, there was no need to migrate (application) data between resources. Methods for topology-agnostic identification of connections did not seem to be applied, as communication took place via standard TCP/IP protocols (the prototype used RTP/RTMP for video streaming).

*3) Mobile computation outsourcing framework*

Chonglei Mei [3] proposes a mobile computation outsourcing framework for the Android platform, which is implemented in Java and according to [3] it can be easily deployed on any backend platform. The framework consists of three main components, which are located at the cloud side: a proxy, a code repository and a server. The proxy server acts as a gateway between mobile devices and the cloud. It has access to a code repository, which contains popular code components that can be executed in the cloud. When a mobile device wants to offload computation to the cloud it contacts the proxy server with the name of a specific Java class component it wants to execute.

According to [3], three cases need to be differentiated: (1) the code might already been running on one of the servers, so the mobile device can make use of it immediately; (2) the code might also be available in the code repository, but has not been deployed on any of the servers, thus the proxy server can push the code to one of the servers and start it; (3) if the code is neither running nor available in the repository, the mobile device needs to upload it accordingly. Communication between mobile devices and servers, which execute the code, takes place directly.

For resource discovery, [3] proposes to place the proxies outside the cloud, so that they can run a multicast DNS (mDNS) to advertise their services to mobile devices in their network. This is done by broadcasting information about the offered service type and name to neighbors. The proxies could be placed at e.g. WiFi access points. If the proxies are placed at the cloud side, [3] proposes to associate a well-known URL with the proxy, so that devices can connect to it.

At the mobile device's side, a client is responsible for management of computation offloading. The client monitors available resources of the mobile device and creates a performance history of offloaded application components. Based on both data it then decides whether to offload computation to the cloud or not.

The Mobile Computation Offloading Framework uses DNS and especially mDNS to address cloud computing resources. Due to the large numbers of cloud computing resources this is considered inefficient, as the current DNS structure may become overloaded. Therefore, this solution is rated as moderate, see Table II. Discovery of resources is achieved by placing proxy servers close to network access points, which advertise cloud services via mDNS. This is considered to be good, as it offers a dynamic way to discover resources and gives users the possibility to be redirected to resources near their current location. Mechanisms for handover or roaming support could not be identified.

## IV. DISCUSSION

The solutions presented in this paper focus on different aspects of cloud computing in connection with mobile usage. Table II gives an overview over the different solutions and their rating.

The **Cuckoo** framework presents a solution to offload application parts to cloud computing resources. It handles possible connectivity problems with the help of local fallback procedures and offers mechanisms for handover support between resources.

**VOLARE** with its middleware approach offers context-aware adaptive cloud service discovery for mobile systems. Services are chosen based on the current context of the device and currently provided service levels are continuously monitored and compared against the requirements. If deviations are asserted, VOLARE reacts on it by searching for a more appropriate service and initiates a rebinding.

The **Mobile Computation Offloading Framework** proposed by Mei et al. (2011) offers an interesting approach on how to discover nearby cloud computing resources with the help of multicast DNS.

TABLE II. COMPARISON OF THE DIFFERENT SOLUTIONS.

| RNA | RDD | RDL | HSR | HLR | HDLP | RS |
|---|---|---|---|---|---|---|
| Cuckoo | | | | | | |
| n/a | ø | – | + | n/a | n/a | ø |
| VOLARE | | | | | | |
| n/a | + | + | + | + | n/a | ø |
| Mobile Computation Offloading Framework | | | | | | |
| ø | + | + | – | n/a | n/a | – |

Current cloud computing infrastructures tend to be designed for usage with wired or fixed devices mainly. However, due to the fast development of wireless broadband internet connections for mobile devices, this design has to be changed and mobile devices need to be considered as parts of those cloud computing infrastructures as well in the near future. Several open issues that are associated with the integration of those mobile devices into cloud computing infrastructures, are still not yet satisfactorily being addressed by current cloud computing solutions. One example is the

topology-agnostic identification of mobile devices, which tend to change their location (and therefore their current IP address) more often.

## V. PROPOSED SOLUTION

None of the discussed solutions can satisfy all the criteria related to mobility and resource discovery as well as mobility and cloud session connectivity in a proper way. Therefore, in this section we will introduce different approaches that can improve the solutions discussed above.

### *A. Naming and adressing of resources (RNA) / Roaming support for mobile devices (RS)*

A possible solution that can satisfy these criteria is described in the internet-draft "Locator/ID Separation Protocol (LISP)". LISP is a network-based protocol that enables addressing of devices independent from their topological location by separating IP addresses into two new numbering spaces: Endpoint Identifiers (EIDs) and Routing Locators (RLOCs) [1]. It uses EIDs to identify devices independently from the network topology and RLOCs, which are topologically assigned to network attachment points, for routing and forwarding of packets through the network.

LISP has been designed with simple and incremental deployment in mind, meaning that no changes to current host protocol stacks or core internet infrastructure are needed.

When a LISP enabled router receives a packet with a non-routable EID assigned to it (as destination address), it maps the EID to a routable RLOC and encapsulates the packet for further forwarding. The packet can then be forwarded to a LISP enabled router at the destination by non-LISP routers. The LISP enabled router at the destination decapsulates the packet and forwards it to the recipient specified by the EID.

With this approach it is possible to establish stable communication between devices independent of their current geographical and topological location and to solve the connectivity problems mentioned before.

### *B. Dynamic discovery of cloud resources (RDD)*

The Universal Description, Discovery, and Integration (UDDI) protocol is a standard developed by the Organization for the Advancement of Structured Information Standards (OASIS). UDDI defines a "standard method for publishing and discovering the network-based software components of a service-oriented architecture" [5]. It acts as a registry for web services on public or private networks by offering information about available services with the help of standards like e.g. XML.

Like web services offer some sort of service to their consumers, cloud computing services do so in the same way with resources. UDDI, or a modified version of UDDI more fitted to cloud computing might therefore be an interesting approach to establish a standardized way for the discovery of cloud computing resources.

## VI. CONCLUSION AND FUTURE WORK

In section II of this paper, an introduction into cloud computing and its different service models has been given in order to answer the first research question (RQ1), which was intended to show which cloud computing services can be used by smart phones and other resource starved devices. The three different service levels have been analyzed with regard to mobile usage, where IaaS with its hardware oriented approach only seems to be appropriate for storage provisioning. The two other service levels, namely PaaS and SaaS seem to be of more interest, as they offer the possibility to run whole applications or parts of them in the cloud.

In section III, several open issues that need to be taken into account, when making use of cloud computing services on mobile devices, have been described in order to answer the second research question (RQ2). Furthermore, different criteria have been established in order to evaluate solutions for those issues. This resulted in a description of currently developed solutions that (partially) address the open issues mentioned before and provided an answer to the last research question (RQ3).

Each of the presented solutions offers at least one satisfactory approach for one of the open issues that are associated with the mobile usage of cloud computing resources. By combining the different approaches and merging them into a common solution, it might be possible to provide a new solution that covers most of the open issues currently identified. Such a solution might have the possibility to finally make cloud computing usable on mobile devices, resulting in new and interesting usage scenarios and offering execution speedups and energy savings to mobile users.

## REFERENCES


[1] Farinacci, D., V. Fuller, D. Meyer, and D. Lewis, Locator/ID Separation Protocol (LISP). Internet Draft, July 9, 2011. Available online at: http://tools.ietf.org/id/draft-ietf-lisp-15.txt.

[2] Kemp, R., N. Palmer, T. Kielmann, and H. Bal, Cuckoo: a Computation Offloading Framework for Smartphones, in Proceedings of the Sixteenth annual conference of the Advanced School for Computing and Imaging 2010. 2010: Veldhoven, the Netherlands. p. 70-77.

[3] Mei, C., J. Shimek, C. Wang, A. Chandra, and J. Weissman, Dynamic Outsourcing Mobile Computation to the Cloud. 2011, Department of Computer Science and Engineering, University of Minnesota: Twin Cities.

[4] Mell, P. and T. Grance, NIST SP 800-145. The NIST Definition of Cloud Computing (Draft). Recommendations of the National Institute of Standards and Technology. 2011.

[5] Organization for the Advancement of Structured Information Standards (2004). "Introduction to UDDI: Important Features and Functional Concepts." Retrieved July 19th, 2011, from http://uddi.org/pubs/uddi-tech-wp.pdf.

[6] Papakos, P., L. Capra, and D.S. Rosenblum, VOLARE: context-aware adaptive cloud service discovery for mobile systems, in Proceedings of the 9th International Workshop on Adaptive and Reflective Middleware. 2010, ACM: Bangalore, India. p. 32-38.

[7] Zhang, Q., L. Cheng, and R. Boutaba, Cloud computing: state-of-the-art and research challenges. Journal of Internet Services and Applications, 2010. 1(1): p. 7-18.