Mobile cloud computing – open issues and solutions

Markus Schüring
University of Twente
P.O. Box 217, 7500AE Enschede
The Netherlands

m.schuring@student.utwente.nl

ABSTRACT
This paper describes a research in the area of mobile cloud computing. Cloud computing can be considered as a model that can provide network access to a shared pool of resources, such as storage and computing power, that can be rapidly provisioned and released with minimal management effort.

The solutions discussed in this paper focus on different aspects of cloud computing in connection with mobile usage. By combining the different approaches and merging them into a common solution, it might be possible to generate a new solution that covers most of the issues currently experienced. Such a solution might have the chance 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.

Keywords
Cloud computing, mobile devices, smart phones

1. 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 [10]. 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 [15].

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 and is an interesting topic. However research still needs to be done on several open issues like discovery of cloud computing resources, 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, which open issues exist and how those issues can be effectively 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 four separate sub-questions, which will be answered throughout the research.

First, an introduction is given on which cloud computing services are available and which of them can be used by mobile devices. This leads to the following first sub-question:

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

The question will be answered in section 2 (Cloud computing services).

The introduction on cloud computing services will be followed by an explanation of communication network architectures that can be used to support those services on mobile devices, which leads to the second sub-question:

RQ2: Which communication network architectures can be used in order to support cloud computing services on smart phones and other resource-starved devices?

This question will be answered in section 3 (Communication network architectures).

After explaining which communication network architectures can be used, an investigation on current open issues will be done. This leads to the third sub-question:

RQ3: What are the open issues associated with mobile cloud computing techniques to support smart phones and other resource-starved devices?

This question will be answered in section 4 (Open issues and solutions).

Finally, possible solutions that address and sufficiently solve the current open issues will be presented, which leads to the last sub-question:

RQ4: Which solutions are addressing and efficiently solving these open issues that are associated with mobile cloud computing techniques?

The last research question will also be answered in section 4 (Open issues and solutions).

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

15th Twente Student Conference on IT, June 20th, 2011, Enschede, The Netherlands.

Copyright 2011, University of Twente, Faculty of Electrical Engineering, Mathematics and Computer Science.
The models will be discussed in detail in the following sections. Finally, the remaining sources were used to create a consistent insight view in the chosen topic and to answer the earlier set-up research questions.

The literature study has been performed using proven scientific literature including peer reviewed publications in journals, proceedings and books from authors that are known to be an accredited authority on the topic that they are writing about. A research plan was created that contained the research questions and a survey on available literature was done from the sources mentioned above. The available literature was subsequently analyzed and filtered according to relevance and up-to-date information on the topic.

The qualitative comparison was done by identifying open issues, defining criteria for those issues and finally comparing possible solutions according to the criteria. The remainder of the paper is organized as follows. Section 2 briefly describes the available cloud computing services. Section 3 discusses several communication network architectures that can be used to support the mobile devices when accessing the cloud. The open issues and their solutions are described in section 4. The discussion and comparison of the different solutions is presented in section 5. Section 6 concludes and provides recommendations for future activities.

2. 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 [12].

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.

2.1 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 [12] 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.

The management of the underlying infrastructure is done by the service provider of the cloud, though the user has full control over operating systems, applications and storage and eventually partial control over network devices.

Examples for this type of service model are Amazon EC2 for computation power and Amazon S3 for storage provisioning.

Looking at mobile usage of cloud computing services, this service model might also 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.

2.2 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.

The management of the underlying infrastructure and operating systems is in the hands of the service provider. Though the user has full administrative control over the applications he wants to host on the cloud system.

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.

2.3 Software as a service (SaaS)

The third service model, named ‘software as a service’ focuses on the provisioning of applications.

The management of the underlying infrastructure, operating systems and even the configuration of the application itself (with exemption of some partial elements) is completely done by the service provider.

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.

3. COMMUNICATION NETWORK ARCHITECTURES

Several communication network architectures have been developed since the early beginning of the information age, with some of them especially designed for usage with mobile devices. Some of the most important ones will be presented in the following, as they pave the way for new services like mobile cloud computing to be used on those devices.


3.1 GSM, GPRS and EDGE
The Global System for Mobile communication (GSM) is an internationally accepted standard for digital cellular telephony developed by the European Telecommunications Standards Institute (ETSI) [9]. It was originally designed to operate in the 900 MHz band and got expanded later on to other frequencies as well (e.g. 1800 MHz), with a primary focus on voice telephony [7]. However, from the early beginning GSM offered the possibility for circuit-switched data connections with a data rate of up to 9600 bit/s, a value that was equal to fixed connections at that time (1989). With the development of broadband internet connections this bitrate soon could not cope with the bitrate of fixed connections anymore, resulting in an enhancement of the GSM standard in the year 2000. The General Packet Radio Service (GPRS) extended the circuit-switched data connections with packet-switched functionality and offered data rates up to 171 kbit/s by aggregating several GSM time slots [6]. In 2003 this bitrate got enhanced another time with the development of EDGE (Enhanced Data rates for Global Evolution), which increased data rates up to 384 kbit/s [5].

3.2 UMTS
In 2001 the first network based on the third generation of mobile cellular technology, developed by the 3rd Generation Partnership Project (3GPP), was launched [9]. The Universal Mobile Telecommunication System (UMTS) is based on wideband code division multiple access (W-CDMA) radio technology and offers greater spectral efficiency as well as higher bandwidth [8]. UMTS establishes data rates up to 384 kbit/s and was later on extended to provide up to 7.2 Mbit/s with the development of the High-Speed Downlink Packet Access (HSDPA) protocol.

3.3 LTE
Initiated in 2004, Long Term Evolution (LTE) is an enhancement of the UMTS standard, developed by the 3rd Generation Partnership Project (3GPP) and is handled as a successor of the 3G standard [1]. It is designed to increase capacity and bandwidth of mobile cellular networks. Some of the key features of LTE are its high spectral efficiency, as well as its very low latency and the possibility of variable bandwidths. LTE offers peak rates of at least 100 Mbit/s in downlink and 50 Mbit/s in uplink together with radio access network (RAN) round-trip rates of less than 10 ms [4, 13].

3.4 Wireless LAN
The IEEE 802.11 (also known as Wireless LAN or Wi-Fi) defines a set of standards for implementation of wireless local area networks (WLAN) and is published by the Institute of Electrical and Electronics Engineers. It is separated into different protocols, which operate in 2.4, 3.6 and 5 GHz frequency bands. The IEEE 802.11b protocol was the first being widely accepted and offered data rates up to 11 Mbit/s. Its successor, the IEEE 802.11g protocol increased available data rates up to 54 Mbit/s, while the recently introduced IEEE 802.11n protocol allows the bonding of streams with the use of multiple antennas, to reach a total data rate of up to 600 Mbit/s.

4. 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 [10]. Related to that is the fact that connectivity with remote cloud computing resources can suddenly disappear [10]. 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 [10]. 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 companions, 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 providing 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 expand the amount of resources and services to large scales 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 [10]. The main security
solutions are e.g., related to (1) data integrity, where the unmodified 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 time and 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
3. Overhead due to use of cloud

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

4.1.1 Mobility and resource discovery
The criteria associated with the open issue “mobility and resource discovery” are:

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) Bad: 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) Bad: 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) Bad: Time experienced by a manual discovery process.

4.1.2 Mobility and cloud session connectivity
The criteria associated with the open issue “mobility and cloud session connectivity” are:

Handover support between resources (HSR)
The ability to migrate from one resource to another one. Two grades are used for rating this criterion: (1) Good: Supported; (2) Bad: 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) Bad: Long interruption.

Data loss probability during handover (HDLP)
The probability that data loss will occur during handover from one resource to another. Two grades are used for rating this criterion: (1) Good: No data loss at all; (2) Bad: 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) Bad: Neither of the two is supported.

4.1.3 Overhead due to use of cloud
The criteria associated with the open issue “overhead due to use of cloud” are:

Energy usage (EU)
The amount of energy consumed. Three grades are used for rating this criterion: (1) Good: Low energy usage; (2) Moderate: Moderate energy usage; (3) Bad: High energy usage.

Overhead in signaling load due to cloud (OH)
The amount of overhead produced, where overhead is defined as:

(a) The number of requests between mobile devices and resources
(b) The amount of data traffic between mobile devices and resources

Three grades are used for rating this criterion: (1) Good: Both (a) and (b) are low; (2) Moderate: Either (a) or (b) is low; (3) Bad: Neither (a) nor (b) are low.

Data Latency imposed by the cloud (DL)
The amount of extra time needed for the data exchange with the cloud. Two grades are used for rating this criterion: (1) Good: Short interruption; (2) Bad: Long interruption.

4.2 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 4.1 gives an overview over the symbols used in the scoring tables and their meaning.

Table 4.1 Legend for evaluation of solutions and criteria.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>Good</td>
</tr>
<tr>
<td>ø</td>
<td>Moderate</td>
</tr>
<tr>
<td>-</td>
<td>Bad</td>
</tr>
<tr>
<td>n/a</td>
<td>No information available</td>
</tr>
</tbody>
</table>

The exact meanings of the ratings for each criterion have been defined in section 4.1 earlier.

4.2.1 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 [10].

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 furthermore 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 infinitely afterwards without the need to upload it again.
The discovery of resources is done manually at the moment. 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) then gets 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.

Cuckoo supports two modes of operation: early and late binding to remote resources. When set to early binding, Cuckoo uploads services to a remote resource, as soon as an application binds to a service, with as a result an improvement of speed in the way that a service is (likely) already available when the application wants to make use of it. If set to late binding, uploading of services takes place when a service is first used, avoiding unneeded overhead and energy usage.

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 5.1. 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. If configured accordingly it might also be possible to establish topology-agnostic connections (e.g. by using P2P implementations). Energy usage of Cuckoo is considered to be moderate, as cloud execution is always preferred over local execution, which might not always be the best solution and should be improved in the future. To avoid unnecessary overhead, applications initially need to be uploaded to the same remote resource one time only.

4.2.2 Cloud agency

The approach presented in this subsection is not really a mobile phone oriented one. However it has very much potential because of the way it handles mobility.

Cloud agency is an architecture proposed by Aversa et al., which combines three different technologies, namely GRID computing, cloud computing and mobile agents [2].

It offers “Virtual Clusters on the top of an existing GRID, that can be easily configured with the support of mobile agents based services”, from [2]. Cloud agency makes use of the benefits of each of the three technologies:

According to [2], it uses GRID as “a common, well-known platform, High Performance computational resource, together with its stable and well-accepted security infrastructure”, from [2]. Furthermore it makes use of cloud computing to distribute virtual clusters to end-users, giving them full administrative control over an easily scalable resource. To provide end users with a simple and flexible way to configure those virtual clusters, mobile agents are utilized.

A mobile agent is an application with the feature of autonomy, social ability, learning and mobility. It is able to move itself from one computer to another autonomously, by saving its state with its data intact and being able to continue execution in the new environment. With the help of mobile agents it is possible to move services through different virtual machines and thus through different cloud services.

Figure 4.1 The cloud agency architecture, based on: [2]

With the help of virtual cluster node images, cloud agency offers the possibility to create new virtual clusters that can be directly integrated into the current GRID infrastructure. The images contain all the software that is needed for this process. Such a virtual cluster consists of a front-end node and several computing nodes that exchange data on a private network. The front-end server acts as a proxy between the GRID and the mobile agent platform. When a cluster is made available, users can configure it by making use of mobile agents that can be deployed on the cluster with the help of a client that runs the Cloud agency software. The client queries a repository for available services and retrieves a services list from it. The user can then choose a service from the list, which will then be installed into the virtual cluster.

Due to the certified Globus container (GC) hosted on every virtual cluster and the participation in the underlying GRID, Cloud agency is able to discover available resources within the GRID infrastructure. The use of mobile agents offers the possibility to migrate services from one resource to another or to recover them. Due to the fact that Cloud agency is not focused on the usage with mobile phones, there was no information found in [2] about roaming support, energy usage or overhead available, see Table 5.1.

4.2.3 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 [14].
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, see Figure 4.2. 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.

According to the researchers, 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 [14], the “rebinding lasts an average of 0.963 seconds”, from [14], which is considered a very low latency, see Table 5.1. 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). Energy usage could not be metered, as the simulation took place on a mobile emulator. Overhead is considered as being good, as VOLARE continuously monitors the current device context (e.g. available bandwidth) and provided service levels and readapts services accordingly, keeping overhead low.

4.2.4 CloneCloud

CloneCloud is a system that automatically transforms mobile applications so that they can make use of cloud resources [3]. While mobile application are mainly designed as a monolithic process, where the whole execution is done on the mobile device, or as a traditional server-client model, where the server often does the most work, CloneCloud offers the possibility to individually offload application parts for execution on cloud resources. What makes CloneCloud different from other solutions is the fact that developers do not need to modify their application in order to make use of this. Instead, the system automatically transforms those applications and optimizes them for distributed execution in a cloud, based on the capabilities of the device and the cloud.

The system is primarily designed for applications that are executed in application-layer virtual machines (VMs), which according to [3] are widely used in mobile platforms. Examples for those VMs are the Java VM, Dalvik VM and Microsoft .NET.

![Figure 4.2 The VOLARE middleware modules, based on: [14]](image)

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:

1. The service request module
2. The context monitoring module
3. The adaption module
4. The QoS monitoring module
5. The service binding module

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.

![Figure 4.3 Partitioning analysis framework of CloneCloud, based on: [3]](image)

CloneCloud uses an off-line partitioning mechanism to divide parts of an application for local and remote execution and stores the corresponding results in a database. Neither the source code of the application nor special annotations are required for this mechanism to work. Figure 4.3 illustrates how the partitioning framework is composed: The application is analyzed by the static analyzer according to several migration constraints, resulting in a list of legal partitions of the application that can be migrated. Furthermore the dynamic profiler collects...
The optimization solver then uses both results to partition the application accordingly by selecting which parts of the application should be executed locally or remotely.

When a user runs the partitioned application, at some point in the execution process a migration point might be reached. The migration points are set upon launch according to the current context of the mobile device (e.g. availability of cloud resources and network link characteristics). If a migration point is reached, the corresponding thread gets suspended and the current state is captured in a package. The package then gets transferred to a synchronized clone in the cloud, where execution can be resumed. Upon reaching a re-integration point, a similar process is run, that captures and retransfers the corresponding thread back to the mobile device. Only one thread can be offloaded at a time, resulting in other threads, which want to communicate with the offloaded thread, to block until the offloaded thread is transferred back to the mobile device. Figure 4.4 illustrates how migration of a thread in CloneCloud takes place.

Documentation about the prototype implemented by the researchers gave no evidence about usage of any resource discovery or handover mechanisms. Neither any information about fallback mechanisms in the case of connectivity problems could be found. However, a saving of energy up to twenty times could be reached by the prototype in conjunction with offloading of a face detection application. Due to the fact that only the current state of a thread needs to be transferred between a mobile device and the cloud, the produced overhead due to data traffic as well as the number of requests might be considered low. However, due to the fact that both the mobile device and the clone need to be in a synchronized state (which probably also includes parts of the file system) an initial overhead is assumed, resulting in a lower rating, see Table 5.1.

4.2.5 Mobile Computation Outsourcing Framework

Chonglei Mei [11] proposes a mobile computation outsourcing framework for the Android platform, which is implemented in Java and according to [11] it can be easily deployed on any backend platform. In their prototype they used the Amazon EC2 cloud service to offload different applications from mobile devices to the cloud.

The framework consists of three main components, which are located at the cloud side, see Figure 4.5:

1. Proxy
2. Code repository
3. 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 [11], 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, [11] proposes to put 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 WiFi access points, which might offer the possibility to advertise cloud resources located nearby the mobile device. When the proxies are placed at the cloud side, [11] 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 proposed by Mei et al. uses DNS and especially mDNS to address cloud...
computing resources. Due to the large numbers of cloud computing resources, the current DNS structure can become overloaded. Therefore, this solution is rated as moderate, see Table 5.1. Discovery of resources is achieved by placing proxy servers close to network access points, which then advertise cloud services via mDNS. This is considered to be good as it offers a dynamic way to discover resources and also gives users the possibility to be redirected to resources near their current location. Mechanisms for handover or roaming support could not be identified. Energy usage is considered to be good, as the system monitors energy usage during local and remote execution and with the help of an outsourcing controller (which is currently in development) the measured values are used to decide whether application parts should be offloaded or not. The prototype revealed a decrease of energy consumption up to nine times. The produced overhead is considered moderate, as popular application components are stored at a code repository near the cloud.

5. DISCUSSION
The solutions presented in this paper focus on different aspects of cloud computing in connection with mobile usage. Table 5.1 gives an overview over the different solutions and their rating.

Table 5.1 Comparison of the different solutions.

<table>
<thead>
<tr>
<th>RNA</th>
<th>RDD</th>
<th>RDL</th>
<th>HSR</th>
<th>HLR</th>
<th>HDP</th>
<th>RS</th>
<th>EU</th>
<th>OM</th>
<th>DL</th>
</tr>
</thead>
<tbody>
<tr>
<td>CloneCloud</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>Cloud agency</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>VOLARE</td>
<td>n/a</td>
<td>+</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>Mobile Computation Offloading Framework</td>
<td>n/a</td>
<td>+</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
</tr>
</tbody>
</table>

The **CloneCloud** framework offers dynamic partitioning of unmodified mobile applications for partial remote execution in cloud services. It does this by analyzing applications for possible parts that can be offloaded to a remote resource and generates a cost model which is then used to determine which parts will be executed in the cloud.

Finally, 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.

Although there were several solutions that dealt with the discovery of resources, none of them proposed to make use of geographical localization services like GPS that most of today’s mobile devices have got built-in. However, by making use of such technologies it might be possible to better select cloud resources that are known to be nearby the current location of the mobile user. The user could then benefit from lower latencies and network providers could keep the produced data traffic low by providing cloud resources at geographically strategic positions.

Another interesting finding was the fact that none of the solutions considered roaming support for mobile devices in the way that connections between mobile devices and cloud resources are identified as being topology-agnostic. One of the problems with roaming of mobile users is the fact that IP addresses of mobile devices can change while roaming from one location to another, possibly resulting in a reset of currently on-going sessions. By using an identification mechanism that does not simply rely on the IP address of a device but rather on a unique identifier that maybe linked to the current IP address of a device, this issue might possibly be solved.

6. CONCLUSION AND FUTURE WORK
In section 2 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 smartphones 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 3 an outline of past, current and possible future communication network architectures was presented to answer the second research questions (RQ2), which covered different communication network architectures that can be used in order to support cloud computing services on smart phones and other resource-starved devices. With the fast development of fixed broadband internet connections in the past decades, their wireless complements were extended continuously in order to measure up, resulting in current available technologies that offer mobile data rates up to several hundred megabits per second. UMTS, Wireless-LAN and low-latency technologies like LTE present an interesting possibility to offer cloud computing services to mobile users.

In section 4, 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 third research question (RQ3). 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 (RQ4).
Each of the presented solutions offers an approach for one of the open issues that are connected 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 generate a new solution that covers most of the issues currently experienced. Such a solution might have the chance 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.

7. REFERENCES