Future Generation Computer Systems 29 (2013) 84–106

Contents lists available at SciVerse ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Mobile cloud computing: A survey

Niroshinie Fernando *, Seng W. Loke *, Wenny Rahayu

Department of Computer Science and Computer Engineering, La Trobe University, Australia

ARTICLE INFO

Article history:
Received 31 August 2011
Received in revised form
22 May 2012
Accepted 30 May 2012
Available online 6 June 2012

Keywords:
Mobile cloud computing
Task offload
Pervasive networks

ABSTRACT

Despite increasing usage of mobile computing, exploiting its full potential is difficult due to its inherent problems such as resource scarcity, frequent disconnections, and mobility. Mobile cloud computing can address these problems by executing mobile applications on resource providers external to the mobile device. In this paper, we provide an extensive survey of mobile cloud computing research, while highlighting the specific concerns in mobile cloud computing. We present a taxonomy based on the key issues in this area, and discuss the different approaches taken to tackle these issues. We conclude the paper with a critical analysis of challenges that have not yet been fully met, and highlight directions for future work.

1. Introduction

The increasing usage of mobile computing is evident by the study by Juniper Research, which states that the consumer and enterprise market for cloud-based mobile applications is expected to rise to $9.5 billion by 2014 [1]. In recent years, applications targeted at mobile devices have started becoming abundant with applications in various categories such as entertainment, health, games, business, social networking, travel and news. The popularity of these are evident by browsing through mobile app download centers such as Apple's iTunes or Nokia's Ovi suite. The reason for this is that mobile computing is able to provide a tool to the user when and where it is needed irrespective of user movement, hence supporting location independence. Indeed, 'mobility' is one of the characteristics of a pervasive computing environment where the user is able to continue his/her work seamlessly regardless of his/her movement.

However, with mobility comes its inherent problems such as resource scarceness, finite energy and low connectivity as outlined by Satyanarayanan in [2]. These pose the problem of executing many useful programs that could aid the user and create a pervasive environment. According to Tim O'Reilly 'the future belongs to services that respond in real time to information provided either by their users or by nonhuman sensors' [3]. Real time applications are just one type of mobile applications that demand high levels of responsiveness, that in turn, demand intensive computing resources. Some mobile applications, such as location based social networking, process and make use of the phone's various sensor data. However, extensive use of sensors, such as obtaining a GPS reading, is expensive in terms of energy and this limits the mobile phone in providing the user a better service through its embedded sensors. Furthermore, consider applications that require extensive processing – image processing for video games, speech synthesis, natural language processing, augmented reality, wearable computing—all these demand high computational capacities thus restricting the developers in implementing applications for mobile phones. Considering the trends in mobile phone architecture and battery, it is unlikely that these problems will be solved in the future. This is, in fact, not merely a temporary technological deficiency but intrinsic to mobility [4], and a barrier that needs to be overcome in order to realize the full potential of mobile computing.

In recent years, this problem has been addressed by researchers though cloud computing. Cloud computing can be defined as the aggregation of computing as a utility and software as a service [5] where the applications are delivered as services over the Internet and the hardware and systems software in data centers provide those services [6]. Also called 'on demand computing', 'utility computing' or 'pay as you go computing', the concept behind cloud computing is to offload computation to remote resource providers. The key strengths of cloud computing can be described in terms of the services offered by cloud service providers: software as a service (SaaS), platform as a service (PaaS), and infrastructure as a service (IaaS) [7]. Extensive surveys on cloud computing such as [6,5,8,7,9,10] can be found in the literature, and here we focus on the potential of, and the challenges faced by mobile cloud computing.

The concept of offloading data and computation in cloud computing, is used to address the inherent problems in mobile

* Corresponding authors.
E-mail addresses: niro_ucsc@yahoo.com, tnfernando@students.latrobe.edu.au (N. Fernando), s.loke@latrobe.edu.au (S.W. Loke), w.rahayu@latrobe.edu.au (W. Rahayu).

0167-739X/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2012.05.023

computing by using resource providers other than the mobile device itself to host the execution of mobile applications. Such an infrastructure where data storage and processing could happen outside the mobile device could be termed a 'mobile cloud'. By exploiting the computing and storage capabilities of the mobile cloud, computer intensive applications can be executed on low resource mobile devices.

Some of the key questions needing to be answered are: How does mobile cloud computing differ from cloud computing? What approaches have been made towards mobile cloud computing and how do they differ from each other? How can computation be offloaded and distributed to the cloud efficiently and in which ways does this differ from traditional distributed computing? What incentives can be used to persuade surrounding surrogate devices to participate in sharing the workload? How can context information be used in a beneficial way? How does mobility affect the performance of a mobile cloud?

The goal of this paper is to discuss in detail the current research that addresses these issues. We review the proposed solutions, and explore the upcoming research challenges in mobile cloud computing.

Organization of paper

The remainder of this paper is organized as follows: In Section 2, we present the motivation for mobile cloud computing, and discuss potential applications and scenarios. In Section 3, we briefly introduce cloud computing and mobile cloud computing, and identify three definitions of mobile cloud computing. Next, in Section 4 we propose a taxonomy of mobile cloud computing based on the key issues, and review how each issue has been tackled in related research. We provide a discussion on the current challenges in Section 5. Finally, conclusions and directions for future research are identified in Section 6.

2. Motivation: the need for a mobile cloud

The case for mobile cloud computing can be argued by considering the unique advantages of empowered mobile computing, and a wide range of potential mobile cloud applications have been recognized in the literature. These applications fall into different areas such as image processing, natural language processing, sharing GPS, sharing Internet access, sensor data applications, querying, crowd computing and multimedia search. However, as explained in [11], applications that involve distributed computation do have certain common characteristics, such as having data with easily detectable segment boundaries, and the time to recombine partial results into a complete result must also be small. An example is string matching/manipulation such as grep and word frequency counters. The different applications and scenarios presented in recent literature are described in detail below:

1. Image processing: In [11], the authors have experimented with running GOCR,[1] an optical character recognition (OCR) program on a collection of mobile devices. In a real life scenario, this would be useful in a case of a foreign traveler who takes an image of a street sign, performs OCR to extract the words, and translates words into a known language. A similar scenario is given in [12] where a foreign tourist Peter is visiting a museum in South Korea. He sees an interesting exhibit, but cannot understand the description since it is in Korean. He takes a picture of the text, and starts an OCR app on his phone. Unfortunately his phone lacks the resources to process the whole text. Although he could connect to a remote server via the Internet, that would mean he use roaming data which is too expensive. Instead, his device scans for nearby users/devices who are also interested in reading the description, and requests sharing their mobile resources for the task collaboratively. Those who are interested in this common processing task create an ad hoc network with Peter and together, their mobile cloud is able to extract the text, and then translate it to English. This can be applied to many situations in which a group is involved in an activity together. Another example is a group performing archaeological expeditions in a desert.
2. Natural language processing: As mentioned above, language translation is one possible application, and this is mentioned in [11] as a useful tool for foreign travelers to communicate with locals. Translation is a viable candidate since different sentences and paragraphs can be translated independently, and this is experimentally explored in [11] using Pangloss-Lite [13]. Text-to-speech is also mentioned in [11], where a mobile user may prefer having a file read to them, especially in case of the visually impaired.
3. Crowd computing: Video recordings from multiple mobile devices can be spliced to construct a single video that covers the entire event from different angles, and perspectives [14]. In [15], two scenarios of this nature are described in detail: 'Lost child' and 'Disaster relief'.
   The 'Lost child' scenario takes place at a parade in Manhattan. John, a five year old child who is attending the parade with his parents goes missing among all the people, and his parents only notice he is missing after some time. Fortunately, a police officer sends out an alert via text message to all mobile phones within a two mile radius, requesting them to upload all photographs they have taken in the parade during the past hour, to a server that only the police has access to. With John's parents, the police officer searches through these photographs via an app on his phone. After looking through some pictures, they are able to spot John in one of the images, which they identify to be taken at a nearby location. Soon, the relieved parents are reunited with their child.
   In the 'Disaster relief' scenario, a massive earthquake measuring 9.1 on the Richter scale has occurred in Northern California, resulting in much human loss, and infrastructure and property destruction. Disaster relief teams are facing an uphill task because of limited manpower, lack of transportation, and poor communication. Internet infrastructure has been destroyed. Previous maps on terrain and buildings are suddenly rendered obsolete, contributing to slow disaster relief. Data on Google Earth and Google Maps on this area is now useless since highways, bridges, landmarks and buildings have now all collapsed. To conduct efficient search and rescue operations, new data must be gained and a clear picture of the terrain and buildings state must be constructed. To do this, the relief teams use camera based GigaPan sensing.[2] Local citizens are asked to use their mobile phones to photograph disaster sites, and these are collected at a central server. The collected images are then sewn together to create a whole, panoramic image. The new face of the area emerges, and relief teams can now conduct their work with accurate maps and information on inaccessible areas.
4. Sharing GPS/Internet data: It is more efficient to share data among a group of mobile devices that are near each other, through local-area or peer-to-peer networks. It is not only cheaper, but also faster [14]. Rodriguez et al. [16] present a case study of a hiking party at Padjelanta National Park, which is a deserted land in the Arctic circle lacking power access

[1] http://jocr.sourceforge.net/.

[2] http://www.gigapan.com/.

points and network coverage. A data set contains Bluetooth scans for discovering devices and GPS reads of 17 persons. The paper reports up to 11% energy savings by sharing GPS readings. However colocation of most participants was low, so this energy savings should be much higher in a conventional hiking party, or other social situations such as pubs, restaurants, and stadiums where energy saving could be up to 40%. A similar scenario is a mobile device requesting to access a p2p file which is downloaded or is currently being downloaded by another mobile in the vicinity [12].

5. Sensor data applications: Since most mobile phones are equipped with sensors today, readings from sensors such as GPS, accelerometer, light sensor, microphone, thermometer, clock, and compass can be timestamped and linked with other phone readings. Queries can then be executed on such data to gather valuable information. Such queries could be "What is the average temperature of nodes within a mile of my location?" or "what is the distribution of velocities of all nodes within half a mile of the next highway on my current route?" Sample applications for this are traffic reporting, sensor maps, and network availability monitoring [14].
6. Multimedia search: Mobile devices store many types of multimedia content such as videos, photos, and music. For example, Shazam is a music identification service for mobile phones, that searches for similar songs in a central database. In the context of the mobile cloud, the searching could be executed on the contents of nearby phones [14].
7. Social networking: Since sharing user content is a popular way we interact with friends on social networks such as Facebook, integrating a mobile cloud into social networking infrastructure could open up automatic sharing and p2p multimedia access, and this will also reduce the need to back up and serve all of this data on huge servers [14].

2.1. Example scenario: using mobile cloud with distributed computation, and collective sensing

Now let us consider the following detailed scenario: In the aftermath of a natural disaster such as the Indian Ocean tsunami in 2004, the immediate provisioning of emergency services becomes of great importance. Among these services, searching for missing persons is one of the most critical yet excruciating tasks. In this kind of chaotic situation, infrastructure is destroyed, limiting access to computers and data, making such a search even more difficult. Often, missing person reports are filed, but the persons in question may be injured with no means of communication, or even deceased. One way of dealing with this is to photograph every person found, gather all images to a central location, and perform search and match operations with images of missing persons. However, this approach is not very realistic considering the limited human and machine resources in such a situation. Several questions exist in this scenario:

1. How and who would capture the images necessary?
2. How would the captured images be collected?
3. How would the collected images be processed?

The first question is easily answered. Anyone with a camera phone of decent quality could contribute to this. However, the second and third questions—data collection and processing, are more tricky. Acquired data could be uploaded to a remote server, but as is often the case in such disaster sites, connectivity would be a problem. Also this method could take a while, especially if a centralized server node is not already set up. Images could be processed locally, but mobile devices are typically not equipped with enough resources to carry out such operations (individually).

Let us now consider the possibility of employing a local mobile cloud for the aforementioned scenario. In this case, photographs taken by various individuals would constitute the data against which the missing persons will be matched. Relief workers and communities working together at the disaster site could collaboratively 'lend' their mobile devices' storage and processing resources to a 'local mobile cloud', that could effectively carry out the image processing needed to identify the missing persons.

A key challenge here is the fact that the number of, and the type of available resources cannot be known or predicted beforehand. How then, can the work be efficiently distributed and load balanced? Furthermore, in such situations it most likely that devices will encounter other unknown nodes, rather than familiar devices. Therefore, it is important that the mobile cloud be able to give a performance gain even without prior information.

The aforementioned scenario is only one example demonstrating the need for a mobile cloud computing framework. In wearable computing, two major challenges are to reduce the bulkiness of systems for every day use and not having enough battery power [17]. This could be solved by offloading/sharing the computational jobs to the local 'mobile cloud', while sensors and peripherals facilitate the pervasive experience for the user. In the area of augmented reality, it has been suggested that using cloud resources [18] can solve similar problems. In biomedical engineering, wearable medical devices forming Body Area Networks (BANs) can enable real time collection and analysis of patients' medical data [19].

2.2. Remote proxy versus local resources

Today we do have mobile applications connected to the cloud, such as Apple iCloud,[3] Google's Gmail for Mobile,[4] and Google Goggles.[5] Using mobile devices for disaster situations has also been explored in work such as [20–22]. However, current mobile cloud applications, or apps connect to a remote server where the brunt of the computations are performed. The mobile devices exist purely as thin clients that connect to a remote proxy providing complex services. Although these apps are becoming popular, they can perform well only under high speed connectivity. However, it is not practical to assume speedy connections, affordable data access fees and good response times in most places of the world. Except city areas, this holds true even in most developed countries. In contrast, short range communication consumes less energy, and this is a key factor since mobile devices usually operate on a limited energy source. Also, connecting to local resources would be cheaper and promise faster connectivity and better availability. As explained in [23] by Satyanarayanan, compared to WiFi LAN bandwidths of 400 Mbps, mobile wireless Internet operates at a bandwidth of 2 Mbps. Depending on user interaction, latency could vary significantly. For example, it is 80 ms versus 16 ms for a 4 MB image, and this would greatly hinder the execution and usefulness of the app, as well as the user experience. Satyanarayanan [23] predicts that, considering the current trajectory of Internet evolution, although bandwidth is likely to improve, latency is not.

Therefore, considering the data access fees, issues with latency and bandwidth [24], and also the high demands of energy when using 3G connectivity, the local cloud would be a better alternative to the remote cloud [23]. Furthermore, using the local mobile resources is an efficient way of making use of available

[3] http://www.apple.com/icloud/.
[4] http://www.google.com/mobile/mail/.
[5] http://www.google.com/mobile/goggles/#text.

computation power, that would otherwise be idling [25]. Since typically mobile devices are equipped with sensing capabilities, a cloud made up of mobile devices will be able to provide the users with context and location aware services as well, leading to a more personalized experience.

By combining the local cloud with other mobile devices as opposed to local servers, we are able to support mobility without needing additional infrastructure. Considering the trends for smart phones, which shows that they are getting more powerful each year, a local mobile cloud will be able to provide sufficient resources for intensive mobile apps. It is feasible to envision the future mobile clouds as hybrids, where the users themselves would act as cloud resources, but with the ability to connect to remote servers in cases of good connectivity and other conditions such as access fees, available battery, and response time. This would require the mobile cloud architecture to be proactive, self-adaptive and equipped with cost–benefit analysis capabilities.

To summarize, the reasons for sharing/offloading work from a mobile device would be: limited computational capability, limited battery power, limited connectivity, opportunity to gather more sensing data (such as encountering other mobile devices with different sensing abilities), access to different content/data sets, and to make use of idling processing power.

The advantages of sharing work with local nearby resources versus a remote proxy would be due to: limited connectivity to remote servers (such as in remote areas, and dead zones), limited battery power inhibiting long range communications, data access fees, and high availability of local resources.

However, concerns about privacy and security are a major issue when sharing work. Would users be comfortable sharing their resources with unfamiliar people? Would mobile clouds consisting of 'known groups' such as co-workers, friends and family be more feasible? What incentives can be provided to entice people to share their resources, and what security and privacy measures can be taken to ensure safety? Even from a trusted user, his/her mobile device may be unfamiliar. Furthermore, mobile environments are typically dynamic and unpredictable. In such cases, how can a mobile cloud function opportunistically to ensure maximum gain? These are valid challenges that concern the future of mobile cloud computing, and we shall discuss these in detail in later sections.

3. Cloud computing vs. mobile cloud computing

3.1. Cloud computing

"Cloud computing refers to both the applications delivered as services over the Internet and the hardware and systems software in the datacenters that provide those services" [6].

A cluster of computer hardware and software that offer the services to the general public (probably for a price) makes up a 'public cloud'. Computing is therefore offered as a utility much like electricity, water, gas etc. where you only pay per use. For example, Amazon's Elastic cloud, Microsoft's Azure platform, Google's App Engine and Salesforce are some public clouds that are available today. However, cloud computing does not include 'private clouds' which refer to data centers internal to an organization. Therefore, cloud computing can be defined as the aggregation of computing as a utility and software as a service.

Virtualization of resources is a key requirement for a cloud provider—for it is needed by statistical multiplexing that is required for scalability of the cloud, and also to create the illusion of infinite resources to the cloud user. Ambrust et al. [5] holds the view that "different utility computing offerings will be distinguished based on the level of abstraction presented to the programmer and the level of management of the resources". To take an example from the existing cloud providers, an instance of Amazon's EC2 is very much like a physical machine and gives the cloud user almost full control of the software stack with a thin API. This gives the user a lot of flexibility in coding; however it also means that Amazon has little automatic scalability and failover features. In contrast, Google's App Engine enforces an API on the user but offers impressive automatic scalability and failover options. Microsoft's Azure platform is something in between the aforementioned providers by giving the user some choice in the language and offers somewhat automatic scaling and failover functions. Each of the aforementioned providers has different options for virtualizing computation, storage and communication.

3.2. Mobile cloud computing

There are several existing definitions of mobile cloud computing, and different research alludes to different concepts of the 'mobile cloud':

1. Commonly, the term mobile cloud computing means to run an application such as Google's Gmail for Mobile[6] on a remote resource rich server (in this case, Google servers) as displayed in Fig. 1,[7] while the mobile device acts like a thin client connecting over to the remote server through 3G. Some other examples of this type are Facebook's location aware services, Twitter for mobile, mobile weather widgets etc.
2. Another approach is to consider other mobile devices themselves too as resource providers of the cloud making up a mobile peer-to-peer network as in [14]. Thus, the collective resources of the various mobile devices in the local vicinity, and other stationary devices too if available, will be utilized as shown in Fig. 2. This approach supports user mobility, and recognizes the potential of mobile clouds to do collective sensing as well. Peer-to-peer systems such as SATIN [26] for mobile self-organizing exist, but these are based on component model systems representing systems made up of interoperable local components rather than offloading jobs to local mobile resources. This paper focuses primarily on this latter type of work.
3. The cloudlet concept proposed by Satyanarayanan [23] is another approach to mobile cloud computing. Fig. 3 illustrates this approach where the mobile device offloads its workload to a local 'cloudlet' comprised of several multi-core computers with connectivity to the remote cloud servers. PlugComputers[8] can be considered good candidates for cloudlet servers because of their form factor, diversity and low power consumption. They have the same general architecture as a normal computer, but are less powerful, smaller, and less expensive, making them ideal for role small scale servers installed in the public infrastructure. These cloudlets would be situated in common areas such as coffee shops so that mobile devices can connect and function as a thin client to the cloudlet as opposed to a remote cloud server which would present latency and bandwidth issues.

Mobile cloud computing would also be based under the basic cloud computing concepts. As discussed by Mei et al. in [10] there are certain requirements that need to be met in a cloud such as adaptability, scalability, availability and self-awareness. These are also valid requirements for mobile cloud computing. For example, a mobile computing cloud also needs to be aware of its availability and quality of service and enable diverse mobile computing entities to dynamically plug themselves in, depending

[6] http://www.google.com/mobile/mail/.

[7] Google translate image from http://www.ausbt.com.au/iphone-app-review-google-translate.

[8] http://www.plugcomputer.org/.

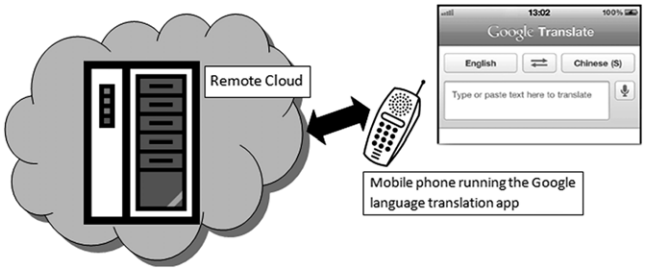

Fig. 1. A remote cloud server catering to mobile devices though the internet.

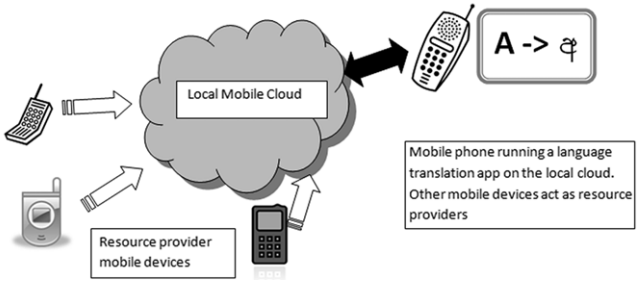

Fig. 2. A virtual resource cloud made up of mobile devices in the vicinity.

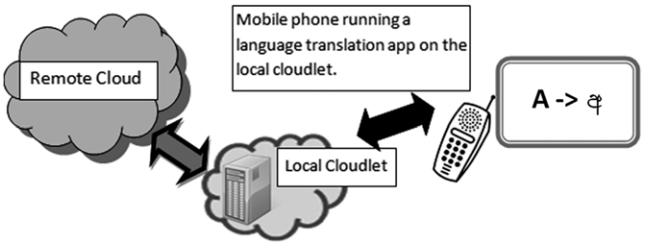

Fig. 3. A cloudlet enabling mobile devices to bypass latency and bandwidth issues while benefitting from its resources.

on the requirements and workload. And in order for mobile users to efficiently take advantage of the cloud, a suitable method of self-assuming one's own quality is needed—since the internal status and the external environment is subject to change. However, in addition to the similar requirements, a mobile cloud needs to consider other aspects such as mobility, low connectivity and finite source of power as well.

4. A taxonomy of mobile cloud computing

We present a taxonomy of current approaches in mobile cloud computing research based on issues related to Operational, End user and Service levels, and also in areas of Security, Context awareness and Data management as illustrated by Fig. 4. Our criteria for defining the taxonomy is based on the key issues in mobile cloud computing, and how they have been tackled in academia. We focus on:

- Operational level issues
- End user level issues
- Service and application level issues
- Privacy, security and trust
- Context-awareness
- Data management

as the main areas.

These issues at the top tier of the taxonomy are applicable to many areas, and not just mobile cloud computing. We believe these similarities would help give a comparison on how mobile cloud computing relates to other fields. Moreover, we expand each issue to highlight the unique set of challenges in mobile cloud computing, and how they have been tackled in existing work.

4.1. Operational issues

Operational issues refer to underlying technological matters such as the method of offloading computations, cost–benefit models that aid in taking the decision to offload or not, how the mobility of devices is managed/supported, and connection protocols used.

4.1.1. Method of offloading

The main operation in any mobile cloud would be the offloading of jobs that take place from the resource constrained mobile

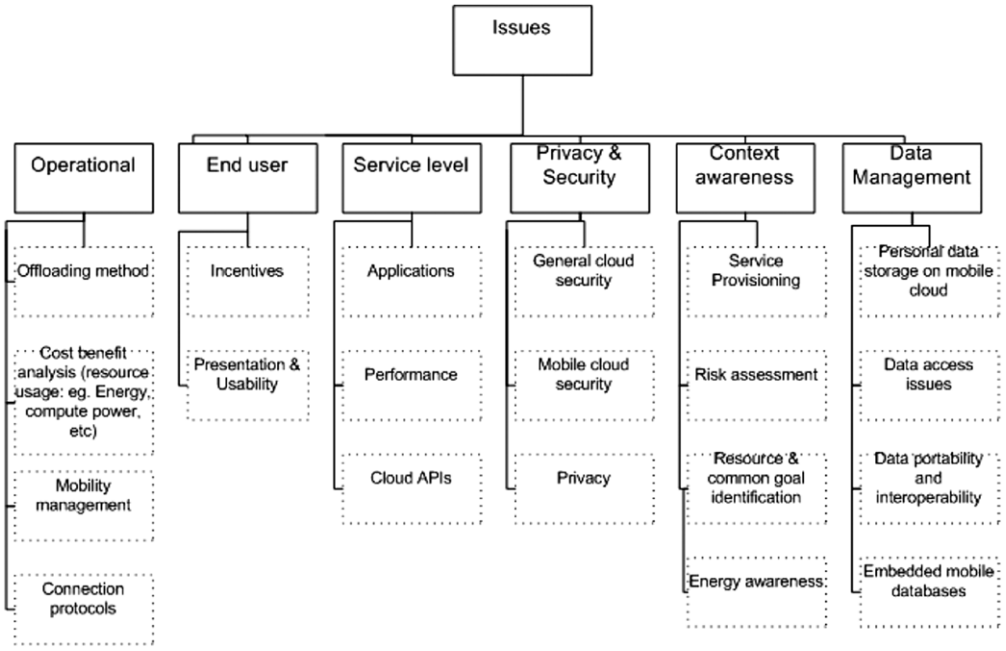

Fig. 4. A taxonomy of issues in mobile cloud computing.

device to the cloud. Because of issues such as the physical distance separating the mobile device and the cloud and the heterogeneity of the underlying systems, different research has tackled this in a variety of ways. Current research discusses offloading methods in three main directions; Client–Server Communication methods, Virtualization, and Mobile agents.

Client–Server Communication. In the Client–Server Communication process communication is done across the mobile device(offloader) and surrogate device via protocols such as Remote Procedure Calls (RPC), Remote Method Invocation (RMI) and Sockets. Both RPC and RMI have well supported APIs and are considered stable by developers. However, offloading through these two methods mean that services need to have been pre-installed in the participating devices. This is a disadvantage when considering the ad hoc and mobile nature of a mobile cloud and restricts the mobility of users if in the vicinity of devices that do not support the needed services.

Spectra [27] and Chroma [28] are two examples of systems that use pre-installed services reachable via RPC to offload computation. Applications use RPC to invoke functionality in remote and local Spectra servers. When the device needs to offload an application, the Spectra client consults a database that stores information about Spectra servers such as their current availability, CPU load etc. These servers are pre-installed with application code acting as services. Developers need to manually partition the applications by specifying which methods might be candidates for offloading. Spectra decides at runtime depending on the resource pool, which of those aforementioned methods, if any will be offloaded and to which surrogate.

Marinelli [14] has presented 'Hyrax' for Android smartphone applications which are distributed both in terms of data and computation based on Hadoop[9] ported to the Android platform. Hyrax explores the possibility of using a cluster of mobile phones as resource providers and shows the feasibility of such a mobile cloud. As a sample application, they present 'HyraxTube'; which is a simple distributed mobile multimedia search and sharing program. The objective of HyraxTube is to allow users to search through multimedia files in terms of time, quality, and location. Apache Hadoop is an open source implementation of MapReduce [29] and provides a virtualized interface to a cluster of computers scaled randomly. In Hyrax, a central server with access to each mobile device coordinates data and jobs, and the phones communicate with each other on an isolated 802.11g network. Just like in a normal Hadoop implementation, Hyrax also has a NameNode and a JobTracker instance running on a central server with access to each of the client mobile devices. The central server does not do any of the processing, and is responsible for coordinating data and jobs. Each mobile phone runs instances of the DataNode and the TaskTracker in separate Android services. Additionally, each phone runs threads that stores the phones' multimedia data on the Hadoop Distributed File System (HDFS) and threads that record sensor data. The TaskTrackers and DataNodes use a periodic heartbeat call through RPC to JobTracker and NameNode, and vice versa on heartbeat response. The heartbeat is sent from TaskTrackers to show the JobTracker that they are 'alive' and JobTracker can assign jobs in the response of this heartbeat.

Another framework based on Hadoop is presented by Huerta-Canepa and Lee [12], for a virtual mobile cloud focusing on common goals in which mobile device are considered as resource providers. They argue that people's location plays a major role in their activities; hence, collocation leads to common activities, especially place bound tasks such as visiting a museum, and performing archaeological expeditions. The system's Offloading manager module organizes sending and receiving jobs to and from other devices and creating virtual machines on surrogates. On a surrogate device, the tasks are executed on a virtual machine acting as a protected space thus ensuring the security of device data. The implementation was tested using a Korean OCR application. Their results do not show a speedup, however, though they suggest an energy saving, since the processing time is actually less than when executed on a single mobile device.

The 'Cuckoo' framework [30] presents a system to offload mobile device applications onto a cloud using a Java stub/proxy model. Cuckoo can be offloaded onto any resource that runs the Java Virtual machine, be it a commercial cloud such as Amazon EC2[10] or a private mini cloud comprised of laptops and local clusters. However, other mobile phones have not been mentioned as potential resource providers. Implemented for Android, Cuckoo's offloading objectives are to enhance performance and reduce battery usage. The Ibis High Performance Programming System [31] is used as the basis for Cuckoo's communication component. To use Cuckoo, the applications need to be re-written such that the application supports remote execution as well as local execution. For this purpose, a programming model, functioning as an interface of the system, is made available to application developers. The programming model uses the Android's existing 'activity/service' model that separates the services (compute intensive methods that are candidates for offloading) and activities (interactive methods of the application). A proxy object is created at the activity which is linked to the actual implementation. Here, in addition to the normal local implementation, Cuckoo generates code for the same implementation for the remote service that may or may not be identical to the local one, since the remote version could be run on a multi-core computer for instance, and take full advantage of parallelism. If remote resources are not available (network connectivity is not available) then the application can run on local resources (the phone) entirely. They have re-implemented two existing applications 'eyeDentify' and 'PhotoShoot' in Cuckoo to demonstrate the effectiveness of the framework and report that they gained a speedup of a factor of 60 and reduced the battery consumption by a factor of 40 in 'eyeDentify' by offloading. Although they mention that they have gained considerable speedups for 'PhotoShoot' as well, exact figures are not mentioned. However, they do not provide a method to decide whether to offload or not. This has been included as a future research step, but for now, the framework always offloads by default if the phone can connect to a cloud of resources.

The Mobile Message Passing Interface (MMPI) framework [32] is a mobile version of the standard MPI over Bluetooth where mobile devices function as fellow resource providers. Instead of the typical star network structure of normal piconets, MMPI employs a fully interconnected mesh structure so that each node can communicate with the other. Tasks such as device discovery, and connections are handled by the libraries provided in the framework, eliminating the need for writing any Bluetooth specific code explicitly. The framework is implemented in Java and the third party library BlueCove [33] is used to handle Bluetooth operations. The master mobile device passes job parameters to the slave devices, which they then proceed to execute. As a sample application, they have tried with fractal generation over different devices (phones, laptops, PDAs) on cross-platforms and reported the time cost. However, they do not give the cost in terms of using just one machine (in the conventional way) so it is not clear what the speedup was. The setting up of MMPI system is in three steps: Device discovery, Service discovery, and Network formation. Their

[9] http://hadoop.apache.org.

[10] http://aws.amazon.com/ec2/.

tests show that service discovery time increases as more devices are discovered. Network formation takes the smallest time with 136 ms for a two node network and 2.3 s for a four node network.

In [34], Deboosere et al. propose a grid model, where a mobile device connects to a server as a thin client over a classic thin client protocol such as VNC (Virtual Network Computing) or a streaming protocol. In this system, user input is sent via the wireless network to the server, and after processing the input, the server sends back the appropriate graphical output, which is then displayed on the mobile device. In particular, this research focuses on server selection algorithms that are needed when the mobile device's location changes. To minimize the delay from server, and provide a quick response time, the application may need to be migrated to a nearby server, which may affect the performance. The authors aim to provide effective algorithms to minimize this degradation in performance, while supporting user mobility as well.

Virtual machine (VM) migration. VM migration refers to transferring the memory image of a VM from a source server to the destination server without stopping its execution [35]. In such a live migration, the memory pages of the VM are pre-copied without interrupting the OS or any of its applications, thereby providing an illusion of seamless migration. This method ensures that no code changes are needed when programs are offloaded, and provides a relatively secure execution, since the VM boundary insulates the surrogate device. However, VM migration is somewhat time consuming and the workload could prove to be heavy for mobile devices.

Rather than connecting to a distant cloud, Satyanarayan et al. [23] suggests 'cloudlets' as a solution. A cloudlet is similar to a small data center that is situated on designated areas/places and is connected to a larger cloud server via the Internet. They state that ''internally, a cloudlet resembles a cluster of multi-core computers, with gigabit internal connectivity and a high-bandwidth wireless LAN''.

Thus, the mobile devices are in physical proximity to the resource rich cloudlet and it can function as a thin client while all the resource intensive computation happens inside the cloudlet. The mobile device would be connected to the cloudlet by a low-latency one hop high bandwidth wireless connection, thereby guaranteeing real time interactive response. If the user moves away from the cloudlet, the mobile device could fall back to a degraded service mode that connects to a distant cloud server—or at worst case—even operate offline. Cloudlets would be decentralized, widely dispersed and self-managing requiring little power, internet connectivity and control for setup. They could be owned by a particular local business (as opposed to cloud ownership by companies such as Amazon, Google etc.), such as a coffee shop or a dentist's office. Also, a cloudlet would only contain cached data that is available elsewhere, so that the loss of a cloudlet would not be disastrous. Instead of tightly restricting the software that can be run on the cloudlet, the research suggests having minimal restrictions on the software and simplifying management. Their solution is to use ''transient customization of cloudlet infrastructure using hardware VM technology''. The transient nature means that pre-use customizations and post-use cleanups would restore the cloudlet infrastructure to its original software state after each use. The VM would encapsulate and separate the guest software from the cloudlet's host software. A VM based approach is more stable than other alternatives such as process migration and would also be more flexible than language-based virtualization. Two approaches are considered to transfer the VM state from the mobile device to the cloudlet: VM migration, and dynamic VM synthesis. Of those two, the authors propose to use dynamic VM synthesis because its performance depends solely on local resources and WAN failure wouldn't affect synthesis (in which case base VMs could be transferred to the cloudlet via physical media).

MAUI [24] uses a combination of VM migration and code partitioning. Their objective is to save energy. Applications are offloaded from phones to surrounding infrastructure—i.e. local and remote servers. Implemented in.NET, MAUI's partitioning is done at runtime and is very dynamic. It can use either 3G or WiFi for connectivity. Developers annotate which methods can be offloaded and at the time of execution, if there is a remote server available, MAUI decides whether or not to offload these methods.

CloneCloud [36] also uses VM migration to offload part of their application workload to a resourceful server through either 3G or WiFi. Because they use device clones, the mobile applications are unmodified and there is no need of even annotating methods such as done in MAUI [24]. They have a 'cost model' that analyses the cost involved in migration and execution on the cloud and compares the cost against a monolithic execution. CloneCloud was tested using Android phones with the clones executing on a Dell desktop running Ubuntu. For testing purposes, they considered three applications; a virus scanner, image search, and a privacy-preserving targeted advertising and report speedups up to 21.2× with WiFi giving better performance over 3G. However, they assume that the clone VM environment is by default a trusted one, and mention establishing trust as future work.

MobiCloud [37] discusses using cloud computing technology for MANETs (mobile ad hoc networks) in a secure way. Traditional MANETs can be transformed into a service oriented architecture by MobiCloud. Each mobile node is considered as a 'Service Node' that can be used as a service provider or a service broker depending on its computation and communication capabilities and available resources. Every service node is incorporated on to the cloud as a virtualized component and is mirrored in the cloud. These Extended Semi-Shadow Images (ESSIs) are not exactly the same as virtual images since an ESSI could be an exact clone, a partial clone, or merely an image that has extended functions of the physical device. A virtualized MANET routing and communication layer is established by these ESSIs to assist the physical mobile nodes that they represent. The key focus of MobiCloud is to provide a security service architecture and they present 'Virtual Trusted and Provisioning Domain' (VTaPD), which is a service to handle information flows in various security domains, using programmable routing [38].

Mobile code. Scavenger [39] is another framework that employs cyber-foraging using WiFi for connectivity, and uses a mobile code approach to partition and distribute jobs. It also introduces a scheduler for cost assessment. Its method of cost assessment is based on the speed of the surrogate server and it uses a benchmarking method to do this. Using its framework, it is possible for a mobile device to offload to one or more surrogates and its tests show that running the application on multiple surrogates in parallel is more efficient in terms of performance. However, it does not discuss fault tolerance mechanisms and since its method is strictly about offloading on surrogates and not sharing, it is not really dynamic. Also its surrogates are all desktops and it is unclear if Scavenger is too heavy to run on mobile phones.

Discussion. With the exception of Hyrax [14], Virtual cloud [12] and Cuckoo [30], the most recent works have used either VM migration or Mobile code to offload tasks. Even the aforementioned projects, are based on much older frameworks; Hadoop [40] and Ibis [31], designed for distributed and grid programming. Therefore it is safe to say that the trends in this particular area favor VM migration and Mobile code over the conventional Client–Server Communication systems. The advantages of these two approaches over Client–Server Communication methods such as RPC can be given as a reason for this. Although Client–Server Communication methods have well supported APIs and are robust, they require the applications to be pre-installed. Also, disconnected operations are not supported in this method. Considering the ad-hoc nature

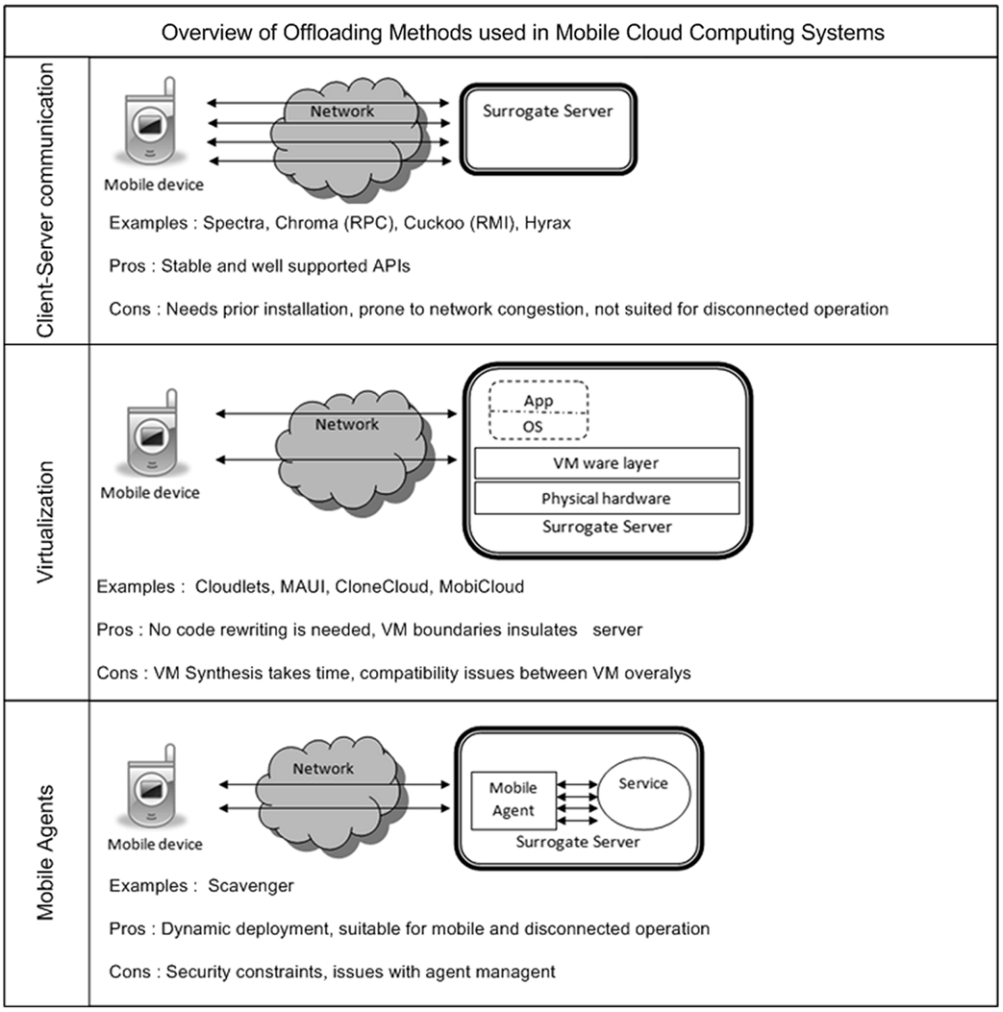

Fig. 5. An overview of the major approaches in offloading methods for mobile cloud computing, and their advantages and disadvantages.

of mobile systems, this is a disadvantage. Furthermore, the continuous on-going interaction and communication between the client and server may lead to network congestion.

VM migration is used by a majority of frameworks, including Cloudlets [23], MAUI [24], CloneCloud [36], and MobiCloud [37]. Virtualization greatly reduces the burden on the programmer, since very little or no rewriting of applications is required. However, full virtualization with automatic partitioning is unlikely to produce the same fine grained optimizations as that of hand coded applications, although rewriting each and every application for code offload is also not practical. MAUI [24] actually does not rely on pure VM migration as done in CloneCloud [36] and Cloudlets [23], but uses a combination of VM migration and programmatic partitioning. However, in cases where the mobile device user is within range of a surrogate device for a few minutes, using VM migration may prove to be too heavyweight, as is pointed out in [39] which uses mobile agents in light of its suitability in a dynamic mobile environment.

Fig. 5 provides a general overview of these points discussed.

Although the evaluation results have been given in some projects, comparing them against each other is difficult since the performance and energy savings depend on the application as well. In fact even when using the same framework, performance varies for different applications. The input size and connection protocol (whether 3G or WiFi) also plays a key role. For example, MAUI [24] reports maximum energy savings of 90%, 45%, and 27% for three applications (face recognition, chess, video game) and maximum performance speedups of roughly 9.5, 1.5, and 2.5 for the same respectively. CloneCloud [36] also reports test results on three applications; virus scanning, image search, behavior profiling. They show maximum speedups of 14.05, 21.2, and 12.43 respectively for the aforementioned applications. Scavenger [39] also reports test results evaluated using an image editor application, and their results show a maximum speedup of 38.7 and a maximum power saving of 24%.

4.1.2. Cost–benefit analysis

It is important to analyze the costs of offloading on to the cloud such as time, energy and monetary, versus monolithic execution/storage beforehand.

Walker et al. [41] discusses when a consumer should take the decision to offload storage to a remote cloud such as Amazon EC2. Their model for evaluating the benefits of leasing storage from a cloud service as opposed to buying hard drives, takes into account factors such as cost of electricity, cost of hard disks, disk power consumption, cloud storage price per GB, expected storage requirement, and human operator salary.

Li et al. in [42] propose a model with a suite of metrics to calculate the cloud cost. They consider two main costs of cloud computing; namely, Total Cost of Ownership (TCO) and Utilization Cost. Total Cost of Ownership (TCO) is generally used as a financial estimate to determine the costs attributing to owning and managing an IT infrastructure. With respect to cloud computing, TCO is deemed appropriate to function as a basis for providing

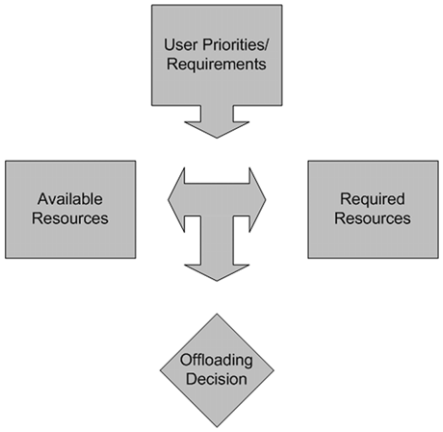

Fig. 6. Cost–benefit analysis and decision making in a mobile cloud.

an estimate for the commercial value of cloud investment, and takes into account server cost, software cost, network cost, support and maintenance cost, power cost, cooling cost, facilities cost, and real-estate cost. Utilization Cost refers to the actual resources being consumed by a particular user or application according to the dynamic demand. Because of the 'elasticity' in a cloud, the amount of resources such as servers, software, power and facilities including UPS and battery system, can be dynamically added or removed as per demand from the resource pool. Therefore rather than statically summarizing the cash outlays, cloud cost analysis must consider the impact of elastic utilization. Furthermore, Li et al. argue that virtual machines are the unit of resource in cloud, since virtualization is widely adopted in cloud computing. Therefore VMs are considered as the inputs in a three layer derivation based method that also takes cloud amortization into account.

Although some of these issues such as power consumption and cloud pricing are applicable to mobile cloud computing as well, additional issues and perspectives need to be considered.

The importance of cost–benefit analysis. In a mobile cloud, because of the essential mobile hence dynamic nature, the resources are likely to change in any given moment. Therefore, a cost–benefit analysis is essential to weigh the benefits of offloading against the potential gain by evaluating the predicted cost of execution with user specific requirements as illustrated in Fig. 6.

Cost models using resource monitoring and profiling. Spectra [27] and Chroma [28] are two of the earliest cyber-foraging systems for mobile devices that employ methods to weigh the cost vs. the benefit of offloading to surrogate servers. In Spectra, contradictory goals such as performance, energy conservation, and quality are evaluated when considering if an application should be offloaded, and if so, on to which surrogate device. The performance, energy consumption and quality are predicted for each potential surrogate, and the selection is made by balancing the competing goals with each other so as to give the best possible trade-off. Since a pervasive environment is made up of mobile devices constantly changing in terms of resource availability, Spectra constantly monitors the resources such as CPU, network, battery, file cache state, remote CPU, and remote file cache state. Changes in these resources are taken into consideration when estimating the best placement. A 'self-tuning' approach is adopted to match the available resources with the application's resource demands since it is not practical for the applications to define a priori the exact resource requirements. Rather, the self-tuning method observes application execution and maintains profile histories for each known surrogate. These profiles are continuously updated, and consulted for future actions.

To estimate the best trade-off, the gain also needs to be predicted. For this, Spectra uses previous work done by Narayanan et al. [43] by using a prediction model based on the assumption that a resource consumption of an operation is similar to recent executions of similar operations. In the case of encountering a new operation for which history data does not exist, the model employs linear regression to give an estimate. Chroma uses a somewhat similar approach called 'tactics' specified in a declarative language, while building on ideas from the same earlier work Odyssey [44]. Chroma also employs resource monitoring and history based predictions to weigh the outcome of each tactic plan against the estimated cost. History data for applications are logged offline, and the predictors update the logs online and use machine learning to optimize the predictions. To come to a decision, Chroma performs a trade-off using utility functions that perform comparisons between attributes such as power consumption and speed. However, unlike Spectra, the adaptive policies are separated from the decision making and policy enforcement at runtime.

The Scavenger [39] framework also employs a cost assessment method to decide whether offloading should be done or not. This is carried out by the 'scheduler' component which considers the following factors:

1. Relative speed and current utilization of the surrogates: A CPU benchmarking method is employed to evaluate the performance of potential surrogate devices. Scavenger does not use the CPU clock speed values of devices since it would make little sense for comparison purposes due to the vastly heterogeneous nature of the device set. The benchmarking method uses the NBench suite[11] to get a score that gives an assessment of the strength of a device. The number of tasks running in the potential surrogate is used as a measurement of current resource utilization.
2. Network bandwidth and latency to the surrogates: Network connection information is statically configured to avoid unnecessary traffic using and this information on expected bandwidth, Scavenger estimates the cost of remote execution.
3. Task complexity: This does not refer to the Big *O* asymptotic time, rather this means an estimation of how much time it would take to execute the task on the surrogate device. History data recorded whenever a task is performed is used for this estimation similar to Spectra and Chroma. However, unlike Spectra and Chroma, for each task Scavenger employs a 'dual profiling' technique where not one but two profiles are recorded per task; a task-centric profile using global task weight and a peer-centric profile for each device–task pair. The peer-centric profile is consulted first, and if the peer is a hitherto unknown device, the scheduler looks into the task-centric profile. Eq. (1) shows the calculation of global task weight, where T_{duration} is the time taken to complete the task, P_{strength} is the NBench score, and P_{activity} is the number of other tasks being run on the device at the time.

$$T_{\text{weight}} = T_{\text{duration}} \frac{P_{\text{strength}}}{P_{\text{activity}}}. \tag{1}$$

4. Input and output size: Input size is available at runtime, but the programmer needs to specify the information on output size.

MAUI [24] carries out a cost–benefit analysis by profiling each method in an application through serialization. Measurements of network bandwidth and latency are also taken to incorporate into the cost. Specifically, MAUI's profiler takes three issues into consideration:

[11] http://www.tux.org/mayer/linux/bmark.html.

1. The devices' energy use: An energy profile of the mobile phone is prepared by first measuring the battery usage from a hardware power meter. A JouleMeter [45] style benchmarking system is used to measure the CPU utilization. Initially the profiler is 'trained' on the device by collecting data on CPU utilization and energy consumption. These values are used to construct a linear model that is able to estimate how much energy will be consumed by a method. This estimate is given in the form of the number of CPU cycles it requires to execute that particular method. They report that the model's mean error is less than 6% with a standard deviation of 4.6%.
2. Application characteristics: Since profiling an application for each of its possible paths will be expensive, MAUI uses the past invocations of methods to predict the future invocations.
3. Network characteristics: They follow a simple procedure to measure network throughput. 10 kB of data is sent to a MAUI server and is observed to obtain an idea of network bandwidth and latency. Also, the profiler records the statistics of network quality every time a method is offloaded, and this history data is utilized for future estimates.

The data from the profiler as mentioned above is then fed into the MAUI 'Solver' to decide if a method should be executed locally or remotely. The Solver tries to give the best possible partitioning strategy that will give the least amount of phone battery consumption.

Clonecloud [36] employs a 'Dynamic Profiler' to collect data used in the cost–benefit analysis, that is then fed in to the 'Optimization Solver' to decide which methods needs to be migrated, such that the cost of migration and execution will be minimized. Here, the cost could refer to execution time, energy consumption or resource footprint.

In [46] Zhang et al. take four attributes into consideration when calculating the cost of migrating mobile apps to the cloud: power consumption, monetary cost, performance attributes, and security and privacy. These are inferred from various sensing modules in the mobile device and cloud, that monitor data such as battery, network, device loads, cloud loads and latency. After processing these inputs, the cost model decides on a suitable course of action such as, migrating apps to the cloud/mobile device, switching between different networks and allocating cloud resources.

Cost models using parametric analysis. In [47], Kumar and Lu provides an analytical model for comparing energy usage in the cloud and the mobile device. The model takes the following parameters into consideration; the speeds of the mobile device (M) and the remote cloud (S), the number of instructions of the computation (C) (assuming both mobile and cloud versions have the same number of instructions), the number of bytes to be transferred (D), network bandwidth (B), the energy consumed by the mobile device in idle (P_i), computing (P_c) and communicating (P_{tr}) states. Assuming the cloud is F times faster than the mobile device, they infer the amount of energy saving to be given by the Formula (2).

$$\frac{C}{M}\left\{P_c - \frac{P_i}{F}\right\} - P_{tr}\frac{D}{B}. \tag{2}$$

When the formula gives a value greater than zero, an energy saving is possible, i.e. $\frac{D}{B}$ should be lower compared to $\frac{C}{M}$ and F should be sufficiently large. Thus, according to this model, offloading is beneficial in cases where heavy computation is needed with comparatively low amounts of communication.

Analyzing the conditions for optimal computation offloading, Wang and Li in [48] identify four kinds of cost factors; Computation cost, Data communication cost, Task scheduling cost and Data registration cost. These costs are expressed as functions of run-time parameters such as buffer size, input size and command line options. These are then fed into their partitioning algorithm to determine an efficient partitioning depending on the parameters.

Cost models using stochastic methods. For a mobile cloud service of the model given in MobiCloud [37] Liang et al. [49] proposes an economic mobile cloud computing model based on Semi-Markov Decision Process (SMDP) [50] for resource allocation. MobiCloud describes a system where mobile devices use application components named 'weblets' which can be either migrated to the cloud, or run on the mobile device itself. The SMDP model is based on three states in the mobile cloud; new weblet request or an inter-domain request, intra-domain transfer weblet request, and weblet leaves domain. When the cloud receives a request for migration from a mobile device, it will only accept it if there is an overall system gain. The overall system gain is based on maximizing the cloud profit and reducing the expense of the mobile user. The expense of the mobile user depends on the trade-offs of energy consumption in a mobile device vs. the monetary cost of offloading to the cloud. They argue that an intra-domain weblet transfer from one service node to another would usually generate more profit than a new weblet migration from a mobile device, or an inter-domain transfer in which the transfer happens from another cloud service provisioning domain. Besides the monetary gain, the overall system gain also takes into consideration the CPU cost in the cloud server due to virtual image occupation. A 'reward model' is used to calculate the costs based on the system state and its corresponding action.

Discussion. Existing cost models in current mobile cloud computing systems mainly fall into three categories: history based profiling, parametric, and stochastic. An overview of these are given in Table 1.

Spectra and Chroma have two of the oldest history based profiling cost models, and are quite similar, with Chroma being a result of lessons learned from Spectra. Several later works build on concepts from these two lines of research while adding new methods to address their shortcomings. For example, the cost models of aforementioned work and MAUI rely on the assumption that past invocations of the same, or a similar operation is a good indicator of its current resource usage. However, the energy consumption of an operation in MAUI is expressed as a function of the number of CPU cycles it requires, while in Spectra, energy measurements are directly taken from the mobile device's battery. Scavenger also records history data and maintains profiles similar to Spectra, but optimizes the concept by recording dual profiles per task.

While a thorough cost–benefit analysis is crucial for optimal performance, the cost of cost analysis itself should not be overlooked. This issue is discussed in MAUI, where the overhead of performing frequent profiling and accurate estimations based on latest data, are balanced to give a positive outcome.

4.1.3. Mobility management

One of the key issues encountered in a mobile cloud is the design of intelligent mobility management techniques that support user mobility while providing a seamless service. Although research has been done in location management in wireless networks [51], here, we focus on the methods followed to manage and support mobility in mobile cloud computing systems. Determining a device's current location is helpful to assess its potential to move away from or towards the active mobile cloud. Work on localization primarily falls into either *infrastructure based* techniques, or *peer-based* techniques. Infrastructure based methods use technologies such as GSM, WIFi, ultra-sound with RF, GPS, RFID, and IR. Among these, current GPS very rarely works indoors, and the more accurate techniques need additional infrastructure and need dense deployment of access points as well.

Table 1
Overview of cost models in mobile clouds.

Name and type	Objective	Resource monitoring	Benchmarking	Assumptions
Spectra [27], Chroma [28]: History based profiling	User specifiable; reduction in execution time, energy usage, and increasing fidelity	Yes: CPU, network, battery, file cache state, remote CPU, memory usage and remote file cache state	No	Resource usage of an operation will be similar to the amount used by recent operations of similar type.
Scavenger [39]: History based profiling	Improve performance and energy saving	No	Yes. Uses the NBench suite.	Task duration is proportional to the NBench rating such that a task that takes 1 s to perform on an idle surrogate with an NBench rating of 40, should take about 2 s to perform on an idle surrogate with a rating of 20.
MAUI [24]: History based profiling	Primary goal is saving energy. Speed is also considered.	Yes: CPU, network bandwidth and latency	Yes. Uses a method similar to JouleMeter.	Past invocations of a method can be used as a predictor of future invocations concerning energy usage.
CloneCloud [36]: History based profiling	User specifiable; reduction in execution time, energy usage.	Yes: CPU, network, storage	No	All objects have the same cost per byte.
Analytical model by Kumar and Lu [47]: Parametric model	Saving energy	Yes: network bandwidth, energy consumption	No	The number of instructions in the task is similar in the cloud version and the mobile version.
Analytical model by Wang and Li [48]: Parametric model	Improve performance and energy saving	No	Yes	–
MobiCloud [37]: Stochastic method	Saving energy and minimize monetary cost	Yes: CPU, battery state, network	No	–

Furthermore, these methods are energy consuming and hardly suit the conservative needs of mobile cloud devices. In contrast peer-based techniques are better suited to manage the mobility of participating devices, considering that *relative location* information is adequate, and most can be implemented with short range low power protocols such as Bluetooth.

Peer based techniques for determining the position of a mobile device. In 'Escort' [52], a human localization system using Mobile Phones is presented. Without using GPS or WiFi, the position of a person is inferred using social encounters between users via audio signaling, and monitoring the walking traits of different individuals via phone compasses and accelerometers. Their experiments in parking lots and university premises show that a user can be brought to within 8 m of their target using this technique. Each user's movement is captured by his/her mobile phone by using its compass and accelerometer, and when it encounters another phone, the encounter is recorded in along with the timestamp. These movement traces are then used to construct a global view of users positions and paths. For example, if A wants to locate B, 'Escort' creates a route composed of various encounters. If A had met C recently, and C had met B, the route is first calculated to the point where A met C, then to the place where C met B. Of course, since there could be many possible paths, the 'Escort' server will select the optimal one. Although this system deals exclusively with electronic escorting, with the existence of a dedicated 'Escort server', the idea behind this localization method can be used to determine the position of a participating mobile node in the mobile cloud. For example, in the case where a delegating device is about to assign work to a worker device, or if it is waiting for results from a worker device, it will be beneficial to know if the aforementioned worker is moving away, i.e., about to disconnect. Not using GPS or WiFi, which are both battery draining methods, is especially useful as well. In the case of 'Escort', fixed beacon transmitters placed at random locations are used to correct errors in routing, since they deal with fairly long distance paths. However, for the purpose of mobile cloud, where devices operate in relatively close range, such error correction methods will not be needed.

In 'Virtual Compass' [53], short range protocols such as Bluetooth and WiFi are used to construct a two dimensional representation of nearby devices. Peer-to-peer messaging is used to estimate the distance via signal strength, and to pass information about each device's neighbors and their distances. The latter information makes it possible for devices to gain knowledge about nodes that are beyond the immediate range. In the context of the mobile cloud, this is beneficial to assess the potential of new devices about to join or previous nodes returning to the cloud environment. Furthermore, since communication in the mobile cloud is most likely to happen via short range protocols such as Bluetooth and WiFi (since it uses nearby devices as resource providers), passing information related to this techniques will not necessarily add an extra burden. In fact, localization data can piggy-back on the general cloud communication messages. One challenge in mobile cloud is to trade off between the possibility of energy drain with continuous scanning, opposed to missing out on encountering potential resources without. A solution exists in the method followed in 'Virtual Compass', where it employs a self-adaptive scanning technique that regulates its scanning intervals by keeping track of changes in its neighbor graph. This regulation makes use of a central server however, where ideally a decentralized solution is preferable in the context of mobile cloud.

One such decentralized implementation is 'Friends Radar' [54], which employs location updates in peer-to-peer fashion using XMPP. Friends Radar differs from the previously discussed systems in that only 'known' contacts, or friends' locations are visible, and that it uses GPS. In a situation where the mobile cloud is comprised opportunistically with random unfamiliar devices, and conducted indoors, this method will fail. However, in cases where all participating devices are pre-known and trusted, such as work sharing among a group of friends for example, and the computation is being done outdoors, this can be viable solution. However, using only known contacts can also be a plus point in terms of privacy and security. As the authors have mentioned, this work would be more applicable in terms of mobile cloud, if it is extended for indoor use using signal strength techniques instead of GPS only.

Similar peer-based localization methods such as NearMe [55], and Beep Beep [56] also exist, but these do not work for more than two nodes. DOLPHIN [57] needs to have special ultrasound hardware, that cannot be expected to exist on a normal phone.

Managing mobility via fault tolerance methods. In [12], the location and the number of surrogate devices are important since the cloud's objective is to provide support for users with similar goals. They argue that users in similar locations tend to share similar goals. Hence, a context manager tracks potential and existing surrogates corresponding to people moving in groups.

Also, the number of devices in the vicinity is needed in the scaling of applications. Using an ad hoc discovery mechanism, a p2p component monitors the resource pool, and notifies the context manager if a change occurs; i.e. new devices come in, or existing ones leave. The authors mention that they hope to use mobility traces to establish 'communities' moving together that are stable environments for task distribution. Furthermore, mobility tracking is an important part of fault tolerance. If an unstable node could be identified beforehand, the system could take precautions by promoting task redundancy.

Mobility is one reason in mobile clouds for disconnectivity, and disconnection, similar to hardware failure in a distributed system. In Hyrax [14] disconnectivity arising from the mobile nature of devices is handled through the fault tolerance mechanisms of Hadoop. As given in [40], one of the main assumptions of Hadoop and the HDFS architecture is that 'hardware failure is common'.

In [58], the availability of a mobile device as a resource provider is determined by its mobility. Hence, devices/users with a high degree of mobility are termed as less reliable due to them being prone to disconnection. To predict the 'availability status', recordings of changes in mobile resources over time are used with a Markov chain model. For example, users are classified into three groups: low mobility users, middle mobility users, and high mobility users, and this is used to calculate the probability of mobility. Although this is sound in a contained environment such as a workplace or university, where information about user mobility patterns is known, for a mobile cloud operating in a public area with little or no prior information, this method will not be viable.

Supporting mobility via component and proxy migration. MoCA (Mobile Collaboration Architecture) [59], is a middleware for collaboration on context-aware mobile devices. MoCA currently works with 802.11 and follows the client–server model with a framework for implementing application proxies and basic services for collaborative applications. The servers and proxies run on static networks, while clients run on mobile devices. A proxy acts as the intermediary between the client and server. Here, user mobility is supported by monitoring the locations of the users and switching to an application proxy more suited to the new location. Mobile clients query the service—which contains registered information on available application server proxies—and discover the means to access a collaborative service at their closest proxy. The 'closest proxy' is determined by the location of the mobile device which is inferred using its radio frequency signal pattern as done in project RADAR [60]. Measurements gained at predefined reference points are used for comparisons.

Hydra [61] facilitates developing distributed mobile applications in a pervasive environment by the construction of a virtual computer through the participation of a networked set of pervasive computers so that the application satisfies a mobile user's requirement changes based on location, current tasks and number of people. User mobility is supported by moving the mobile agent based software components to other servers situated along users movement. In cases when components are dependent on each other and the moving of one may affect another, this component migration is carried out by way of 'hooks', which dictate two types of policies; either a component will be able to 'follow' another, or replicate itself and make the replicate follow. In the second policy, the clone becomes independent of the source component. Which of the two policies to follow is decided on by the component itself. Using RFID, spatial regions such as parts of a room or a building can be identified within a meter. The positions of objects (mobile devices) are identified by identifying these spatial regions that contain them.

In [62], a mobile service cloud based on an overlay-based distributed infrastructure is presented. This is an extension of previous work in Service Clouds [63] where an overlay-based network supports dynamic and on demand prototyping and deployment of services. Here, mobile devices connect to overlay hosts who implement services on the wireless edge. User mobility is handled by migrating the proxy service to different locations following the user. This proxy migration is done to maintain quality of service, minimize resource consumption, and ISP policy which may not allow mobile devices connecting to access points belonging to another provider.

Discussion. A majority of mobile clouds support mobility through component and proxy migration. Although this works for mobile devices connecting to remote servers, it is not a viable mechanism in the following cases; where mobile devices are resource providers themselves and are moving in ad-hoc manner, and where the mobile device offloads tasks to a local resource provider such as a cloudlet. A potential solution for the cloudlet model is to use the same technique as in 'Follow Me' [64], a localized and decentralized location sharing system using PlugComputers as Bluetooth scanners. As mentioned in [12] keeping track of other mobile resources moving together with the client to form 'communities' is the only solution proposed so far in a mobile cloud of this type, and even that has not yet been fully implemented.

4.1.4. Connection protocols

The current mobile cloud computing research uses a variety of connection protocols for communication including WiFi, Bluetooth, and 3G, though the majority has employed WiFi for many reasons.

WiFi. WiFi (wireless Ethernet 802.11b) and Bluetooth both operate in the unlicensed 2.4 GHz ISM band. WiFi was initially intended as replacement for cabling for resource and peripheral sharing (such as printers, shared storage devices) among PCs, terminals etc. for wireless local area networks (WLANs). WiFi has a longer range, with a radius within 100 m and supporting up to 11 Mbps data rates.

Bluetooth. Bluetooth on the other hand, was intended for nonresident equipment and applications such as wireless headsets etc wireless personal area network (WPAN), and is characterized by its low power requirements and low-cost transceiver chips [65]. The range for Bluetooth is typically in a radius of 10 m, depending on the device class, power, and physical obstacles in the environment. However, according to Bluetooth specifications, future versions will be faster up to 24 Mbps and consume less energy.[12]

3G. 3G (third generation mobile telecommunications) is a technology for mobile service providers and it shares the basic business model with that of the telecommunications services model. The infrastructure is owned and managed by the service provider and sold to customers typically on a monthly usage basis. Although the focus of cellular technology has been voice telephony, data services has also started to attract attention. Mobile broadband access of several Mbps is available via recent 3G releases such as 3.5G and 3.75G [66], although this is substantially lower than the data rate of WiFi.

Experimental results. Based on the experimental results presented in related research on mobile clouds, the energy consumption of 3G is shown to be higher than WiFi [24], though data for similar statistics for Bluetooth exists.

In MAUI [24], the mobile phone using 3G to offload work to a remote server consumed three times as much energy as WiFi with a 50 ms RTT, and five times the energy of WiFi with a 25 ms RTT, meaning that downloading a 100 kB file repeatedly over 3G will deplete the battery in less than two hours.

[12] https://www.bluetooth.org.

Table 2
Overview of connection protocols used in mobile clouds.

Protocol	Frameworks	Pros	Cons
WiFi	Spectra [27], Chroma [28], Cuckoo [30], Cloudlets [23], MAUI [24], CloneCloud [36], MobiCloud [37], Hyrax [14], Virtual cloud [12], Scavenger [39]	Better range than Bluetooth (100 m), better performance and lower energy consumption compared to 3G	Inter-operability issues between brands or deviations can cause limited connection or lower output speeds, security threats
3G	Cuckoo [30], MAUI [24], CloneCloud [36]	Near-ubiquitous coverage [24]	Round-trip times over 3G are lengthy and bandwidth is limited, poor performance and high energy consumption
Bluetooth	Cuckoo [30], MMPI [32]	Low power usage [67], widespread availability as opposed to other protocols [68]	Limited range (10 m)

In CloneCloud [36], experiments conducted on three applications with WiFi displayed a latency of 69 ms and bandwidth of 6.6 Mbps, while offloading with 3G resulted in a latency of 680 ms, and bandwidth of 0.4 Mbps. Concerning speedups, WiFi gave speedups of 12×, 20×, and 10× while the energy consumption was 12×, 20×, and 9× less energy than the monolithic application. However, test results of 3G offloading gave lower gains; 7×, 16×, and 5× speed-up, and 6×, 14×, and 4× less energy for the same applications respectively. Greater latency and lower bandwidth of 3G is given as the cause of this.

An overview of connection protocols used, and their advantages and disadvantages regarding a mobile cloud is illustrated in Table 2.

4.2. End user issues

End users issues relate to issues that directly involve users such as incentives for participating, interoperability and cost. When using a mobile cloud, one of the key challenges experienced by the end users is the transaction infrastructure. In particular, we hope to answer the following questions in this section; in what ways would users of mobile cloud services be billed? In cases of collaborative mobile clouds, how is credit represented? And how can users be persuaded to contribute to the resource cloud? What are the presentation and usability issues that need to be addressed for mobile cloud services?

4.2.1. Incentives to collaborate

In cases of mobile devices themselves acting as resource providers as discussed in Section 3.2 and in works such as Hyrax [14], the participating devices need to have incentives as to 'loaning' their resources. Furthermore, there need to be mechanisms to prevent 'free riding'.

In [12], users are enticed to participate in sharing their mobile resources by 'common goals'. If many users need to execute the same task, it can be partitioned so that each user only has to do a small part. The result of the task will be shared among all the participants. For example, consider the case of a group of people performing an activity together, such as visiting a museum. Say someone is interested in translating a foreign text on an exhibit, but his/her phone does not possess the capabilities or resources to process such a task. Connecting to a remote server via the Internet would mean paying for data roaming. His/her solution is to collaborate with other people in the same group who would also be interested in translating the aforementioned text. The authors argue that people sharing the same location are also likely to share common objectives, thereby providing them with an incentive to share their resources.

For the same kind of collaborative cloud computing, monetary incentives can also be considered in the means of micropayment schemes as discussed in [69,70]. In [71], the use of monetary and social incentives for mobile distributed systems based on opportunistic networks have been discussed. Their focus is on message transmissions in delay tolerant networks (DTN) formed by typical mobile devices. The monetary incentives are proposed to be implemented in the form of prioritizing the payload in the order of importance. Thus, a selfish mobile host will only relay a message with a certain priority or higher since it implies the sender is willing to pay a price for successful delivery. The authors argue that, as the mobile hosts become more and more selfish, the delivery rate of high priority messages become higher while the low priority message delivery rate slowly decreases. However, this result depends on the high to low priority ratio as well, since if all messages have a high priority, there will be no discrimination.

The social incentives are based on the premise that even a selfish host will have a set of social relationships, and hence, will not display the same behavior towards all the other hosts. The authors surmise that since the selfish host will also need to send messages using other hosts at times, it is in the selfish host's interest not to ask for monetary payment from hosts in its work group/community. They also implemented a simulation where each host records encounters with other hosts. Whenever a selfish host receives a message from another host, it refers to these records and calculates the percentage of meeting days, and the standard deviation of the daily meetings. Based on this, the selfish hosts display an altruistic behavior towards hosts they meet often, and the authors show that these hosts are better off than the hosts who are always selfish.

Other methods include enforcement schemes employed in peer-to-peer file sharing systems to control free riding [72].

4.2.2. Presentation and usability issues

Although there is lack of focus in this issue in mobile cloud computing research, presentation in the user interface does pose a valid challenge. This has been frequently discussed in mobile computing research [73–76], and lessons can be drawn from these to apply for mobile clouds as well.

User interface issues in mobile computing. Mobile devices span a large number of heterogeneous platforms. To design and develop separate user interfaces (UIs) for each and every type of device would be highly inconvenient and unrealistic for the UI developers [74].

Since the display area of a mobile device is small, which information and the way to present it to the user is a problem. Applications designed for mobiles may interact differently with users from normal desktops, for example, less data entry, and using popup menus to conserve the meager screen space [75].

Furthermore, user interaction methods with the application may depend on the user's context such as location, bandwidth and remaining energy [76].

In addition, users of mobile cloud computing frameworks would also need some user level controls in the interface specific mobile clouds. For example, users may need to specify constraints, select from available surrogates, define their priorities, and deal with cost negotiation.

4.3. Service and application level issues

Service and application level issues relate to the factors concerned with performance measurements of the system, and

the QoS of the system. For example, in what ways do mobile cloud computing systems ensure availability? What are the fault-tolerance (FT) mechanisms employed to ensure smooth execution and uninterrupted service? Cloud APIs providing libraries to support cloud application development for mobiles are also discussed.

4.3.1. Fault-tolerance for meeting availability requirements

Fault-tolerance is a highly important aspect in a mobile cloud, even more so than a conventional cloud because of the mobile nature of the devices, i.e. "mobility is inherently hazardous" [2]. Disconnection can happen due to user mobility as devices enter and leave a network. Running out of battery power, network signal loss, or hardware failures are other common factors.

Redundancy. The FT support in Hyrax [14] comes from the FT mechanisms of Hadoop on which Hyrax is based. Hadoop recovers from task failure by re-execution and redundancy. If node failure is anticipated, the task is replicated on another node/s that is deemed to be stable. In their testing and evaluations, where applications such as Sort, Grep and Word Count were ported, it was found that Hyrax was able to recover more effectively when the number of nodes was higher.

In [12] although FT is not implemented, it is mentioned as future work, where the authors suggest using context-awareness for fault-tolerance purposes. Context information would be used to judge if a node is unstable, and if so, task redundancy could be carried out to increase the success level of task completion.

Proxy migration. In [62] FT is achieved by migrating the proxy service. In the event that a proxy node fails, its place is taken up by another node in the service cloud so as to ensure minimal disruption to the communication stream/s. This was tested using a testbed comprising PlanetLab nodes and hosts on a university intranet, and two strategies were implemented: on demand backup where another service is migrated as soon as system detects failure, and ready backup where a backup node is configured by default at the time of service composition. Of these two strategies, the ready backup was slightly faster, as can be expected, since the system only needs to reconfigure the relay to forward the stream in that case. The authors also suggest using the client middleware to trigger reconfiguration faster in future work.

Resource tracking. In [77], Palmer et al. proposes using the Ibis grid computing platform to address similar problems in mobile computing. The Ibis framework enables users to integrate their mobile devices onto the grid taking advantage of the grid's computational power. Here, FT is achieved by the Ibis system's resource tracking model using the 'JEL' API standing for 'Join, Elect, Leave'. As the name suggests the JEL API gives the system malleability, enabling it to adapt as new mobile nodes join and leave the network. The 'Join' operation notifies the application when a new node connects to the distributed system, thereby facilitating the applications to scale up. The 'Elect' operation is used to elect a node into the coordinating role. Whenever a node is disconnected, whether by choice or fault, the 'Leave' operation notifies the application and triggers an 'Elect' to select a new node to fill in.

4.3.2. Supporting performance at service level

A majority of Application Programming Interfaces used to build mobile applications targeted for cloud computing are based on service oriented architecture such as REST and/or SOAP. Mobile applications are able to connect to and request services hosted on a remote cloud through interfaces. However, mobile Web services need to consider additional constraints other than standard Web services: frequent loss of connectivity, low computational resources, and low bandwidth.

Web service caching. To improve the user experience that can be hindered because of disconnection, caching and prefetching has been proposed in research [78,79]. This approach enables the user to continue his/her work for a period of time while in offline mode. Furthermore, caching and prefetching also gives an increased response time. In [78], CRISP, a SOAP cache that can be embedded in client side application, or deployed as a separate proxy, is introduced. In [79] a dual caching strategy is proposed, where both the nomadic client and server have caches running, storing request–response pairs. Even though the overhead of storing these pairs is considerable over time, the performance gain is significant.

4.3.3. Cloud APIs

Mobile clouds have been implemented using APIs provided by distributed computing frameworks such as Hadoop [14] and Ibis [77]. Futhermore, there are cloud APIs catering to mobile devices as well. For example, the Funambol Cloud API[13] provides server and client side SDKs to develop mobile cloud applications and services that make use of images, calendar, contacts etc. stored in a Funambol server. Other open source APIs include Eucalyptus,[14] Nimbus,[15] and OpenNebula.[16] Commercial cloud APIs include frameworks such as Dropbox,[17] Azure,[18] Amazon and Google Apps.

4.4. Privacy, security and trust

Whether offloading intensive computations, or data storage, using the cloud for mobile devices does pose questions of security and trust issues. In her article in [80], Kharif outlines the potential pitfalls in using cloud services for mobile devices. Because of the low capacity of mobile device storage, many users are starting to store data such as contacts, calendars and SMS on clouds. However, these cloud services are stated to be vulnerable and users may lose their data if the services go out of business, or simply if the services fail due to technological problems.

Recent examples. For example, in the October of 2009, a large number of T-Mobile Sidekick users discovered that their personal data stored in the mobiles had disappeared due to a server failure [81]. The Sidekick data, such as users' contacts, photos and calendar appointments, was stored in a cloud service operated by the Microsoft company Danger. Although the majority of users did recover most of their personal data by November, this outage caused many difficulties, and raised many questions about how secure and robust the cloud really is.

Is cloud security applicable to mobile clouds as well? Mobile cloud computing inherits the security threats of conventional cloud computing in cases when the definition of mobile cloud means to connect mobile devices to a remote cloud. In this case, the remote cloud server would be the same as a conventional cloud computing provider, making the general cloud security threats valid. At the same time, mobile clouds present a group of issues that are particular to mobile devices offloading jobs through wireless communication channels. Furthermore, security concerns that are specific to mobile devices such as battery exhaustion attacks [82], mobile botnets and targeted attacks [83] should also be considered.

13 https://capi.forge.funambol.org/.
14 http://open.eucalyptus.com/.
15 http://www.nimbusproject.org/.
16 http://opennebula.org/.
17 https://www.dropbox.com/.
18 http://www.microsoft.com/windowsazure/products/.

4.4.1. General cloud security

In [84] Brodkin outlines seven security risks users need to consider in Cloud computing;

1. Privileged user access: offloading sensitive data to the cloud would mean the loss of direct physical, logical and personnel control over the data.
2. Regulatory compliance: the cloud service providers should be willing to undergo external audits and security certifications.
3. Data location: the exact physical location of user's data is not transparent, which may lead to confusion on specific jurisdictions and commitments on local privacy requirements.
4. Data segregation: since cloud data is usually stored in a shared space, it is important each user's data is separated from others with efficient encryption schemes.
5. Recovery: it is imperative that cloud providers provide proper recovery mechanisms for data and services in case of technological failure or other disaster.
6. Investigative support: since logging and data for multiple customers may be co-located, inappropriate or illegal activity should they occur may be very hard to investigate.
7. Long-term viability: assurance that users data would be safe and accessible even if the cloud company itself goes out of business.

4.4.2. Mobile cloud security

In addition to the aforementioned concerns, securing a mobile cloud introduces the following challenges as discussed in [85] where the authors propose a security model for elastic applications made up of 'weblets' that can be migrated to and from a cloud to a mobile device:

1. authentication between the weblets that would be distributed between the cloud and the device,
2. authorization for weblets that could be executing on relatively untrusted cloud environments to access sensitive user data, and
3. establishment and verification of trusted weblet execution cloud nodes.

Their security framework is based on the assumption that the cloud elasticity service (CES), including the cloud manager, application manager, cloud node manager, and cloud fabric interface (CFI), is trustworthy. The security threats are categorized as threats to mobile devices, threats to cloud platform and application container, and threats to communication channels. The authors propose a framework with the following security objectives: Trustworthy weblet containers (VMs) on both device and cloud, authentication and secure session management needed for secure communication between weblets and multiple instantiation concurrently, authorization and access control enforcing weblets on the cloud to have the lowest privileges, and logging and auditing of weblets.

MobiCloud [37] aims to provide a security services architecture for MANET clouds in three ways:

1. Acting as an intermediary for identity, key, and secure data access policy management: Identity management is supported by Attribute-Based Identity Management (ABIDM), which supports user-centric identity management schemes also known as Identity 2.0. They propose ABKM, a system for key management, which is an extension of identity-based cryptography. However, in ABKM, the Trust Authority (TA) generates private key components for each user depending on their public attributes, and the key exchange protocol is not required. Therefore, this is effective for delay tolerant MANETs where the source and the destination do not usually talk prior to sending the data.
2. Protect information belonging to mobile users by means of security isolations: MobiCloud has Virtual Trusted and Provisioning Domains (VTaPD), which are virtual domains enforced with resource isolation. A VTaPD contains various nodes corresponding to different physical systems. Nodes in the same VTaPD support the secure MobiCloud communication system when passing messages to each other. A cryptography based approach is used to enforce data access control and information isolation.
3. Assess risks by monitoring MANET status: the centralized data collection and processing in the MANET is used by the risk management service to identify malicious nodes and take preventive measures according to estimated risks.

4.4.3. Privacy

As the recent incident regarding CarrierIQ being installed and collecting information from mobile phones [86] shows, it is important for mobile phone users to have transparency and choice. Users need to be aware of what personal information is exactly visible to the public, and to have control over their personal data that is stored on their smartphones. It is vital that any personal data that is shared is done so with users consent, and that they can choose to opt out of any data collecting program at any time.

In [87], Fahramair et al. present the following requirements of a mobile and ubiquitous system that satisfies user privacy: protection against misuse, identification of pirated datasets, adjustment of laws (to provide additional security under certain circumstances), and ease of use.

These are valid requirements for a mobile cloud as well. In a mobile cloud where mobile device share work with other mobile devices, a primary concern is malicious devices. In a setting where the device users are unknown and the mobile cloud is formed opportunistically, this is a most serious concern. Although the Public Key Infrastructure (PKI) is an appropriate method for the security issue, the problem is that it draws a high operational overhead that is not practical on resource constraint mobile devices. Furthermore, the connections between the devices in a mobile cloud are highly dynamic, and adaptive. At a given moment, new devices may be joining while current devices may be leaving. In such a scenario, the frequency of user authentication requests will increase to such an extent that it could result in insufficient resources to perform asymmetric key operations and transmit heavy messages [87]. A solution to this problem is proposed in [88], in which PKASSO, a PKI based authentication protocol is introduced. To solve the resource constraint problem, PKASSO offloads the complicated PKI operations from the mobile device, to a remote resource rich server. Although this is a valid option in hybrid clouds that have connectivity to Internet, this is not viable in cases that long range connectivity is a problem. However, the cloudlet concept [23] is useful in this scenario: cloudlets could operate as the local infrastructure to which the PKI operations can be offloaded.

Techniques of anonymous routing such as onion routing can also be used to provide privacy for mobile nodes in a decentralized mobile cloud. Examples exist in the p2p domain, such as [89–91]. However, there are certain overheads and a risk of unreliable delivery associated with most anonymous p2p routing protocols [92]. As a solution, the degree of anonymity should be flexible and depend on the context. For an example, a mobile cloud operating in a public environment where the potential for malicious nodes are high, should have a high level of privacy, but this would incur higher transmission (e.g.: longer paths) and computation costs (e.g.: cryptography processing overheads).

In addition to an authorization scheme, users of the mobile cloud should also have the ability to change their privacy settings and dictate what information can be seen. For example, a mobile cloud participant may not want other devices to record his/her location information. In [93], the authors propose such a system, called the Privacy Rights Management for Mobile Applications (PRiMMA) project. PRiMMA's key objectives are to provide the users with a tool to control and add privacy policies, resolve inconsistencies between user privacy policies, and predict the privacy requirements using monitoring mechanisms.

4.5. Context-awareness

Schilit et al. [94] describes the three important aspects of context as: the user's location, other users in the vicinity, and the resources in the user's environment. For example, in a mobile user's perspective, 'context' means things such as lighting, noise level, network connectivity, communication costs, communication bandwidth, and even the social situation.

Importance of context-awareness for mobile clouds. Systems with context-awareness are able to use contextual information to change and automatically reconfigure their configurations to adapt to the context [44]. This behavior is very useful in the case of mobile systems since these deal with an execution environment that is subject to constant change. In the case of mobile cloud computing, context awareness can be used in forming resource clouds as well as processing information. For example, a device can infer its location through GPS, Bluetooth, or some other forms of positioning and use that information to prepare itself for upcoming processing.

In the rest of this section, we review the use of context-awareness for mobile cloud computing systems, and also discuss a key concern of mobile device, energy awareness.

4.5.1. Context-aware service provisioning

It has been suggested that, mobile clouds can utilize the sensing abilities of their mobile devices such as location, acceleration, etc. and act as providers of context awareness/information. In [95] the authors suggest utilizing the sensing capabilities of mobile Internet devices to provide such context-aware service provisioning. Consider a mobile device connected to a remote cloud service through the Internet. As the context of the user changes, this prompts invocation of different cloud services based on the current context. With this kind of context-awareness, a service would not be bound to a user. Instead, when a mobile user invokes a cloud service, the request is accompanied by his/her context information, and the most suitable service is selected based on that information. Therefore, context is used to provide personalized services, and also as fault tolerant mechanisms such as rectifying low quality of service problems. Here, the authors identify a model consisting of four layers of context elements:

1. Monitored context: refers to current monitored context consisting of: device context which includes the environmental and device settings, user preference for user-specific preference settings, such as those regarding services selection and invocation, situational context relating to monitored data on user location, time etc., and Service Context information such as QoS.
2. Types of gaps: refers to gaps that happen as a result of content changes. For example, when the user's service is changed from Service1 to Service2 owing to a context change, there is a gap between the two services. Mainly, two types of gaps are identified: gap on functionality that relates to an available service and the needed service, and gap on nonfunctionality relating to differences in QoS values between the previous service and the current one.
3. Types of causes: refers to the factors that can cause the aforementioned gaps. A service may have multiple interfaces, and an interface may have different implementations. Similarly, there can be different component instantiations for the same service component. The gaps arise because of the mismatches between these: Service-level Unmatched, Service Interface-level Unmatched, Service Component-level Unmatched, and Component Instance-level Unmatched.
4. Adapters: refers to the remedial actions that should be taken to remove the aforementioned causes.

Based on this model, the authors propose a context aware service provisioning architecture consisting of three tiers: User Layer that is made up by the mobile devices where the applications run on, Agent Layer that adapts the services according to context, and Service Layer that deploys the services.

Volare [96] introduces a middleware for monitoring the context of a mobile device that is connected to a cloud service, and dynamically adapts the services so as to make them more resource efficient, reliable and cost efficient. Acting as an intermediary, VOLARE intercepts service discovery requests from applications while monitoring mobile device's context such as battery consumption, CPU usage, network bandwidth, user preferences like low power operation etc. Depending on this contextual information, VOLARE tries to adapt each service request by comparing the current QoS level with predefined thresholds. Instead of modifying the applications, a declarative language has been created to describe the adaptation policies. If at any time, the QoS level and the cost changes beyond the predefined values, VOLARE will automatically rebind to another service that can satisfy the requirements.

In MoCA [59], a proxy can register an 'interest expression' on a mobile, i.e., "FreeMem$<$ 10 kB" for a particular client, which would result in the proxy getting notified if and when the client's free memory drops below 10 kB. These context specific "interests" depend on those specific applications requirements. For example, if the proxy gets an update that a client's wireless connectivity has gone down a certain point, it could take certain actions such as compressing the data.

Using 'Intelligent access' for Mobile Cloud Computing is discussed in [97], where use of context information provided by terminals, network nodes, or sensors deployed in the users environment enables efficient network access management across different Radio Access Technologies (RATs) such as GPRS, WCDMA/HSPA, LTE, WiMAX, cdma2000 and WLAN. Conventional intelligent access schemes assume that all categories of dynamic context information such as user profiles, terminal status and sensor information and external sensor networks would aid in improving mobile access. However, mobile cloud computing requires a wireless connection with a set of different necessities than classical heterogeneous access scenarios: connectivity for long periods, scalable bandwidth, network selection and usage based on energy costs. To satisfy these requirements, the paper proposes Intelligent Radio Network Access (IRNA). The status and attributes of each RAT is considered, while at the same time effort is taken to entertain the user requirements based on environmental factors. Their proposed context management architecture (CMA), based on the producer–consumer role model such as given in [98], is responsible for acquiring, processing, managing, and delivering context information. To control the supply of context information according to the mobile cloud's requirements, the framework has a Context Quality Enabler (CQE). The CMA is made up of three components:

1. Context Provider (CP): this is where the context information originates from and is provided to other components of the architecture. The communication between CPs and other components are done through context requests.
2. Context Broker (CB): acting as a middleman, the CB keeps a registry of available CPs and their capabilities, and provides a CP look-up service. Also, the CB itself is able to forward the data it receives from CPs.
3. Context Consumer: these are the entities that take context data as inputs for their actual functionality, e.g.: network services, applications for end users, and service enablers.

In MiPeG [25], a middleware for integrating mobile devices into grid environments, several mechanisms for providing and using context awareness are discussed. Context information such as location, resources and environment conditions is used to

adapt the services provided by the grid, using Semantic Web technologies, thus supporting the 'pervasive grid' concept. The 'Context Reasoning Engine' forms the main component of the context service. Its responsibilities include handling the context, information gathering from sensing devices, and forming higher level semantic rules depending on the context information received.

4.5.2. Risk assessment using context-awareness

In MobiCloud [37], context information is used to facilitate risk assessment and routing decisions. MobiCloud introduces Virtual Trusted and Provisioning Domains (VTaPDs), which is a service that can isolate different information flows in in different domains by way of programmable router technologies. VTaPDs identify these separate flows and create virtual domains. By doing this, a user is able to securely run multiple applications on different security domains, and to separate services for different settings based on context. The MANET's contextual information such as device sensing values, location, and neighboring device status are recorded by the VTaPD manager, and used for risk management and intrusion detection procedures. Parameter values related to devices, network, content, and security such as battery level, connectivity, predefined goals, and privacy, are used to provide context-aware service migrations. Risk management is aided through context information because the status of the entire system (end-to-end communication delay, reachability of the destination, security status of each mobile node, etc.) is available. From this centralized data collection and processing, knowledge of the full MANET system is gained, and MobiCloud can easily identify malicious nodes.

4.5.3. Identifying potential resources and common activities using context-awareness

In [12], a Context Manager component is deployed to sense context information and store it to be used for other components such as Application Manager, which launches, intercepts and modifies an application according to the current context. Location and number of nearby devices in the vicinity are the key contexts, with location information used for mobility traces, and number of devices used to aid the forming of the mobile cloud. Thus, the system is made aware if a new device enters the resource pool, or leaves it, thereby leading to better scalability and content distribution. Furthermore, this information is used to infer if a particular device is 'stable' or not, which is essential to decide if it is following a common movement pattern with the other devices, leading to common activities.

4.5.4. Energy awareness

Because a mobile device operates on a finite supply of energy contained in its battery, energy is one of the key resources that needs to be used carefully [99]. In the context of mobile clouds, the cost of participation (such as power consumption) should be less than the benefit gained [100]. Also, it will enable the mobile device to take appropriate component level action to minimize unnecessary energy consumption, and hence, lengthen the system's life span by unloading unneeded software components, redeploying energy intensive components to more resourceful hosts, and collocating frequently communicating hosts, as suggested in [101]. For these reasons, being aware of a device's energy usage is vital.

In the following we discuss the research on *energy consumption*, mainly focusing on work on energy profiling and energy usage estimation.

Energy profiling. In PowerScope [102] the authors present a profiling tool for mobile applications. The tool maps power usage to specific code components in applications and the operating system, allowing an analysis of power draining procedures. Using this analysis, the developers can modify their software to be more energy efficient. The authors report a 46% energy saving by using PowerScope to profile an adaptive video application run on the Odyssey platform [44]. Experiments were carried out with several laptops and pocket computers. Profiling is done offline after collecting data, to ensure no overheads are added to the analysis. A digital multimeter measures the electric current used by the profiling computer and the energy profile is generated using these correlated current measurements. The apps can only be profiled on an open source operating system however, as small modifications to the kernel are required.

In [103], Rice and Hay present a power consumption measurement framework, specifically for mobile phones. In particular, they explore the effect of message size and send buffer size when transmitting data on two Android 1.5-based phones. Power usage during connection to a WiFi network and idle power costs for WiFi, 3G and 2G are also discussed. Power consumption is measured by sampling the voltage drop across the phone battery and a high precision resistor. The mobile device first downloads the test script from a central server and uploads the results to the same server. The central server aligns the test results with the traces and logs. Their findings can be summarized as follows:

1. When connecting to a WiFi network through DHCP, a significant portion of the time and therefore power, are taken by the ARP Probe packets and the delay between them.
2. In the case of idle power, WiFi has the lowest energy cost, followed by 3G and 2G respectively. However, it should be noted that the locations of the base stations will also affect radio transmission.
3. Although it would be logical to expect that the energy cost per byte will decrease as the message size increases, results show that this is not the case. There appears to be a sharp increase of power cost from sending a 7 kB message to 8 kB. However, the reason for this is not clear.
4. The choice of buffer size can significantly affect the power consumption.

The work discussed previously measures the energy via hardware. A different approach is to take the measurements via software to query battery levels as done in PowerSpy [104], implemented in the Windows operating system. PowerSpy operated in two stages; event tracking and analysis. In the event tracking stage, the application is run and tracked for CPU time, I/O activity and energy consumption. In the analysis stage, the data acquired in the previous stage is processed. To filter out the energy consumed by I/O activities, an estimation of energy usage by various devices to run particular tasks (such as energy used by the disk to read 1 kB) as specified by the devices manufacturer, is used. This estimation is subtracted from the total of recorded energy use, and the remainder is taken as the energy taken up by CPU threads. Next, energy consumption is drilled down to individual threads, on the assumption that CPU power usage is proportional to the number of CPU cycles spent on a thread.

Work done by Cano et al. [105,106] provides insight into the energy consumption of the Bluetooth protocol. The focus of their work is on the different states of Bluetooth such as Startup, Standby, Inquiry, Connection and especially the low power modes provided by the protocol.

Energy usage estimation. Work done by Chiyoung et al. [101] and Seo et al. [107] focused on the energy (electrical) consumption estimation of Java based pervasive systems at the level of its system level components. The initial estimation in [101] is done prior to runtime—during construction time. Then the estimation is refined during runtime automatically depending on certain system

parameters such as size of data exchanged over the network, inputs to the components' interfaces, and invocation frequency of components' interfaces. The exact formulas for calculating the estimation are given in [107]. Their methodology gives an Energy Cost Framework as follows:

- Overall Energy cost = Computational energy cost + Communicational energy.
- System Energy cost = Overall Energy cost + Overall infrastructure energy cost

where infrastructure cost is the cost incurred by the operating system. The computational cost is determined by the level of its public interfaces and the computational cost of an jth invocation of an ith interface on a given JVM is modeled in terms of byte codes, native methods and monitor operations. Communication cost due to the jth invocation of component c_1's interface I_1 on host H_1 is calculated in terms of the size of the transmitted and received data and the energy consumption of transmission/receive cost of H_1. The authors claim that based on their evaluations in which they ran distributed applications on Kaffe 1.15 JVM on Compaq iPAQ PDAs, their estimations are within 5% of the actual power consumption.

4.5.5. Discussion

Context-awareness has been utilized in mobile cloud frameworks for several tasks, including service provisioning with a better QoS, risk assessment, identifying resources and common activities. A majority of the works discussed use context information to provide a better service by personalizing the services, and providing fault tolerance through context such as user preferences, location, current QoS, network bandwidth, battery consumption, and CPU usage.

However, the performance and energy costs of employing the sensing capabilities and the trade-offs with benefits gained have not been discussed. Although certain contextual information such as monitoring battery consumption will not add an extra toll, other information such as location monitoring could prove to be too expensive.

4.6. Data management

For many cloud users and providers, managing data on the cloud raises many complications. In mobile cloud computing, as the name itself suggests, data that would traditionally be only accessible only to the mobile device's owner, would now be stored on, accessible to, shared with external devices or users. For many mobile users, this raises privacy and security questions. Also, data representation in mobile devices vary, and in a heterogeneous mobile cloud, this would lead to problems with portability and interoperability. Computations in a mobile cloud would be spread across a distributed file system, where multiple devices may need to access and modify files.

Furthermore, special considerations must be given to accessing files from mobiles over wireless networks. In a mobile cloud, users geographical locations are not fixed, and bandwidth must be conserved because of data access costs. However, it is also vital that mobile databases contain policies to safeguard against data loss while ensuring it conforms to mobility constraints. We discuss these issues in the following subsections. Issues regarding data privacy and security are not discussed here, since they have already been explained at length in Section 4.4: Privacy, Security and Trust.

4.6.1. Personal data storage on mobile cloud

One of the key concerns for people about using a mobile cloud is that their personal data on mobile device could be stored on, or accessed by the cloud. A mobile device contains contact lists, text messages, personal photos and videos, calendars, location information, and these data can reveal many things about someone's personal life. However, a personal computer also stores many such personal data such as photos and other multimedia, chat logs, emails, passwords, financial records or access to such records, calendar and contact lists. Today, thousands of monetary transactions are being done in online shopping sites such as eBay, and Amazon. Therefore, the risks involved in mobile cloud are not necessarily greater than those involved in traditional clouds. However, the issue here is whether the means of handling those risks have been properly implemented in the mobile cloud. Despite reservations, people do tend to use their mobile devices with the cloud. Some recent examples of mobile cloud storage are Apple's iCloud, Google Drive and Dropbox. Apple iCloud enables users of iOS devices to synchronize their application data such as photos, iTunes music, calendars, email, and messages. An initial 5 GB of iCloud storage is free for an Apple user, with additional storage available for a monthly fee. Although iCloud offers an Apple user an impressive user friendly feature suite (such as Find my Phone), it is only for data related to Apple devices. In contrast, Dropbox is less specialized, but works across heterogeneous platforms including Microsoft Windows, Linux, Mac OS, iOS, Android, and Blackberry. Dropbox also allows sharing stored data with friends, and file revisions.

4.6.2. Data access issues

Compared with traditional cloud computing, mobile cloud computing poses a challenge in the way mobile devices access data stored on the cloud. This is due to the inherent challenges of mobile computing such as low bandwidth, mobility and limited storage. I/O operations performed at the file level consume a lot of bandwidth, which is a problem for limited connectivity options in the mobile cloud [108]. Methods to minimize the I/O costs such as [109] exist, but do not take access methods into consideration. An approach more suited for mobile cloud is the concept of 'Pocket Cloudlets' [110], where a local storage cache based on nonvolatile memory is used to store parts or full cloud services in the mobile devices. However, this method needs the mobile device to decide a priori which portions of cloud services it will need to cache locally. Hence, the dynamic nature is compromised.

4.6.3. Data portability and interoperability

The mobile cloud will cater to many different mobile devices including Android, BlackBerry, and iPhone. In addition, these participating devices have various sensing capabilities, and as a result, contain sensor specific data as well. Since a mobile cloud will also have to communicate with typical cloud structures containing large scale servers, and PCs in addition to mobile devices, it is important that a platform independent representation is provided. Consider these cases regarding data interoperability in mobile cloud services [111,112]:

1. Palm Pre users were in dismay when updates from Apple's iTunes disabled the Palm Pre's ability to sync its multimedia with iTunes software.
2. For people using Funambol cloud services[19] with their iPhones, only syncing contacts were allowed by the Apple SDK through the official client. It was possible to sync calendars via the SQLite database, but in order for this to happen, the iPhone needed to be unlocked, which would make the warranty void.

Open cloud computing standards could be the answer to such data lock-in problems. An open standard would incorporate multiple cloud computing service providers to present a uniform interface. One possible solution for the mobile cloud is the Mobile Agent Based Open Cloud Computing Federation (MABOCCF) [113], where

[19] http://funambol.com/.

data and code are transferred from one device to another via mobile agents. When a user encapsulates code/task inside a mobile agent, there should be certain information in the data structure at the head of the agent such as, whether or not if it is a mobile agent, resources needed for that particular task, mobile agent code, and application code. Each mobile agent is executed in a virtual machine called Mobile Agent Place (MAP), and the mobile agents are able to move between MAPs, and also to communicate and negotiate with each other, realizing portability among heterogeneous cloud computing service providers.

4.6.4. Embedded mobile databases

Because of obvious limitations, mobile databases (or embedded mobile databases, as they are called) cannot possess all the functionalities of a traditional database. In the context of the mobile cloud, mobile databases need to be lightweight, and require the ability to download data from a remote repository and execute on this data even in a disconnected state, and also should be able to synchronize the modified data during the downtime with the enterprise whenever the network becomes available again. Furthermore, databases for mobile clouds need to have a quick start up time since faults in mobile devices can be more frequent than for a fixed host. Security constraints regarding access and real time processing are also important [114].

These are usually integrated with the operating system and specific application, and in essence are similar to traditional databases which could be relational or object oriented. However, mobile embedded databases are directly driven by procedure calls, as opposed to traditional databases which are not bound to the operational system or a particular application, and are designed for data storage in persistent media [114].

SQLite is popularly used by mobile platforms (e.g.: Android and iPhone) and web browsers (e.g.: Mozilla Firefox). Others include database systems such as MiniSQL, BerkelyDB, and Sybase.

Database technologies used in p2p systems such as PeerDB [115], and CouchDB[20] and recently CouchOne Mobile[21] derived from the peer-based CouchDB, can also be facilitated for the mobile cloud, especially in cases where mobile devices act as resource providers and form decentralized p2p connections.

5. Challenges

Based on the related literature, we find that the following issues have not been sufficiently solved. These are the gaps in the reviewed work that would prove to be directions for future work.

- Supporting continuous mobility while ensuring connectivity to the cloud: While mobile devices connecting to remote cloud servers to run apps such as Google translate can connect while mobile, this depends on the user's 3G connection. Even if the reception is sufficient, data costs and latency has a huge impact on these kinds of mobile cloud computing apps.
  The 'cloudlet' concept [23], presented to address the aforementioned problems could aid in this issue somewhat, but this does not fully support a mobile user who needs to work while on the move. Cloudlets, or other frameworks such as CloneCloud [36] that offload jobs to a local resource rich server could only support the needs of mobile device users who are actually stationary for the time being, such as waiting at a coffee shop, or at the airport, and are within range of such a resource rich server. Moving away out of range from the server, while the job is still being processed, would prove fatal to the task—unless there are such resource providers along the path of user's movement. This is highly unrealistic as of today, even in most developed countries. The need for additional infrastructure in public places is another problem in this method.
  Other research such as Hyrax [14] where mobile devices themselves act as resource providers, are promising, but there are gaps in supporting a decentralized ad hoc cloud mechanism. For instance, Hyrax relies on a central server that is responsible for task allocation. While distributing computing frameworks such as Hadoop [40] are useful in laying the groundwork for such frameworks, they do not fully support decentralized task scheduling. Unlike in distributed processing, thorough cost–benefit analysis needs to be carried out when considering mobile resources. Except in MAUI [24], the cost of cost analysis has not been evaluated in many works, even though this is a valid concern when considering the low resources on mobile devices.
  When supporting mobility and connectivity, some of the questions we need to contemplate are; How can a user device know of impending disconnectivity? In what ways can the most 'stable' and 'efficient' surrogates be chosen so as to ensure seamless connectivity? What fault-tolerance mechanisms can be employed to minimize potential failures?
- Security in mobile clouds: Although an issue of paramount importance, little research has been carried out in this regard. Users would need to feel confident when offloading their jobs to other surrogates such that their privacy would not be violated. Although many of the reviewed frameworks mention the need for security and trust, very few of them have actually implemented it and have left the implementation for future directions. As discussed in [23], consider a mobile user offloading a language translation program to a surrogate server, and a malicious VM manager distorting the translation with the intention of sabotaging the user's business transaction. As suggested in [23], avenues towards this need to be examined, possibly with trust establishment methods or reputation-based trust.
- Incentives for surrogates: If users are to be persuaded to collaborate and share their resources with others, there needs to be motivation either through monetary or social incentives to do so. An interesting method is using common goals [12], but in the absence of common activities this will not prevail. In the case of monetary incentives, several questions need to be answered such as: how is credit represented in a mobile cloud? how will monetary transactions proceed in a secure method? how will the price of resources be decided? Using social incentives such as suggested in [71] also raises challenges such as preventing free riding and enforcing standards.

5.1. Proposed architecture

For an operational mobile cloud, that exploits the locally available mobile resources, while ensuring user privacy, security, and operates at optimum cost, we propose an architecture such as shown in Fig. 7. Our proposed system is mainly composed of five components; Job handler, resource handler, cost manager, privacy and security manager, and context manager. The resource handler shall be responsible for searching for and discovering other mobile resources, connecting, maintaining connections and communicating with the external mobile devices, and also monitoring them for potential nodes entering or leaving the cloud. Due to the high probability of disconnectivity, the resource handler must opportunistically exploit available resources, while ensuring that the system gain, privacy, and security is also not compromised. The cost manager must determine the user priorities (e.g.: battery conservation, fast execution, monetary gain) and by taking

[20] http://couchdb.apache.org/.

[21] http://www.couchbase.com/press-releases/couchio-becomes-couchone-launches-couchone-mobile.

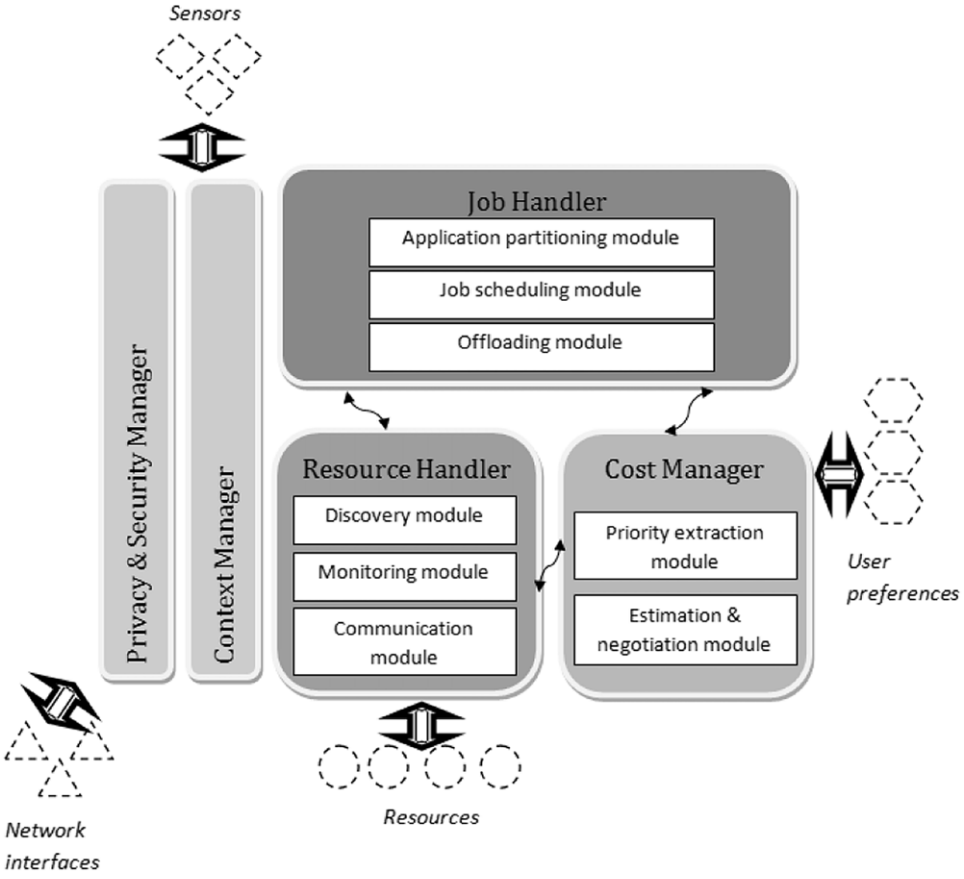

Fig. 7. A generic architecture for implementing a mobile cloud.

into account the job at hand, available resources, and required resources, come to a decision whether to offload or not. The job handler dynamically partitions the application and or data set required, offloads the generated jobs, and maintains the job pool. Of course, for these modules to function, intercommunication is needed. For example, the cost model needs inputs from the resource handler about the capabilities of the available resources, and the resource handler needs sensor inputs from the context manager in resource monitoring, to manage mobility inside the mobile cloud. The privacy and security manager needs inputs from sensors and user in addition to network interfaces to determine the most suitable policy to enforce. For example, the strict authentication settings needed in a public setting will not be needed when operating among a group of trusted devices such as among a group of friends or family. Or the user might decide on less resource consuming privacy policies since his or her other requirements such as conserving battery are prioritized. Here, we highlight a key characteristic of the mobile cloud: adaptability. Adapting according to the user's requirements, context, and system resources is vital in order to ensure cost efficiency.

6. Conclusion and future work

Mobile cloud computing aims to empower the mobile user by providing a seamless and rich functionality, regardless of the resource limitations of mobile devices. Although still in its infancy, mobile cloud computing could become the dominant model for mobile applications in the future.

We have given an extensive survey of current mobile cloud computing research in this paper. Highlighting the motivation for mobile cloud computing, we have also presented different definitions of mobile cloud computing in the literature. We have presented a taxonomy of issues found in this area, and the approaches in which these issues have been tackled, focusing on operational level, end user level, service and application level, security and context-awareness.

These are still early days in mobile cloud computing, with recent workshops in the area such as MobiSys,[22] MCCTA,[23] CMCVR,[24] and MCNCS.[25] As we pointed out, mobile cloud computing has overlapped with other areas such mobile peer-to-peer computing, application partitioning, and context-aware computing, but yet has its own unique set of challenges. There are numerous new mobile applications that a mobile cloud framework can enable, when many more resources can be made available to the mobile device (via the mobile cloud facility). The future could also explore the potential of local mobile clouds formed from collections of computers in ubiquitous devices in shoes, clothing, watches, jewelry, furniture and other everyday objects, as indeed such embedded computers will become more powerful. And so, the infrastructure, platform or application available as services will be of new forms: the infrastructure could be a powerful massively distributed set of cameras on stationary and mobile devices formed ad hoc and metered to cover an event, or a collection of distributed computers formed to compute a job seamlessly from the user's mobile device while the user is shopping. A car can sell its computational resources and pay for its own parking, or the collection of computers on crowds of people in a busy area forms an "elastic" collective resource for ad hoc use. There is also

[22] http://www.sigmobile.org/mobisys/2011/workshops.html.
[23] http://web.ftrai.org/mccta2011/.
[24] http://www.cmcvr.org/CMCVR11.html.
[25] http://dblab.csie.thu.edu.tw/MCNCS_2011/.

potential to have context sources or sensors (and sensor networks) in the vicinity of a mobile user sold as services to the mobile user, to support context-aware applications. However, challenges are present in order to "elastically" on-demand form clouds of services and resources efficiently, seamlessly and in a robust manner.

References

[1] S. Perez, Mobile cloud computing: $9.5 billion by 2014, http://exoplanet.eu/catalog.php, 2010.
[2] M. Satyanarayanan, Fundamental challenges in mobile computing, in: Proceedings of the Fifteenth Annual ACM Symposium on Principles of Distributed Computing, PODC'96, ACM, New York, NY, USA, 1996, pp. 1–7.
[3] L. Siegele, Let it rise: a special report on corporate it, http://www.economist.com/node/12411882, 2008.
[4] M. Satyanarayanan, Mobile computing, Computer 26 (1993) 81–82.
[5] W. Vogels, A head in the clouds the power of infrastructure as a service, in: Proceedings of the 1st Workshop on Cloud Computing and Applications, CCA'08.
[6] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, Above the clouds: a Berkeley view of cloud computing, Technical Report UCB/EECS-2009-28, 2009.
[7] J. Carolan, S. Gaede, J. Baty, G. Brunette, A. Licht, J. Remmell, L. Tucker, J. Weise, Introduction to cloud computing architecture—white paper, 2009.
[8] R. Buyya, C.S. Yeo, S. Venugopal, J. Broberg, I. Brandic, Cloud computing and emerging it platforms: vision, hype, and reality for delivering computing as the 5th utility, Future Generation Computer Systems 25 (2009) 599–616.
[9] Q. Zhang, L. Cheng, R. Boutaba, Cloud computing: state-of-the-art and research challenges, Journal of Internet Services and Applications 1 (2010) 7–18. http://dx.doi.org/10.1007/s13174-010-0007-6.
[10] L. Mei, W. Chan, T. Tse, A tale of clouds: paradigm comparisons and some thoughts on research issues, in: Proceedings of the Asia-Pacific Services Computing Conference, APSCC'08, IEEE, 2008, pp. 464–469.
[11] J. Cheng, R.K. Balan, M. Satyanarayanan, Exploiting rich mobile environments, Technical Report, 2005.
[12] 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, ACM, New York, NY, USA, 2010, pp. 6:1–6:5.
[13] R.E. Frederking, R.D. Brown, The pangloss-lite machine translation system, in: Proceedings of the Second Conference of the Association for Machine Translation in the Americas, pp. 268–272.
[14] E.E. Marinelli, Hyrax: cloud computing on mobile devices using MapReduce, Masters Thesis, Carnegie Mellon University, 2009.
[15] M. Satyanarayanan, Mobile computing: the next decade, in: Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & #38; Services: Social Networks and Beyond, MCS'10, ACM, New York, NY, USA, 2010, pp. 5:1–5:6.
[16] N. Vallina-Rodriguez, J. Crowcroft, Erdos: achieving energy savings in mobile OS, in: Proceedings of the Sixth International Workshop on MobiArch, MobiArch'11, ACM, New York, NY, USA, 2011, pp. 37–42.
[17] O. Amft, P. Lukowicz, From backpacks to smartphones: past, present, and future of wearable computers, IEEE Pervasive Computing 8 (2009) 8–13.
[18] X. Luo, From augmented reality to augmented computing: a look at cloud-mobile convergence, in: International Symposium on Ubiquitous Virtual Reality, 2009, ISUVR'09, IEEE, 2009, pp. 29–32.
[19] S. Pandey, W. Voorsluys, S. Niu, A. Khandoker, R. Buyya, An autonomic cloud environment for hosting ecg data analysis services, Future Generation Computer Systems 28 (2012) 147–154.
[20] H.-Y. Kung, C.-H. Chen, H.-H. Ku, Designing intelligent disaster prediction models and systems for debris-flow disasters in Taiwan, Expert Systems with Applications 39 (2012) 5838–5856.
[21] N. Aschenbruck, E. Gerhards-Padilla, M. Gerharz, M. Frank, P. Martini, Modelling mobility in disaster area scenarios, in: Proceedings of the 10th ACM Symposium on Modeling, Analysis, and Simulation of Wireless and Mobile Systems, MSWiM'07, ACM, New York, NY, USA, 2007, pp. 4–12.
[22] Y. Sasaki, Y. Shibata, A disaster information sharing method by the mobile servers in challenged networks, in: Advanced Information Networking and Applications Workshops, WAINA, 2012 26th International Conference on, pp. 1048–1053.
[23] M. Satyanarayanan, P. Bahl, R. Caceres, N. Davies, The case for VM-based cloudlets in mobile computing, IEEE Pervasive Computing 8 (2009) 14–23.
[24] 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, ACM, New York, NY, USA, 2010, pp. 49–62.
[25] A. Coronato, G.D. Pietro, Mipeg: a middleware infrastructure for pervasive grids, Future Generation Computer Systems 24 (2008) 17–29.
[26] S. Zachariadis, C. Mascolo, W. Emmerich, Satin: a component model for mobile self organisation, in: R. Meersman, Z. Tari (Eds.), On the Move to Meaningful Internet Systems 2004: CoopIS, DOA, and ODBASE, in: Lecture Notes in Computer Science, vol. 3291, Springer, Berlin, Heidelberg, 2004, pp. 1303–1321. http://dx.doi.org/10.1007/978-3-540-30469-2_31.
[27] J. Flinn, S. Park, M. Satyanarayanan, Balancing performance, energy, and quality in pervasive computing, in: Proceedings of the 22nd International Conference on Distributed Computing Systems, 2002, IEEE, 2002, pp. 217–226.
[28] R. Balan, M. Satyanarayanan, S. Park, T. Okoshi, Tactics-based remote execution for mobile computing, in: Proceedings of the 1st International Conference on Mobile Systems, Applications and Services, ACM, 2003, pp. 273–286.
[29] J. Dean, S. Ghemawat, MapReduce: simplified data processing on large clusters, Communications of the ACM 51 (2008) 107–113.
[30] R. Kemp, N. Palmer, T. Kielmann, H. Bal, Cuckoo: a computation offloading framework for smartphones, in: Proceedings of The Second International Conference on Mobile Computing, Applications, and Services, MobiCASE'10.
[31] R. Van Nieuwpoort, J. Maassen, G. Wrzesińska, R. Hofman, C. Jacobs, T. Kielmann, H. Bal, Ibis: a flexible and efficient java based grid programming environment, Concurrency and Computation: Practice and Experience 17 (2005) 1079–1107.
[32] D.C. Doolan, S. Tabirca, L.T. Yang, Mmpi a message passing interface for the mobile environment, in: Proceedings of the 6th International Conference on Advances in Mobile Computing and Multimedia, MoMM'08, ACM, New York, NY, USA, 2008, pp. 317–321.
[33] BlueCove.org. http://www.bluecove.org/, 2008 (accessed: 17.05.2012).
[34] L. Deboosere, P. Simoens, J.D. Wachter, B. Vankeirsbilck, F.D. Turck, B. Dhoedt, P. Demeester, Grid design for mobile thin client computing, Future Generation Computer Systems 27 (2011) 681–693.
[35] C. Clark, K. Fraser, S. Hand, J. Hansen, E. Jul, C. Limpach, I. Pratt, A. Warfield, Live migration of virtual machines, in: Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation—Volume 2, USENIX Association, 2005, pp. 273–286.
[36] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, A. Patti, Clonecloud: elastic execution between mobile device and cloud, in: Proceedings of the Sixth Conference on Computer Systems, EuroSys'11, ACM, New York, NY, USA, 2011, pp. 301–314.
[37] D. Huang, X. Zhang, M. Kang, J. Luo, Mobicloud: building secure cloud framework for mobile computing and communication, in: Proceedings of the Fifth IEEE International Symposium on Service Oriented System Engineering, SOSE, pp. 27–34.
[38] J. Lockwood, N. McKeown, G. Watson, G. Gibb, P. Hartke, J. Naous, R. Raghuraman, J. Luo, NetFPGA—An open platform for Gigabit-rate network switching and routing, in: Proceedings of the IEEE International Conference on Microelectronic Systems Education, MSE'07, pp. 160–161.
[39] M. Kristensen, Scavenger: transparent development of efficient cyber foraging applications, in: Proceedings of the IEEE International Conference on Pervasive Computing and Communications, PerCom.
[40] D. Borthakur, The hadoop distributed file system: architecture and design, http://hadoop.apache.org/common/docs/r0.18.0/hdfs_design.pdf, 2007.
[41] E. Walker, W. Brisken, J. Romney, To lease or not to lease from storage clouds, Computer 43 (2010) 44–50.
[42] L. Xinhui, L. Ying, L. Tiancheng, Q. Jie, W. Fengchun, The method and tool of cost analysis for cloud computing, in: Proceedings of IEEE International Conference on Cloud Computing, CLOUD'09, pp. 93–100.
[43] D. Narayanan, J. Flinn, M. Satyanarayanan, Using history to improve mobile application adaptation, in: Proceedings of Third IEEE Workshop on Mobile Computing Systems and Applications.
[44] B.D. Noble, M. Satyanarayanan, D. Narayanan, J.E. Tilton, J. Flinn, K.R. Walker, Agile application-aware adaptation for mobility, in: Proceedings of the Sixteenth ACM symposium on Operating Systems Principles, SOSP'97, ACM, New York, NY, USA, 1997, pp. 276–287.
[45] A. Kansal, F. Zhao, Fine-grained energy profiling for power-aware application design, SIGMETRICS Performance Evaluation Review 36 (2008) 26–31.
[46] X. Zhang, A. Kunjithapatham, S. Jeong, S. Gibbs, Towards an elastic application model for augmenting the computing capabilities of mobile devices with cloud computing, Mobile Networks and Applications 16 (2011) 270–284. http://dx.doi.org/10.1007/s11036-011-0305-7.
[47] K. Kumar, Y.-H. Lu, Cloud computing for mobile users: can offloading computation save energy? Computer 43 (2010) 51–56.
[48] C. Wang, Z. Li, Parametric analysis for adaptive computation offloading, SIGPLAN Notices 39 (2004) 119–130.
[49] H. Liang, D. Huang, D. Peng, On economic mobile cloud computing model, in: Proceedings of the International Workshop on Mobile Computing and Clouds, MobiCloud in Conjunction with MobiCASE.
[50] M. Puterman, Markov Decision Processes: Discrete Stochastic Dynamic Programming, John Wiley & Sons, Inc., 1994.
[51] I. Akyildiz, J. McNair, J. Ho, H. Uzunalioglu, W. Wang, Mobility management in next-generation wireless systems, Proceedings of the IEEE 87 (1999) 1347–1384.
[52] I. Constandache, X. Bao, M. Azizyan, R.R. Choudhury, Did you see bob?: human localization using mobile phones, in: Proceedings of the Sixteenth Annual International Conference on Mobile Computing and Networking, MobiCom'10, ACM, New York, NY, USA, 2010, pp. 149–160.
[53] N. Banerjee, S. Agarwal, P. Bahl, R. Chandra, A. Wolman, M. Corner, Virtual compass: relative positioning to sense mobile social interactions, in: Proceedings of the 8th International Conference on Pervasive Computing, Pervasive'10, Springer-Verlag, Berlin, Heidelberg, 2010, pp. 1–21.

[54] R. Mayrhofer, C. Holzmann, R. Koprivec, Friends radar: towards a private p2p location sharing platform, in: Proceedings of the 13th international conference on Computer Aided Systems Theory—Volume Part II, EUROCAST'11, Springer-Verlag, Berlin, Heidelberg, 2012, pp. 527–535.
[55] J. Krumm, K. Hinckley, The nearme wireless proximity server, in: Proceedings of Ubicomp: Ubiquitous Computing, Springer, 2004, pp. 283–300.
[56] C. Peng, G. Shen, Y. Zhang, Y. Li, K. Tan, Beepbeep: a high accuracy acoustic ranging system using cots mobile devices, in: Proceedings of the 5th International Conference on Embedded Networked Sensor Systems, SenSys'07, ACM, New York, NY, USA, 2007, pp. 1–14.
[57] M. Minami, Y. Fukuju, K. Hirasawa, S. Yokoyama, M. Mizumachi, H. Morikawa, T. Aoyama, Dolphin: a practical approach for implementing a fully distributed indoor ultrasonic positioning system, in: N. Davies, E. Mynatt, I. Siio (Eds.), UbiComp 2004: Ubiquitous Computing, in: Lecture Notes in Computer Science, vol. 3205, Springer, Berlin, Heidelberg, 2004, pp. 347–365. http://dx.doi.org/10.1007/978-3-540-30119-6_21.
[58] J. Park, H. Yu, Resource allocation techniques based on availability and movement reliability for mobile cloud computing, in: Proceedings of the 8th International Conference on Distributed Computing and Internet Technology, ICDCIT'12, Springer-Verlag, Berlin, Heidelberg, 2012, pp. 263–264.
[59] V. Sacramento, M. Endler, H.K. Rubinsztejn, L.S. Lima, K. Goncalves, F.N. Nascimento, G.A. Bueno, MoCA: a middleware for developing collaborative applications for mobile users, IEEE Distributed Systems Online 5 (2004).
[60] P. Bahl, V. Padmanabhan, RADAR: an in-building RF-based user location and tracking system, in: Proceedings of the Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM, vol. 2, pp. 775–784.
[61] I. Satoh, Dynamic deployment of pervasive services, in: Proceedings of the International Conference on Pervasive Services, ICPS, pp. 302–311.
[62] F. Samimi, P. McKinley, S. Sadjadi, Mobile service clouds: a self-managing infrastructure for autonomic mobile computing services, in: A. Keller, J.-P. Martin-Flatin (Eds.), Self-Managed Networks, Systems, and Services, in: Lecture Notes in Computer Science, vol. 3996, Springer, Berlin, Heidelberg, 2006, pp. 130–141.
[63] P. McKinley, F. Samimi, J. Shapiro, C. Tang, Service clouds: a distributed infrastructure for constructing autonomic communication services, in: Proceedings of the 2nd IEEE International Symposium on Dependable, Autonomic and Secure Computing, pp. 341 –348.
[64] P. Ypodimatopoulos, A. Lippman, 'Follow me': a web-based, location-sharing architecture for large, indoor environments, in: Proceedings of the 19th International Conference on World Wide Web, WWW'10, ACM, New York, NY, USA, 2010, pp. 1375–1378.
[65] A. Madhavapeddy, A. Tse, A study of bluetooth propagation using accurate indoor location mapping, in: M. Beigl, S. Intille, J. Rekimoto, H. Tokuda (Eds.), UbiComp 2005: Ubiquitous Computing, in: Lecture Notes in Computer Science, vol. 3660, Springer, Berlin, Heidelberg, 2005, pp. 105–122.
[66] W. Lehr, L.W. McKnight, Wireless Internet access: 3G vs. WiFi? Telecommunications Policy 27 (2003) 351–370. Compeitition in Wireless: Spectrum, Service and Technology Wars.
[67] J.-j. Dong, H.-c. Xu, A distributed online test system based on bluetooth technology, in: Proceedings of the Second World Congress on Software Engineering, WCSE, vol. 1, pp. 15–17.
[68] S. Cherry, Update: WiFi takes on bluetooth, IEEE Spectrum 45 (2008) 14.
[69] F. Wang, W. Dong, Y. Ji, A new credit based micropayment scheme, in: Proceedings of the IEEE International Conference on e-Business Engineering, 2008, ICEBE'08, pp. 596–601.
[70] M. Hosseinkhani, E. Tarameshloo, M. Shajari, AMVPayword: secure and efficient anonymous payword-based micropayment scheme, in: Proceedings of International Conference on Computational Intelligence and Security, CIS'10, IEEE, 2010, pp. 551–555.
[71] M. Tanase, V. Cristea, Quality of service in large scale mobile distributed systems based on opportunistic networks, in: Proceedings of The 7th International Symposium on Web and Mobile Information Services, AINA '11, IEEE, Singapore, 2011.
[72] L. Ramaswamy, L. Liu, Free riding: a new challenge to peer-to-peer file sharing systems, in: Proceedings of the 36th Annual Hawaii International Conference on System Sciences, IEEE, 2003, p. 10.
[73] S. Subramanya, B. Yi, User interfaces for mobile content, Computer 39 (2006) 85–87.
[74] J.-H. Ye, J. Herbert, Interface tailoring for mobile computing devices, in: C. Stary, C. Stephanidis (Eds.), User-Centered Interaction Paradigms for Universal Access in the Information Society, in: Lecture Notes in Computer Science, vol. 3196, Springer, Berlin, Heidelberg, 2004, pp. 175–182. http://dx.doi.org/10.1007/978-3-540-30111-0_15.
[75] J. Landay, T. Kaufmann, User interface issues in mobile computing, in: Proceedings of the Fourth Workshop on Workstation Operating Systems, IEEE, 1993, pp. 40–47.
[76] K. Leichtenstern, E. Andre, User-centred development of mobile interfaces to a pervasive computing environment, in: Proceedings of the First International Conference on Advances in Computer–Human Interaction, pp. 114–119.
[77] N. Palmer, R. Kemp, T. Kielmann, H. Bal, Ibis for mobility: solving challenges of mobile computing using grid techniques, in: Proceedings of the 10th workshop on Mobile Computing Systems and Applications, HotMobile'09, ACM, New York, NY, USA, 2009, pp. 17:1–17:6.
[78] K. Elbashir, R. Deters, Transparent caching for nomadic WS clients, in: Proceedings of the IEEE International Conference on Web Services, ICWS'05, vol. 1, pp. 177–184.
[79] X. Liu, R. Deters, An efficient dual caching strategy for web service-enabled PDAs, in: Proceedings of the 2007 ACM symposium on Applied computing, SAC'07, ACM, New York, NY, USA, 2007, pp. 788–794.
[80] O. Kharif, Perils of the mobile cloud. URL: http://www.businessweek.com/technology/content/oct2009/tc20091019_328787.htm, 2009.
[81] M. Shiels, Phone sales hit by sidekick loss. URL: http://news.bbc.co.uk/2/hi/technology/8303952.stm, 2009.
[82] B. Moyers, J. Dunning, R. Marchany, J. Tront, Effects of Wi-Fi and bluetooth battery exhaustion attacks on mobile devices, in: Proceedings of the 43rd Hawaii International Conference on System Sciences, HICSS, IEEE, 2010, pp. 1–9.
[83] J. Oberheide, F. Jahanian, When mobile is harder than fixed (and vice versa): demystifying security challenges in mobile environments, in: Proceedings of the Eleventh Workshop on Mobile Computing Systems & Applications, HotMobile'10, ACM, New York, NY, USA, 2010, pp. 43–48.
[84] J. Brodkin, Gartner: Seven cloud-computing security risks. URL: http://www.infoworld.com/d/security-central/gartner-seven-cloud-computing-security-risks-853, 2008.
[85] X. Zhang, J. Schiffman, S. Gibbs, A. Kunjithapatham, S. Jeong, Securing elastic applications on mobile devices for cloud computing, in: Proceedings of the 2009 ACM Workshop on Cloud Computing Security, CCSW'09, ACM, New York, NY, USA, 2009, pp. 127–134.
[86] J. Mulroy, Carrier iq rootkit reportedly logs everything on millions of phones [updated]. URL: http://www.pcworld.com/article/245229/carrier_iq_rootkit_reportedly_logs_everything_on_millions_of_phones_updated.html, 2011 (accessed: 19.05.2012).
[87] M. Fahrmair, W. Sitou, B. Spanfelner, Security and privacy rights management for mobile and ubiquitous computing, in: Workshop on UbiComp Privacy, pp. 97–08.
[88] K.-W. Park, S.S. Lim, K.H. Park, Computationally efficient pki-based single sign-on protocol, PKASSO for mobile devices, IEEE Transactions on Computers 57 (2008) 821–834.
[89] J. Han, Y. Zhu, Y. Liu, J. Cai, L. Hu, Provide privacy for mobile p2p systems, in: 25th IEEE International Conference on Distributed Computing Systems Workshops, 2005, pp. 829–834.
[90] J. Han, Y. Liu, Rumor riding: anonymizing unstructured peer-to-peer systems, in: Proceedings of the 2006 14th IEEE International Conference on Network Protocols, 2006, ICNP'06, pp. 22–31.
[91] G. Ghinita, P. Kalnis, S. Skiadopoulos, Prive: anonymous location-based queries in distributed mobile systems, in: Proceedings of the 16th International Conference on World Wide Web, WWW'07, ACM, New York, NY, USA, 2007, pp. 371–380.
[92] J. Han, Y. Liu, Mutual anonymity for mobile p2p systems, IEEE Transactions on Parallel and Distributed Systems 19 (2008) 1009–1019.
[93] A. Bandara, B. Nuseibeh, B. Price, Y. Rogers, N. Dulay, E. Lupu, A. Russo, M. Sloman, A. Joinson, Privacy rights management for mobile applications, in: 4th Int. Symposium on Usable Privacy and Security. Pittsburgh.
[94] B. Schilit, N. Adams, R. Want, Context-aware computing applications, in: Proceedings of the Workshop on Mobile Computing Systems and Applications, IEEE, 1994, pp. 85–90.
[95] H.J. La, S.D. Kim, A conceptual framework for provisioning context-aware mobile cloud services, in: Proceedings of the IEEE 3rd International Conference on Cloud Computing, CLOUD, pp. 466–473.
[96] P. Papakos, L. Capra, 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, ARM'10, ACM, New York, NY, USA, 2010, pp. 32–38.
[97] A. Klein, C. Mannweiler, J. Schneider, H.D. Schotten, Access schemes for mobile cloud computing, in: Proceedings of the Eleventh International Conference on Mobile Data Management, MDM, IEEE, 2010, pp. 387–392.
[98] C. Pils, I. Roussaki, T. Pfeifer, N. Liampotis, N. Kalatzis, Federation and sharing in the context marketplace, in: J. Hightower, B. Schiele, T. Strang (Eds.), Location- and Context-Awareness, in: Lecture Notes in Computer Science, vol. 4718, Springer, Berlin, Heidelberg, 2007, pp. 121–138. http://dx.doi.org/10.1007/978-3-540-75160-1_8.
[99] J. Lorch, A. Smith, Software strategies for portable computer energy management, IEEE Personal Communications 5 (1998) 60–73.
[100] K. Mansley, A.R. Beresford, D. Scott, The carrot approach: encouraging use of location systems, in: Proceedings of UbiComp, Springer, 2004.
[101] S. Chiyoung, S. Malek, N. Medvidovic, Estimating the energy consumption in pervasive Java-based systems, in: Proceedings of the Sixth Annual IEEE International Conference on Pervasive Computing and Communications, PerCom., pp. 243–247.
[102] J. Flinn, M. Satyanarayanan, Powerscope: a tool for profiling the energy usage of mobile applications, in: Mobile Computing Systems and Applications, 1999, Proceedings, WMCSA'99, Second IEEE Workshop on, pp. 2–10.
[103] A. Rice, S. Hay, Decomposing power measurements for mobile devices, in: IEEE International Conference on Pervasive Computing and Communications, PerCom, 2010, pp. 70–78.
[104] K. Banerjee, E. Agu, Powerspy: fine-grained software energy profiling for mobile devices, in: International Conference on Wireless Networks, Communications and Mobile Computing, 2005, vol. 2, pp. 1136–1141.

[105] J.-C. Cano, J.-M. Cano, E. González, C. Calafate, P. Manzoni, Evaluation of the energetic impact of bluetooth low-power modes for ubiquitous computing applications, in: Proceedings of the 3rd ACM International Workshop on Performance Evaluation of Wireless Ad Hoc, Sensor and Ubiquitous Networks, PE-WASUN'06, ACM, New York, NY, USA, 2006, pp. 1–8.
[106] J.C. Cano, J.M. Cano, C. Calafate, E. Gonzalez, P. Manzoni, Evaluation of the trade-off between power consumption and performance in bluetooth based systems, in: Proceedings of the International Conference on Sensor Technologies and Applications, SensorComm., pp. 313–318.
[107] C. Seo, S. Malek, N. Medvidovic, An energy consumption framework for distributed Java-based systems, in: Proceedings of the Twenty-Second IEEE/ACM International Conference on Automated Software Engineering, ASE'07, ACM, New York, NY, USA, 2007, pp. 421–424.
[108] H.T. Dinh, C. Lee, D. Niyato, P. Wang, A survey of mobile cloud computing: architecture, applications, and approaches, Wireless Communications and Mobile Computing (2011) n/a–n/a.
[109] Y.J. Nam, Y.K. Park, J.T. Lee, F. Ishengoma, Cost-aware virtual usb drive: providing cost-effective block I/O management commercial cloud storage for mobile devices, in: IEEE International Conference on Computational Science and Engineering, 2010, pp. 427–432.
[110] E. Koukoumidis, D. Lymberopoulos, K. Strauss, J. Liu, D. Burger, Pocket cloudlets, SIGARCH Computer Architecture News 39 (2011) 171–184.
[111] F. Capobianco, Five reasons to care about mobile cloud computing, http://www.ifosslr.org/ifosslr/article/viewArticle/24/47, 2009 (accessed: 17.05.2012).
[112] P. Ganapati, Apple blocks palm pre itunes syncing again, http://www.wired.com/gadgetlab/2009/10/palm-pre-itunes/, 2009 (accessed: 17.05.2012).
[113] Z. Zhang, X. Zhang, Realization of open cloud computing federation based on mobile agent, in: IEEE International Conference on Intelligent Computing and Intelligent Systems, ICIS 2009, vol. 3, pp. 642–646.
[114] W. Li, H. Yang, P. He, The research and application of embedded mobile database, in: International Conference on Information Technology and Computer Science, ITCS 2009, vol. 2, 2009, pp. 597–602.
[115] W. Ng, B. Ooi, K.-L. Tan, A. Zhou, Peerdb: a p2p-based system for distributed data sharing, in: Proceedings. 19th International Conference on, Data Engineering, 2003, pp. 633–644.

Niroshinie Fernando received her B.Sc. degree in Computer Science from University of Colombo, Sri Lanka, in 2007. She is a Ph.D. student affiliated with the Department of Computer Science and Computer Engineering of La Trobe University, Australia. Her research is funded by a Ph.D. grant from La Trobe University. Her main research interests are mobile and cloud computing.

Seng W. Loke is a Reader and Associate Professor at the Department of Computer Science and Computer Engineering in La Trobe University. He leads the Pervasive Computing Interest Group at La Trobe. He has (co-)authored more than 220 research publications including numerous works on context-aware computing, and mobile and pervasive computing. He has been on the program committee of numerous conferences/workshops in the area, including Pervasive 2008. He completed his Ph.D. at the University of Melbourne.

Wenny Rahayu is an Associate Professor at the Department of Computer Science and Computer Engineering, La Trobe University, Australia. Her research areas cover a wide range of advanced databases topics including Spatial and Temporal Databases, XML Databases, Data Warehousing, and Semantic Web and Ontology. To date, she has supervised to completion 10 Ph.D. graduates, and is currently leading a number of collaborative research and industry sponsored projects in the above areas.