2013 IEEE Seventh International Symposium on Service-Oriented System Engineering

Energy Efficient Seamless Service Provisioning in Mobile Cloud Computing

Anuradha Ravi
Sateesh K. Peddoju
Dept. of Electronics and Computer Engineering
IIT Roorkee
Roorkee, India
anuradha24ravi@gmail.com, sateesh@ieee.org

***Abstract*— Accessing Cloud via mobile device proves to be costly because of issues with wireless network. Due to communication overhead, offloading of application execution to Cloud consumes more energy than executing in the device itself. This paper proposes a novel framework in which application execution is offloaded to both Cloud and mobile ad hoc Cloud, in order to reduce this communication overhead. Distributed/Parallel execution of tasks is done to reduce the waiting time of mobile device, provided the cost of offloading is less, compared to cost of executing the application in device. Seamless service provisioning is achieved in this framework, by measuring the signal strength of the wireless medium. Once the signal strength threshold is reached, interim results are got from the device to which task is offloaded. This paper proposes the framework for energy efficient seamless service with features like, connecting heterogeneous mobile devices to form mobile ad hoc Cloud, service discovery in mobile ad hoc Cloud and offloading decisions.**

Keywords-Mobile cloud computing, mobile ad hoc cloud, seamless service, energy efficiency, offloading, MANETS.

I. INTRODUCTION

Mobile Cloud Computing is a booming area of research, because of increase in number of customers preferring to use mobile devices than desktop computers. To add it all, Smartphones have transformed the way we once looked at mobile phones. Now, mobile phones are not just phones meant for calling, sending/receiving text messages, but are also used for running various applications. Mobile phones face major challenges like energy consumption, running applications, which require heavy CPU cycles, memory capacity and low bandwidth to access the internet. To overcome these issues, Cloud is just a boon. The various services provided by Cloud can be utilized to run applications and store data pertaining to mobile devices. According to [1], Mobile Cloud Computing is a tripod with a combination of three aspects – the mobile device, Cloud and the network connectivity. Thus, leveraging the services of Cloud for mobile devices is termed as "Mobile Cloud Computing" (MCC).

According to a research [2], over 240 million business customers would be leveraging services of cloud computing through mobile devices by 2015, generating revenues of $5.2 billion. To keep up with this pace, mobile phone software developing companies like Google and Microsoft have been competing in the field of research, for providing the best service to their customers. Just as every new technology brings along challenges, MCC has issues to deal with. Issues like, availability of network to access Cloud, vendor lock-in by mobile device manufacturers and mobility are just to name a few. Since a user expects that the battery of the device lasts for a longer period, energy efficiency of a mobile device is very important. While Cloud can be used to leverage the application execution of the devices, the communication between Cloud and the device drains out the energy level in the device. Other issues include limited bandwidth, security and heterogeneous access schemes in mobile environment.

MCC can also be thought of as leveraging the execution to peer mobile devices forming a Cloud amongst the mobile devices in ad hoc manner. This again gives raise to various ad hoc related issues like mobility of devices, distributed and parallel computing amongst heterogeneous mobile devices, distributed shared memory access for devices with very low memory capacity, and low bandwidth to transfer execution data. This methodology comes in handy when leveraging execution or content to Cloud via wireless medium proves to be costly or when there is no network access to the Cloud.

To combat the above mentioned issues, various researchers have come up with their ideas. In the second section of this paper, we analyse the related work done and list down the possible areas for research. A mobile device user might want to offload the execution, in case the offloading is beneficial in terms of conserving energy, or access data from the Cloud storage without being want to wait for high network access. So, in the third section, we present our proposed framework to provide seamless service for mobile devices. Considering that the user might want the battery to last longer, we also propose to make this framework energy efficient.

II. LITERATURE REVIEW

Researchers have been on the job to tackle the issues of MCC. Since this paper aims to address the issue of energy efficiency and seamless service provisioning, we focus on the research done so far to improve energy efficiency, by offloading the content to the Cloud/mobile ad hoc Cloud and

978-0-7695-4944-6/12 $26.00 © 2012 IEEE
DOI 10.1109/SOSE.2013.67

provide seamless service for the mobile device and analyse energy efficient offloading methods and seamless servicing from various corners.

A. Based on Energy Efficiency

Robinson [3] states that energy stored in the battery is limited and is growing only at a pace of 5% annually. Another study [4] indicates that various techniques like, adapting new semiconductor technology, executing program slowly, avoid wasting energy by reducing tail and ramp energy and migrating computation, can be adapted to reduce the energy consumption in mobile device.

Migrating computation can be done by executing the application for mobile devices on the Cloud, but the cost of communication between the mobile device and the Cloud needs to be considered. It can be concluded from the analysis of studies on the various wireless network media [6,8,11], that 3G consumes more energy than WiFi or Bluetooth, due to its bandwidth and range. According to Crowcroft et al [5], the energy efficiency of communication depends on data rate, traffic pattern and source/destination location. This study also proves that 3G+ technologies become more energy efficient to transmit large volumes of data compared to WiFi and Bluetooth. Miettinen et al [6] proves that bursty traffic is more beneficial in terms of energy efficiency than smooth traffic, due to the utilization of bandwidth. The final experimental results in this study define a thumb rule as follows: "*For computation offloading to be beneficial, the workload needs to perform more than 1000 cycles of computation for each byte of data*". Karthik and Yung [7] propose a method to decide whether to offload the processing to Cloud or make it run in the mobile device. Their method suggests that, "as the computation increases and there is comparatively small amount of interaction between the mobile device and the cloud, the application can be offloaded to the Cloud".

Schulman et al [8] predicts the future signal strength of a mobile device to download contents from the internet. This study focuses on applications that synchronize (like emails) or stream (like playing videos) and predicts when to download the contents. Prediction is done with the help of past history of signal strength (Received signal strength indication (RSSI) values). This technique fails, if the user changes the speed of moving device or uses some other path for travel. Microsoft has developed Stratus [9], for reducing the energy consumption in mobile phones, by optimizing the usage of wireless communication. So as to achieve this goal, this work performs aggregation and compression of data at the Cloud end and scheduling at the device end. Stratus follows aggregation method to send data packets to mobile devices, thus reducing the number of interactions between Cloud and the device. The number of bits transmitted over the wireless network is reduced by using compression technique. Aggregation and compression technique is deployed on a cloud based proxy server. Care has been taken to reduce the overhead of decompression at mobile device. Signal strength of a communication medium varies from time to time. So, Stratus follows scheduling methodology to schedule packets to Cloud only when the signal strength is high. Since stratus uses Bartendr [8] technique for scheduling, the drawback of this technique follows for stratus as well.

B. Based on Various Offloading Frameworks

Considering the wireless medium, it is difficult to decide, when and what to offload from mobile device, because the communication overhead can prove to be a costly affair compared to running application in the device. There are various offloading frameworks and methodologies suggested in the literature. This section analyses the various frameworks and also algorithms used for decision making.

Analysing the bandwidth of the wireless medium used in the experiment, the thesis work of Karthik [4], decides whether to perform execution in the local device, fully offload or partially offload the execution to server. This study considers the energy required for communication and the idle time spent by the device, for taking dynamic decisions to offload computation to server. Servers are ranked based on the data history, for reducing latency and energy consumption. In this work, computation is offloaded to the server which computes faster and takes less time for returning back the result. This reduces idle time and thus conserves energy when offloading is performed. The flaw with this system is that it considers only one application and does code modification to analyse the performance of server, which is not practically possible for all applications. An adaptive offloading algorithm has been suggested in literature by Koyachey et al [10], which defines three metrics for decision making. This study considers available memory, energy consumption and execution time to derive an algorithm for this decision making process. In study experiments the algorithm with N-queen problem and proves that, offloading is beneficial only when the required computation is large. Since the study considers only WiFi for experimental purpose, it's difficult to say if offloading would be beneficial when 3G is used.

Most of the frameworks proposed in the literature take into consideration - migration cost, computation cost of the device and characteristics of wireless medium to decide whether to offload the computation to Cloud. One among them is CloneCloud, proposed by Chun et al [11]. In this framework, a Clone of the mobile device exists in the Cloud, which is used for computation of high performance applications. CloneCloud does a static analysis by running the application once on the device, and once on the Cloud to find whether offloading would be beneficial or not. The drawback with this model is that it's not dynamic, in sense that profiling is done only once for all application and stored in a binary file which is used every time the application is run. Building upon CloneCloud, Hung et al [12] have realized the virtual execution of mobile devices on the Cloud. This work maintains a Clone of the device as well as

synchronizes the data/application states with the virtual environment. An agent program is run both on physical and virtual phone for synchronization purpose. QoS (Quality of Service) requirements like bandwidth, priority for applications are also met by guaranteeing the required QoS measures for all the applications/threads running on the virtual phone. This work fails to address the energy consumption of constant synchronization between physical and virtual phone. Since communication can drain the battery level of a mobile device, this framework might not be energy efficient.

Reflecting on the principle to reduce energy consumption and increase performance of an application, Cuervo et al [13] proposed MAUI. This framework offloads the code from mobile device to a nearby MAUI server. MAUI depends on the developers to mark a method of an application to be remotable. In order to decide whether the method can be offloaded, MAUI profiler and solver are used in the device. The experimental results considering various RTT's (Round Trip Time) suggest that, as RTT values increase, the performance decreases and energy consumption increases. So, if the server is near to the device, RTT is low, resulting in better performance. The results also prove that using 3G network increases the energy consumption of the device, though the methods are offloaded. The drawback of MAUI is that it requires programmers to mark the methods as remotable. So, if a programmer mistakenly marks the native code interaction with the device as remotable and the method is offloaded, it might consume more energy. SPECTRA proposed by Flinn et al [14], is a self-tuned application to decide if an application module is to be offloaded to server. Self-tuned because, it does not depend on the programmer or the application to specify any metrics. It monitors the application module to collect information under various environmental conditions which is maintained as a log. Though SPECTRA takes dynamic decisions, the cost of maintaining the log is huge and if the log size increases with time, it might even be difficult to search log and maximum execution time would be wasted in computing the logs.

Silva et al [15] proposed SPADE, which distributes the execution of tasks between mobile device and a nearby desktop system that runs a similar application to that of the mobile device. The framework of SPADE contains mobile device, which requires computation power and a cycle provider that provides computation power for mobile devices. SPADE achieves parallelism by distributing the tasks among various cycle providers. Eg: An image rendition can be done by different desktop systems and the result sent back to mobile device, which is then combined to give result for the user. The drawback with SPADE is that, it does not consider the mobility aspect of the device. Since it only considers a laptop and a desktop PC for experimental purpose, the mobility aspect is not covered by the scheduler. Component based offloading has been adapted by March et al [16] in the μCloud framework. Distributed execution is achieved by running components in different environments and thus collaboratively achieving the result. A component repository is maintained for developers to store their components which are used by the devices for execution. Many applications might require the same component to be run or one application might require many components to be run. This can be achieved by running components in different environments and thus collaboratively achieving the result. Application Builder generates an application graph which helps in orchestrating (combining the results of components) the application.

Android [17] framework runs two threads mainly service thread and UI (User Interface) thread. UI thread runs in the foreground and interacts with the mobile user and service thread runs in the background. This principle has been used by various researchers to offload execution of Android application. In one such work proposed by Chen et al [18], virtual smartphone is deployed on the Cloud in order to run all the applications, which a Smartphone device runs. The application backend execution, which is the service part of any Android application, is transmitted to the virtual smartphone on the Cloud. The front end execution or the activity part of the application is left on the device for interaction with the user. This framework adapts application partitioning and decision making dynamically, without intervention of the user/developer. This framework fails to address the communication overhead for offloading the tasks to virtual smartphone. Kemp et al [19] proposed an application development framework named Cuckoo, which integrated with Android framework, identifies the remote method and separates the remote method to be offloaded to Cloud. Developer is required to write the remote method using Cuckoo developing environment, which would be meant for Cloud to execute. This helps in exploiting parallelism and other HPC (high performance computing) properties to be achieved in Cloud for the application. Though Cuckoo offloads the remote method to Cloud for execution, it requires developer intervention and is not suitable for already existing application.

ThinkAir proposed by Kosta et al [20], optimizes mobile code offloading not only at mobile end, but also at the Cloud end. ThinkAir framework requires the programmers of an application to specify which modules can be offloaded and which should run on the device. On the mobile end, it maintains profile information about the device and application, for assisting execution controller in taking a decision to offload execution. Elastic property of Cloud is used to allocate virtual machines (VM), when there is a need for extra memory or computation power for an application to run. This also helps in executing applications in a distributed manner. The drawback of this framework is that it takes time in order of seconds to create or start a VM dynamically, and this factor is not considered while taking decision for offloading the mobile code.

So as to take decision on whether to offload computation to Cloud, the various offloading framework considers the computation cost in the mobile device and communication

cost between the device and Cloud. Studies in the literature [7] prove that offloading could be less beneficial when the communication cost is more. So, it's very important to bring a balance between the communication and computation by utilizing the bandwidth effectively in an energy efficient manner.

C. Based on Mobile Ad hoc Cloud

Researchers are working towards forming Ad hoc Clouds, to share job among the peers within the vicinity. This paper analyses the work done in the Mobile Ad hoc Cloud area.

Map-Reduce [21] programming framework is used in Cloud to distribute job amongst the nodes. The same is realized for mobile device by Marinelli [22]. This work focuses on file sharing and job execution on peer mobile devices. Building its methodology on Hadoop [23], a traditional server acts like the master node containing the NameNode and JobTracker, while the mobile devices act as the worker nodes which contain DataNode and TaskTracker components. Every application runs a HDFS and Hyrax in the background to interact with the peer devices. This study fails to address the question of energy efficiency, i.e: if this framework can really help in conserving the energy of a mobile device. The same MapReduce [21] technology is applied in Misco framework proposed by Dou et al [24]. The Map-Reduce is adapted for mobile systems, where various Misco workers (mobile devices) participate in the MapReduce operation. A mobile device submits jobs to server for execution, which is distributed amongst the workers. Misco server collects the job, partitions it based on the key value pair and assigns a unique identity for each task. These tasks are then sent to the Misco workers when they ask for work. The results from various workers are collected and sent back to the client by Misco server. Though Hadoop and MapReduce satisfy parallel execution of application, the single point failure of the central server is a matter of concern.

Kristensen et al [25] proposed Scavenger framework to realize mobile ad hoc Cloud. In this study, clients offload their computation to surrogates. Surrogates are the mobile devices in mobile ad hoc Cloud within the vicinity of the mobile device (client). Clients take decision to offload and send their request to surrogates for executing the code. It requires Scavenger libraries to execute the code which is installed by the client if one doesn't exist before. The experimental results show that the computation power of each device decides how efficiently a task can be performed in the ad hoc network. The model fails to address issues like fault tolerance.

In the research work proposed by Canepa and Lee [26], peers interact with each other for a common interest. Mobile device use this framework, for discovering a device with the common interest, to download content from Cloud. Thus, two or more devices can collectively get the service provided by the Cloud. In this framework, trust management between mobile devices has not been considered. Also, there is no motivation for mobile device to participate in the ad hoc Cloud and provide service for its peers. Caching can help to reduce the amount of interaction with Cloud and this technique has been used by Microsoft research in their work PocketCloudlets [27]. In this work, caching is done at the mobile end keeping in mind the frequently accessed search results by the user as well as community on the whole. This work fails for dynamically changing web contents, since caching wouldn't help for such contents. Sharing of contents between mobile devices has been a point of research for a long time now. In the work proposed by Raatikainen et al [28], a centralized CMS (content management system) server is maintained, which consist of details about the contents present in various mobile devices. Any device can access contents of any other device via the CMS server. Since this system is centralized, single point failure is a matter of concern.

Job sharing in mobile ad hoc Cloud is an important aspect. Hummel et al [29] proposed a job scheduling algorithm, considering the fairness and fault tolerance in the mobile ad hoc network. In this framework, the job queue is placed in a remote environment, which is accessible to all the mobile devices. This is done to give access for sharing common data amongst various jobs. Fault tolerance, monitoring and maintenance of the network are taken care of in this framework. Since all mobile devices access a centralized remote server for jobs, single point failure is a matter of concern. Fernando et al [30] proposed a dynamic model for forming Mobile Ad hoc Cloud and share jobs to reduce energy consumption. A device equally distributes jobs amongst its peers and devices in the network to collectively complete a task. Here, mobile devices in the environment use Bluetooth for forming the ad hoc network and share jobs initiated by a master device for executing a particular task. This work fails to address the energy efficiency of keeping the Bluetooth device always on, for this model to succeed. Also, it does not distribute the jobs according to the capabilities of the device.

Distributing jobs to peer devices can be a good option when there is no network connectivity with Cloud, but various issues like fault tolerance, low bandwidth and heterogeneity of mobile devices needs to be addressed.

D. Based on Seamless Connectivity

Various middleware frameworks have been proposed in the literature for seamless connectivity between the Cloud and mobile device. One among them is Cloudlets, proposed by Satyanarayanan et al [31]. Cloudlet is a middleware which takes requests from mobile device for executing applications on virtual machines and is also connected to Cloud with a high bandwidth. A mobile device can offload execution to this Cloudlet considering that there already exists a base VM in the Cloudlet. The overlay is sent to Cloudlet for applying it on base VM and for executing the application. Though this is one of the successful approaches in the literature, searching for a base VM Cloudlet is a difficult task. Qingfeng Liu [32]

proposed a solution combining mobile-agent technology with mobile cloud computing. It provides stability of service by the usage of mobile-agent. The architecture states that, every cell region in mobile environment is divided into cloud units, and these cloud units are interconnected with each other and also connected with remote cloud unit for execution of complex applications. Universal Mobile Service Cell (UMSC), a component of this architecture, abstracts the mobile agent to solve the influence of unstable network. This architecture is not suitable for interactive applications, since the service is migrated and if there is low bandwidth, the mobile agent has to be used again and again for transmitting the interactive data.

The nature of wireless medium doesn't support an uninterrupted service provisioning for mobile devices. Techniques like caching or Cloudlets can be a good choice, but optimization is still required in this area to provide seamless service.

III. PROPOSED WORK

A. Motivation

In the present day scenario, a mobile device user expects an uninterrupted service from the Cloud, but due to the characteristics of wireless network, it's difficult to achieve. Signal strength of the wireless network keeps varying depending on the factors like, channel conditions and how far the device is from the base station. This in turn regulates the time taken to upload/download data from the Cloud. Also, constant interaction with the Cloud and communication with outside world can reduce the energy level of the device. Hence, the framework proposed in this paper, provides a seamless service to the mobile user in an energy efficient manner.

The proposed framework avails service for a mobile device both from the Cloud infrastructure and mobile ad hoc Cloud. While performing computation offloading, computation intensive task which requires minimal interaction with the mobile device, and doesn't require data to be shared with other tasks, is offloaded to the Cloud. Keeping in view the nature of mobile device in the mobile ad hoc Cloud, decision is taken to offload tasks to devices in mobile ad hoc Cloud. A mobile ad hoc Cloud works mainly on the principle of MANETS (Mobile ad hoc network). Mobile ad hoc Cloud is used in this framework, to reduce the communication cost between the mobile device and the Cloud. It is assumed that energy consumption amongst devices in mobile ad hoc Cloud would be less compared to continuous interaction with the Cloud. Also, sharing of data common to various tasks of an application becomes easy in mobile ad hoc Cloud environment. It is also assumed that, sharing of data in mobile ad hoc Cloud is beneficial in terms of energy efficiency than using Cloud infrastructure. To the best of our knowledge, there is no work which takes into consideration both the Cloud infrastructure together with mobile ad hoc Cloud for offloading. This gives way to distributed/parallel execution in both the environments. In order to provide seamless service, transfer of jobs is done between the devices or from Cloud to device. To do so, it is important to consider the network connectivity with the devices in mobile ad hoc Cloud, as well as the Cloud infrastructure. The proposed framework also helps in tackling the mobility issue of the device and its peers. A detailed discussion on the framework, service provisioning and offloading decisions are presented in the following sections.

B. Framework

1) Architecture

Figure 1 shows the overall idea of the architecture in the proposed framework. In this framework, a mobile device takes service from the Cloud as well as the mobile ad hoc Cloud. When a client feels the need for availing a service, it immediately checks for availability of connection with the Cloud. If the connection is unavailable, it initiates the process of forming mobile ad hoc Cloud. This paper proposes domain based architecture for such environment. According to this architecture as shown in Figure 1, a mobile ad hoc Cloud consists of various domains. A domain consists of heterogeneous mobile devices with various computation capabilities. A mobile ad hoc Cloud is formed, with various mobile devices that are near to the requesting client mobile device. The client initiates the process of forming the mobile ad hoc Cloud. In turn the devices in this domain, works towards forming more domains by connecting with other devices, which are in their vicinity. Devices in the domain are inter-connected amongst each other for collectively executing a task. Every device in the mobile ad hoc Cloud is connected with the Cloud depending upon the availability of connection.

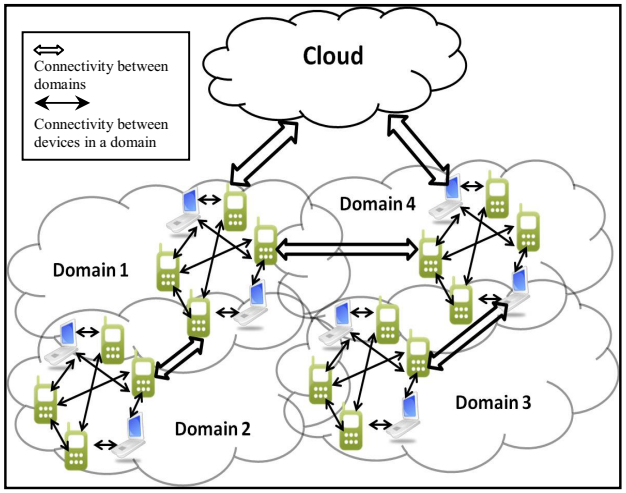

Figure 1. Mobile Cloud Computing Architecture

For example, a mobile device A might be connected with its peer B by means of one network connection. This mobile device B might be connected with another device C which is not in vicinity of A. The devices A and B form one domain

while B and C form another domain. Here, the device B is the connecting device between the two domains. When the device A requires service of C, the connecting device B can be used for communication between the client device and device providing the service. In this manner, many devices can be connected together to avail the service. The number of mobile devices that can form a domain depends on the capacity/limitation of the wireless network connecting the devices in a domain. The client device can know about devices in other domains with the help of service discovery module.

2) Functions

The various functions incorporated in this framework are, service discovery, seamless service and offloading as shown in Figure 2. The interaction between modules in client and peer devices, to provide service for user is also shown. With the help of service discovery module, a client device can discover various devices available in mobile ad hoc Cloud, which can provide various services. The service discovery module between the client and peer device is connected so as to find the various services offered by the peer devices.

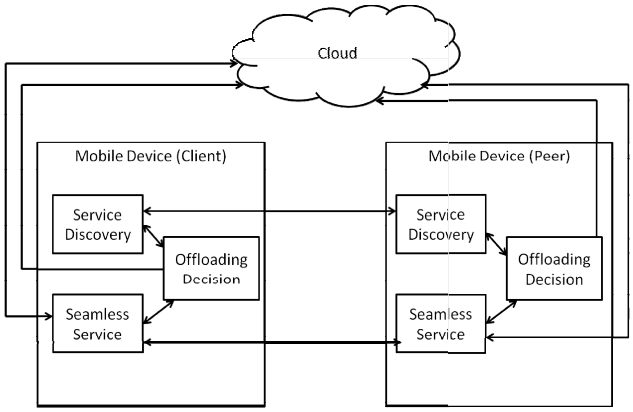

Figure 2. Interaction among modules in proposed framework.

The signal strength of the client device is measured to predict the probability of discontinuity of service from the device and find ways to continue service provisioning for the user. In this process, the client device may try to contact the nodes in its home domain by virtue of other ways such as Bluetooth and Wi-fi. The seamless service module of the client and peer devices interacts to take decision for continuing the service, based on the connectivity with the device. At the same time, keeping energy efficiency in mind, computations are offloaded with the help of both Cloud and mobile ad hoc Cloud. Based on the factors like, computation requirement of a task, available bandwidth, the choice of device/Cloud is made. The offloading decision depends on the various parameters like bandwidth, device and application characteristics. Hence, the offloading module interacts with seamless service and service discovery module to take appropriate decision. A detailed discussion on the various modules is discussed in the forth coming sections.

C. Service Provisioning

1) Service Discovery

Every device in a domain has various services to offer. Some of the services could be, executing applications for client device, providing access to Cloud and providing information (eg: weather forecast, stock rates) requested by the device. The services provided by a device are advertised using the discovery message. Through service discovery, a mobile device in domain 1 can find about services availability in domain 2. Figure 3 shows inter domain service provisioning for a mobile device.

The discovery message is advertised by a device once it enters the mobile ad hoc Cloud. Table 1 shows the service discovery message which is sent by a device when it enters the network. This discovery message is received by all the devices in the network and they update this table from time to time. The parameters in the table help in knowing the characteristics of a device in mobile ad hoc Cloud and the service it can provide. This helps the client in taking a decision to choose the device for getting service it requires.

TABLE I. SERVICE DISCOVERY TABLE

Unique ID	Service Offered	Computation Capability	Network Characteristic	Distance	Available Energy

The parameter “unique ID” in the table is used for identifying a particular device within the mobile ad hoc Cloud. “Service offered” parameter, is the service the device is ready to provide in the network. The parameter “Computational capability”, is the processing ability of the device like the type of processor, memory capacity etc. The parameter “Network characteristic”, is the medium through which it is connected to any other network other than its own along with the bandwidth details. The parameter “Distance” is the hop count to reach the device and when advertising, this value is initially zero. This value is updated by every device once the message is received. If the hop count received is zero, then the device updates the distance to one. This is done to know how many hops it would take to reach the device. It helps in finding the efficient device for requesting service. “Available energy” parameter is the measure of energy which can be used for providing service to other devices. This is used for assigning weights to devices while deciding whether to offload the contents.

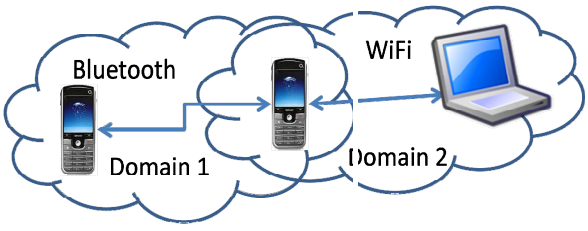

Figure 3. Domain to Domain service provisioning.

A similar approach to GAF (Geographic Adaptive Fidality) [33] routing protocol is used in order to route the service request. We feel that the approach followed in this protocol is suitable for our framework. However, any other routing protocol can be adapted based on the energy efficiency requirements. In GAF, the connection of a mobile device is turned off when it's not participating in any activity. A device can wake up after a time period to advertise its presence for participation. This helps in conserving energy of devices in the network. Also, in order to limit the discovery process, a similar approach used in ZRP (Zone routing protocol) [34] routing protocol is adapted in our framework. In ZRP, the network stops discovering for a node after the maximum hop count value is reached. This can be applied in the framework in order to stop discovery after a maximum hop count is reached for better performance of network. GAF routing protocol is used within a particular domain. If a device is not participating in any activity, it puts itself off to a sleep mode. This message is advertised during service discovery to the client device. When the device gets back from the sleep mode, the connecting device again advertises the service to client device. This way, energy is conserved for devices participating in mobile ad hoc Cloud. ZRP is used in the global scenario while connecting various domains. A mobile device stops looking for services once the maximum hop count value is reached. For example, if the maximum hop count is two, there are two domains between the client device and the device providing service. More number of connecting domains can lead to unnecessary delay, which can also reduce the energy efficiency. Hence, it's important to maintain the maximum hop count value for which ZRP protocol is used.

Since we have divided mobile ad hoc Cloud into different domains, the connecting device between the domains takes care of passing its service discovery table to the adjacent domain. This helps in knowing about the service availability across domains. The devices receiving this discovery table then updates their own table. Every device circulates its service discovery table amongst those devices to which it's connected to. Hence, when the device receives this table, it compares the table with its own and updates the information accordingly. This table can be referred by the device to take appropriate decision on selecting the device for availing the service.

2) Seamless Service

To provide seamless service, it is important to keep track of the signal strength with devices in mobile ad hoc Cloud, as well as availability of connection with the Cloud from time to time. In order to do so, RSSI (Received signal strength indication) value of the signal at the input of the receiver is measured. According to study in [8], a higher RSSI value always results in an energy efficient communication. The client device maintains a threshold for RSSI value and if the RSSI value starts dropping below the threshold, the interim job results are obtained from the device/Cloud and the execution is either continued in the device itself or is offloaded to some other device. If the device which has offloaded the execution itself moves out of the region, we propose that the interim results be obtained from all the devices in the mobile ad hoc Cloud and the execution continued in the device itself.

Since jobs are transferred between domains, the threshold value of RSSI should vary for inter domain and intra domain connections. This is because, it would take time to retrieve result from other domain and pass on the result to the client. The intermediate mobile devices which help a client receive service from other domain devices also execute the threshold policy in order to retrieve interim results from service providing devices. When the intermediate device senses a fall of signal strength from the threshold, it breaks the service and passes it to the nearby device with similar configuration. If such device is not found, the interim result is passed on to the client for further action. Any of the existing models like Okumura or Hata model [35] can be used to define this threshold value. Figure 4 gives an overall view of the threshold policy adapted in this framework. This threshold policy is executed on the client device which requests for service.

Further, elasticity property of the Cloud can be adapted in mobile ad hoc Cloud. Since a mobile device might not be able to provide service for some reason, it can transfer the service request to its peer for execution. Every mobile device has the decision making module to offload the service. So, without informing its client, the device can transfer request to its peer devices for providing service. This way the client device doesn't have to be interrupted from getting service.

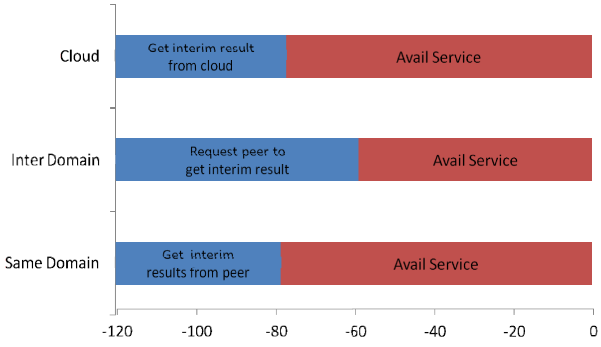

Figure 4. When to avail service

D. Offloading decisions

Code offloading is performed for executing applications in distributed/parallel manner. With the help of past history, the device can learn which module would take how much computation and time to run. If there is a new module without any past history, the device uses the past history of modules with similar characteristics to know the details. We calculate the total computation cost of an application to be the sum of cost of offloading data to the Cloud (C_{cloud}), cost of offloading modules to various peers (C_{peer}) as well as the calculating a part of the application in the mobile device itself (C_{mobile}). Since we can use different peers (n) to offload

different tasks, the cost of offloading to all the peers is summed up. In order to provide energy conservation of mobile device, this total computation cost of using peers, cloud and the mobile, must be less than the computation cost of all the modules if computed only in the mobile device. Equation (1) gives the total computation cost.

$$\text{Total Computation Cost} = C_{cloud} + \sum_{i=1}^{n}(C_{Peer_i}) + C_{mobile} \quad (1)$$

To find the idle peer for offloading, computation cost of running a module (C_i) in the device is considered, along with the channel capacity (D_i/B_i) to send data, where D is the amount of Data to be sent and B is the Bandwidth of the wireless network. For devices which are in different domain from that of client device, the channel capacity of the domains through which the request hops is summed up. The number of hops can be got from the service discovery table parameter "Distance" (d). We also give a weight for each device pertaining to the energy available with the device. The weight (w_i) also depends upon the signal strength (RSSI value) with the device. If the RSSI value is near to the threshold, we give less weight since the communication cost might increase. The cost of computing the module can be found by using the computation capabilities sent in the discovery message. Equation (2) is used for calculating the cost of offloading to a peer device.

$$C_{Peer_i} = \text{Min}(f(i)) \quad (2)$$

$$f(i) = \left(C_i + \sum_{i=i}^{d}(D_i/B_i)\right) * w_i \quad (3)$$

The same formula can be used to find the best peer even for selecting the device for receiving other services like INaaS, AaaS etc. For INaaS, we consider only the wireless medium characteristics making the computation cost zero. This is because there is no computation to be done by the device to get the result. In case of AaaS, all the factors are considered to take a decision.

E. Rewards

To motivate the users to participate in the mobile ad hoc Cloud computing, we propose that rewards be given to the users from Cloud provider. Considering that we have one Cloud provider providing service for all mobile devices participating in the network activity, we can provide the users with reward points which can be claimed by them for utilizing the service provided by Cloud.

F. Issues and Challenges

Considering the above framework, we have a set of challenges to be tackled:

1. *Trust Management* – Trust amongst the peers is very important when getting services. So, it's important to have a trust management for this framework to be secure from unwanted intruders.
2. *Load balancing in mobile ad hoc Cloud* – It is important to balance the load in the network, since no device should feel the burden of being assigned more work. Since any device can send tasks to any other device in the network, this load balancing needs to be distributed.
3. *Elasticity in mobile ad hoc Cloud* – If there is a need for more computation based on infrastructure or service, dynamic decisions needs to be taken.
4. *Heterogeneous connectivity among domains* – A device might be connected in one domain via Bluetooth while another via WiFi. So, care needs to be taken care of so that there is a full utilization of the bandwidth.

IV. CONCLUSION

This paper presents a novel framework for service provisioning, using Cloud together with mobile ad hoc Cloud. This framework helps in achieving energy efficient seamless service provisioning for mobile cloud computing. It helps in utilizing the computation power of heterogeneous mobile devices which are near to the client device. The service discovery in mobile ad hoc Cloud helps in sharing the load amongst all the devices, thus helping to conserve energy. The offloading decision algorithm helps in deciding as to which device to be used for taking service. This algorithm takes into consideration both the computation and communication cost with the peer devices and the Cloud. The tasks which require high computation power, while less interaction can be offloaded to the device for which communication cost is more. This helps in conserving energy of the devices. With the help of signal strength, decision is taken to continue or break the service. If it's decided to break the service with a particular device, interim results from the device is taken, in order to continue the execution in any other device. This helps to provide seamless service to the user. Rewards are provided to mobile devices for encouraging its participation in the mobile ad hoc Cloud.

In future, the work can be taken up in various aspects like, offloading and partitioning of mobile applications for improving energy efficiency of mobile devices, trust management, elasticity in the mobile ad hoc Cloud and load balancing in mobile ad hoc Cloud.

REFERENCES

[1] Z. Sanaei, S. Abolfazli, A. Gani, and R. H. Khokhar, "Tripod of requirements in horizontal heterogeneous mobile cloud computing", http://arxiv.org/ftp/arxiv/papers/1205/1205.3247.pdf, 2012.

[2] Mobile cloud computing forum - http://www.cloudcomputinglive.com/events/170-mobile-cloud-computing-forum.html

[3] S. Robinson, "Cellphone Energy Gap: Desperately Seeking Solutions",

http://www.strategyanalytics.com/default.aspx?mod=reportabstractviewer&a0=4645, Tech. rep., Strategy Analytics, 2009.

[4] Karthik Kumar, "Application-Based Energy Efficient Mobile and Server Computing", PhD Thesis, Purdue University, https://engineering.purdue.edu/HELPS/People/thesis/Kumar.pdf, Aug. 2011.

[5] N. Vallina-Rodriguez, J. Crowcroft, "Energy Management Techniques in Modern Mobile Handsets", *IEEE Communications Surveys & Tutorials*, vol. PP, no. 99, pp. 1-20, Feb. 2012.

[6] A. P. Miettinen, & J. K. Nurminen, "Energy efficiency of mobile clients in cloud computing", in proc. of the *2nd USENIX conference on hot topics in cloud computing*, pp. 4-4, Jun. 2010.

[7] K. Kumar, Yung-Hsiang Lu, "Cloud Computing for Mobile Users: Can Offloading Computation Save Energy?", *IEEE Computer Society*, vol. 43, no. 4, pp. 51-56, Apr. 2010.

[8] A. Schulman, V. Navda, R. Ramjee, N. Spring, P. Deshpande, C. Grunewald, & V. N. Padmanabhan, "Bartendr: a practical approach to energy-aware cellular data scheduling", in proc. of the *ACM 16th intl. conf. on Mobile computing and networking,* pp. 85-96, Sep. 2010.

[9] B. Aggarwal, P. Chitnis, A. Dey, K. Jain, V. Navda, V. N. Padmanabhan, & N. Spring, "Stratus: energy-efficient mobile communication using cloud support", *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 477-478, 2011.

[10] D. Kovachev, Tian Yu, R. Klamma, "Adaptive Computation Offloading from Mobile Devices into the Cloud", in proc. of the *IEEE 10th intl. Symposium on Parallel and Distributed Processing with Applications (ISPA)*, pp. 784-791, Jul. 2012.

[11] B. G. Chun, S. Ihm, P. Maniatis, M. Naik, & A. Patti, "Clonecloud: elastic execution between mobile device and cloud", in proc. of the *ACM 6th conf. on Computer systems*, pp. 301-314, Apr. 2011.

[12] Shih-Hao Hung, Chi-Sheng Shih, Jeng-Peng Shieh, Chen-Pang Lee, Yi-Hsiang Huang, "An Online Migration Environment for Executing Mobile Applications on the Cloud", in proc. of the *IEEE 5th intl. conf. on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS)*, pp. 20-27, June 2011.

[13] Eduardo Cuervo, Aruna Balasubramanian, Dae-ki Cho, Alec Wolman, Stefan Saroiu, Ranveer Chandra, Paramvir Bahl, "MAUI: making smartphones last longer with code offload", in proc. of the *ACM 8th international conference on Mobile systems, applications, and services*, pp. 49-62, Jun. 2010.

[14] Jason Flinn, Dushyanth Narayanan, M. Satyanarayanan, "Self-Tuned Remote Execution for Pervasive Computing", in proc. of the *IEEE 8th Workshop on Hot Topics in Operating Systems*, pp. 61-66, May 2001.

[15] J. N. Silva, L. Veiga, and P. Ferreira, "SPADE – scheduler for parallel and distributed execution from mobile devices", in the proc. of the *ACM 6th intl. workshop on middleware for pervasive and ad-hoc computing*, pp. 25-30, Dec. 2008.

[16] Verdi March, Yan Gu, Erwin Leonardi, George Goh, Markus Kirchberg, Bu Sung Lee, "µCloud: Towards a New Paradigm of Rich Mobile Applications", in proc. of the *8th intl. conf. on Mobile Web Information Systems (MobiWIS 2011)*, Procedia Computer Science, vol. 5, pp. 618-624, 2011.

[17] Android application framework - http://developer.android.com/index.html

[18] E. Chen, S. Ogata, K. Horikawa, "Offloading Android applications to the cloud without customizing Android", in the proc. of the *IEEE intl. conf. on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, pp. 788-793, Mar. 2012.

[19] R. Kemp, N. Palmer, T. Kielmann, and H. Bal, "Cuckoo: a computation offloading framework for smartphones", in the proc. of the *IEEE Computer Society 3rd intl. conf. on Mobile Computing, Applications, and Services (MobiCASE)*, 2010.

[20] S. Kosta, A. Aucinas, Pan Hui, R. Mortier, Xinwen Zhang, "ThinkAir: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading", in proc. of the *IEEE intl. conf. on Computer Communications (INFOCOM)*, pp. 945-953, Mar. 2012.

[21] MapReduce framework - http://static.googleusercontent.com/external_content/untrusted_dlcp/research.google.com/en//archive/mapreduce-osdi04.pdf

[22] Eugene E. Marinelli, "Hyrax: Cloud Computing on Mobile Devices using MapReduce", MS Thesis, School of Computer Science, Carnegie Mellon University, http://reports-archive.adm.cs.cmu.edu/anon/2009/CMU-CS-09-164.pdf, 2009.

[23] Hadoop - http://hadoop.apache.org/

[24] A. Dou, V. Kalogeraki, D. Gunopulos, T. Mielikainen, V. H. Tuulos, "Misco: A MapReduce Framework for Mobile Systems", in proc. of the *ACM 3rd intl. conf. on Pervasive Technologies Related to Assistive Environments (PETRA'10)*, Jun. 2010.

[25] M.D. Kristensen, "Scavenger: Transparent development of efficient cyber foraging applications", in the proc. of the *IEEE 8th intl. conf. on Pervasive Computing and Communications (PerCom)*, pp. 217-226, Apr. 2010.

[26] G. Huerta-Canepa and D. Lee, "A virtual cloud computing provider for mobile devices", in proc. of the *1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond (MCS)*, article no. 6, 2010.

[27] E. Koukoumidis, D. Lymberopoulos, K. Strauss, J. Liu, and D. Burger, "Pocket cloudlets," in Proc. of the *16th Intl. Conf. on Architectural support for programming languages and operating systems (ASPLOS)*, pp. 171-184, Mar. 2011.

[28] Raatikainen et al, Towards Mobile Device Cloud - http://www.cloudsw.org/current-issue/201112226148.

[29] K.A. Hummel, H. Meyer, "Self-Organizing Fair Job Scheduling among Mobile Devices", in the proc. of the *IEEE 2nd intl. conf. on Self-Adaptive and Self-Organizing Systems Workshops (SASOW)*, pp. 230-235, Oct. 2008.

[30] N. Fernando, S.W. Loke, W. Rahayu, "Dynamic Mobile Cloud Computing: Ad Hoc and Opportunistic Job Sharing", in the proc. of the *4th intl. conf. on Utility and Cloud Computing (UCC)*, pp. 281-286, Dec. 2011.

[31] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The Case for VM-Based Cloudlets in Mobile Computing", *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14-23, Oct. 2009.

[32] Hongbin Liang, L.X. Cai, Dijiang Huang, Xuemin Shen, Daiyuan Peng, "An SMDP-Based Service Model for Interdomain Resource Allocation in Mobile Cloud Networks", *IEEE Transactions on Vehicular Technology*, vol. 61, no. 5, pp. 2222-2232, Jun. 2012.

[33] Y. Xu, J. Heidemann, and D. Estrin, "Geography informed Energy Conservation for Ad-hoc Routing", in the proc. of the *ACM/IEEE 7th intl. conf. on Mobile Computing and Net*, 2001, pp. 70–84.

[34] Z.J. Haas, "A new routing protocol for the reconfigurable wireless networks", in proc. of the *IEEE 6th intl. conf. on Universal Personal Communications Record*, vol.2, pp.562-566, Oct 1997.

[35] Theodore S. Rapppaport, "Wireless Communications Principles and Practice", 2nd Ed, Dorling Kindersley (India) Pvt. Ltd.