M^2C^2 : A Mobility Management System for Mobile Cloud Computing

Karan Mitra, Saguna Saguna, Christer Åhlund and Daniel Granlund
Luleå University of Technology, Skellefteå, Sweden
Email: {firstname}.{lastname}@ltu.se

***Abstract*—Mobile devices have become an integral part of our daily lives. Applications running on these devices may avail storage and compute resources from the cloud(s). Further, a mobile device may also connect to heterogeneous access networks (HANs) such as WiFi and LTE to provide ubiquitous network connectivity to mobile applications. These devices have limited resources (compute, storage and battery) that may lead to service disruptions. In this context, mobile cloud computing enables offloading of computing and storage to the cloud. However, applications running on mobile devices using clouds and HANs are prone to unpredictable cloud workloads, network congestion and handoffs. To run these applications efficiently the mobile device requires the best possible cloud and network resources while roaming in HANs. This paper proposes, develops and validates a novel system called M^2C^2 which supports mechanisms for: i.) multihoming, ii.) cloud and network probing, and iii.) cloud and network selection. We built a prototype system and performed extensive experimentation to validate our proposed M^2C^2. Our results analysis shows that the proposed system supports mobility efficiently in mobile cloud computing.**

I. Introduction

The number of active mobile phone users will reach 7.3 billion by the end of 2014 [3]. This includes 1.75 billion smart-phone users. There are a variety of applications running on these devices including those related to utility, healthcare, entertainment, productivity and sports. Most of these mobile applications are transforming towards cloud based applications due to limited resources (compute, storage, memory and battery) on these devices. Spotify[1] and Dropbox[2] are two well known examples of cloud based applications which are used for mobile audio/video streaming and file storage over the Internet, respectively. Cloud computing provides application providers and users with an infinite pool of virtual hardware and software resources. Thus, providing them with significant compute and storage resources at reasonably modest costs [8].

In recent years, industry and academia have focused on developing systems that offload computation and storage to the clouds closest to the users with an aim to minimize end-to-end latency (network and processing delay) and maximize the battery lifetime of devices [9], [11], [13], [16]. For instance, Ha *et al.* [13] shows that computation offloading from Google Glass[3] to a standalone server can provide at least an order of magnitude of performance improvement in terms of faster task processing and energy savings. This is particularly important for applications relating to the area of mobile healthcare, battlefield, and emergency situation-awareness where faster application response time while maximizing battery life of a mobile device is crucial.

It is important to note here that the challenges pertaining to mobile computing are also relevant for mobile cloud computing (MCC). For instance, issues such as handoffs, sporadic network connectivity, high network latency, limited network bandwidth and limited battery lifetime should also be addressed while developing efficient MCC systems [9], [12], [16], [17]. Future mobile nodes/mobile devices (MN) may roam in heterogeneous access networks (HANs), for example, WiFi and 3G where each AN may have different bandwidth, coverage area and stochastic network conditions. Roaming of a MN between different ANs may trigger handoffs that may lead to long delay and high packet losses. Therefore, causing interrupted cloud resource usage. Further, issues associated with cloud resource management should also be addressed for the successful implementation of MCC systems. For example, cloud quality of service (QoS) parameters (CPU, RAM and disk I/O) can vary stochastically based on user demands and workloads [8], [17]. Further, the application components may be distributed across heterogeneous cloud systems adding more complexity to cloud resource management.

The key challenges addressed in this paper relating to MCC are: i.) a MN needs to be aware of QoS related to HANs as well as clouds [12], [17]; and ii.) a MN should be able to select an access network and a cloud that guarantees network and cloud QoS to support users' application requirements. Our key contributions in this paper are:

1) We propose, develop and validate a novel system for mobility management in MCC called M^2C^2: A Mobility Management System for Mobile Cloud Computing.
2) M^2C^2 system includes network and cloud probing mechanisms to provide network- and cloud-QoS awareness to the MN when it roams in HANs. It supports multihoming and incorporates metrics for cloud and network selection based on users' application requirements and network and cloud load.

To the best of our knowledge, this is the first paper that tackles the challenge of mobility management by considering both cloud and network resources while the users are on-the-move in mobile cloud computing systems.

This paper is organized as follows: Section II presents our proposed system, M^2C^2. Section III discusses in detail our prototype implementation and results analysis. Section IV presents the related work and section V presents the conclusion and future work.

[1] www.spotify.com
[2] www.dropbox.com
[3] https://www.google.com/glass/start/

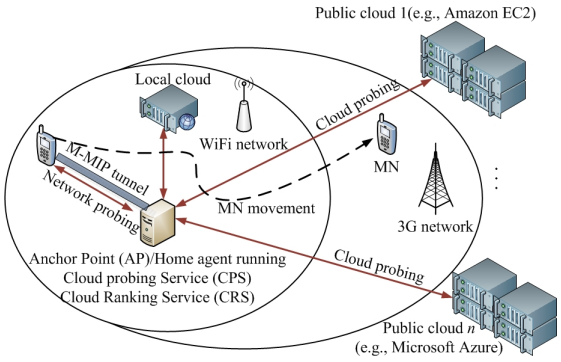

Fig. 1. M^2C^2 : A Mobility Management System for Mobile Cloud Computing.

II. M^2C^2: A Mobility Management System for Mobile Cloud Computing

This section presents M^2C^2, a novel system for mobility management in MCC (see Fig. 1). To support applications in MCC, M^2C^2 incorporates several network and cloud entities. These include: local and public clouds, Home Agent (HA), Cloud Probing Service (CPS), Cloud Ranking Service (CRS), mobile node (MN), WiFi and 3G networks and the Anchor Point (AP). The AP supports cloud and network awareness by providing functionality such as cloud and network probing, and multihoming. In particular, the AP acts as the HA and assists a MN in tracking QoS statistics of access networks by providing network path probing and packet flow re-direction functionalities. The AP can also run the Cloud Probing Service (CPS) to probe local and public clouds; and the Cloud Ranking Service (CRS) to select the best cloud where applications may offload computation and storage. The MN constantly keeps track of the clouds and networks via these M^2C^2 entities such that the applications can decide which network and cloud to select when the user is roaming in HANs. In the following sub-sections, we discuss M^2C^2 in detail.

A. Network Probing and Selection

M^2C^2 incorporates Multi-homed Mobile IP (M-MIP) [7] as a multihomed mobility management protocol to facilitate soft, low latency handoffs with minimal packet loss. M-MIP enables a MN to connect to several access networks simultaneously before initiating the handoff process i.e., a MN performs network discovery, network configuration, and network registration in advance for all available wireless networks. The MN periodically probes the registered network interfaces to select a target network for handoff without disconnecting from the previous network interface, thereby minimizing network delay and packet losses during the handoff process. For selecting the best available network interface $i \in I$, the MN performs passive network path probing to compute the Relative Network Load (RNL_i) metric [7] for each i. A MN probes all the available networks I by periodically sending Binding Update (BU) messages to Home Agent (HA)/Correspondent Node (CN) and receiving the corresponding Binding Acknowledgment (BA) messages from HA/CN (see Fig. 1). The amount of time spent between sending a BU message to HA/CN by the MN and receiving the corresponding BA message from the HA/CN to the MN is the round trip time (RTT). Finally, using the RTT values, RNL metric is computed to estimate the load on access networks. The RNL metric is computed as follows [7]:

$$RNL = Z_n + cJ_n \tag{1}$$

$$Z_n = \frac{1}{h}RTT_n + \frac{h-1}{h}Z_{n-1} \tag{2}$$

$$RTT_n = R_n - S_n \tag{3}$$

$$D_n = RTT_n - RTT_{n-1} \tag{4}$$

$$J_n = \frac{1}{h}|D_n| + \frac{h-1}{h}J_{n-1} \tag{5}$$

S_n is the time to send the BU packet $n \in N$ from the MN to HA/CN. R_n is the time of arrival of the BA packet (corresponding to packet n from the HA/CN to the MN). h is the history window for calculating the weighted average, where $h = 5$ is considered to be an optimal value [7]. c represents the weight of the RTT value compared to the RTT jitter value. For example, if $c = 5$, it means that the RTT jitter value contributes 5 times more than the RTT value. Finally, the variables Z, D and J are initialized as: $Z_0 = RTT_0$; $D_0 = 0$; and $J_0 = D_1$. The network with lower RNL value is the target for handoff. We now discuss our method for cloud selection.

B. Cloud Probing and Selection

For QoS-aware cloud selection, M^2C^2 also includes the mechanisms for cloud probing and cloud ranking. In particular, it enables the Anchor Point (AP) or any other dedicated network element on a particular access network to run the Cloud Probing Service (CPS) as shown in Fig. 1. The CPS keeps track of the QoS statistics of public and local clouds by probing them at regular time intervals using the RESTful Application Programming Interfaces (APIs). The Cloud Ranking Service (CRS) retrieves the stored QoS stats from CPS and computes the rank for each cloud based on a number of criteria including: application type, CPU utilization, memory utilization and disk I/O operations. The MN using the RESTful API can then retrieve the best cloud $k \in K$ based on the computed rank for a particular application. The MN will then use the new cloud for cloud-based processing and storage. The URL of the RESTful API is as follows: `http://<CRS IP addr.></cloudrankservice>`. The tag `<CRS IP addr.>` represents the IP address of the CRS; the tag: `</cloudrankservice>` is the web resource where the CRS is running. The MN makes a GET call to this URL and retrieves the best cloud as an HTTP response. Further a GET call to `http://<CRS IP addr.></cloudrankservice><appl. type>` URL retrieves the best cloud based on the supplied application type where the tag: `<appl. type>` represents the different type of applications, for example, a video streaming service or critical medial response application.

The best cloud is selected by computing the rank by the CRS. In this paper, we consider Simple Additive Weighting (SAW) to compute the cloud rank. SAW is a simple, yet a very effective multi-criteria decision making method (MCDM). The clouds are ranked using the following formula:

$$\Re_k = w_{ma}(QoS_{nk}) + (1 - w_{ma})(Cost_{nk}) \tag{6}$$

where the parameter $n \in N$ represents the n^{th} QoS and $Cost$ parameter for cloud $k \in K$. The parameter $\Re_k$ represents the rank of k^{th} cloud and w_{ma} represents the weights associated

with each parameter QoS_{nk} and $Cost_{nk}$ for each application $a \in A$ where $\sum_{ma=0}^{M} w_{ma} = 1$. The parameter $Cost_{nk}$ can be monitory cost and any other cost related to using a cloud service; the parameter QoS_{nk} may represent several QoS parameters such as, CPU utilization, network throughput and end-to-end latency. Some of the parameters such as cost and CPU utilization needs to be minimized, whereas parameters such as throughput needs to be maximized. Therefore, we need to normalize the parameters using the generic equations as follows:

To maximize:

$$QoS_{nk} = \begin{cases} 1, & if\ QoS_{nk} \geq max(QoS_{nk}) \\ \frac{QoS_{nk} - QoS_{nk_{min}}}{QoS_{nk_{max}} - QoS_{nk_{min}}}, & if\ min(QoS_{nk}) < QoS_{nk} \\ & < max(QoS_{nk}) \\ 0, & if\ QoS_{nk} \leq min(QoS_{nk}) \end{cases} \tag{7}$$

To minimize:

$$Cost_{nk} = \begin{cases} 1, & if\ cost_{nk} \leq min(cost_{nk}) \\ \frac{Cost_{nk_{max}} - Cost_{nk}}{Cost_{nk_{max}} - Cost_{nk_{min}}}, & if\ min(Cost_{nk}) < Cost_{nk} \\ & < max(Cost_{nk}) \\ 0, & if\ Cost_{nk} \geq max(Cost_{nk}) \end{cases} \tag{8}$$

The cloud with the highest $\Re_k$ is selected as the best cloud.

An algorithm for cloud and network selection used by MN in M^2C^2 is shown below. The algorithm shows that as soon as the MN starts roaming in HANs, it constantly computes the RNL_i metric (using Eq. 1 to 5) $\forall\, I$ and retrieves the best cloud k from AP. Using the RNL_I values, the MN makes soft handoffs with low latency and packet loss that does not adversely hamper the applications running on a MN. At the same time, the MN offloads the computation/storage based on the selected k.

ALGORITHM 1: An algorithm for cloud and network selection for mobile cloud computing

Input: Avaliable networks, *I*; available clouds, *K*
Output: Perform activity recogntion on selected cloud, $k \in K$ using selected network, $i \in I$

Initialization: $RNL_{i=\{1...n\}} \leftarrow 0$
Using RSSI, discover *I*;
foreach $(i \in I)$ **do**
  if $RSSI_i \geq$ *Threshold*$(RSSI_i)$ **then**
    Connect to *i*
  end
end
foreach $(i \in I)$ **do**
  Establish tunnel with the Home Agent (HA)
end
foreach $(i \in I)$ **do**
  Compute RNL_i
end
Connect to *i* with min(RNL_i)
Retrieve the best cloud *k* from the Anchor Point (AP)
Perform activity recognition on *k* while connected to *i*

III. Results Analysis

This section presents the results analysis of M^2C^2. We demonstrate how M^2C^2 determines the best cloud k and network i for QoS-aware task processing in a hybrid cloud scenario (considering local and public clouds) while a user roams in HANs. In this context, we developed an activity recognition (AR) application that uses various sensors including accelerometer, RFID and GPS to determine user activities on clouds. We expect a rise in the usage of these type of applications in the near future; applications such as AR can be used in several areas such as personal fitness, emergency management, health care, safer industry and mining operations and battlefield situation-awareness, to name a few.

An AR application usually requires a large amount of sensor data collection, fast data processing (using robust AR algorithms), and timely results delivery to the end user. All these necessary steps, if performed on the mobile device (MN), may lead to reduced battery lifetime and an increase in latency [12], [13], [16]. Therefore, we assert that AR is performed on best cloud(s) instead on the MN to maximize MN's battery lifetime while providing low latency computation. The major challenges posed by AR applications running on cloud are: (i.) efficient data collection from sensors placed in the user environment, for example, light and touch sensors in a smart home; (ii.) timely sensor data delivery to AR application; (iii.) timely AR using the provided sensor data; and (iv.) timely results delivery to the end user. In this paper, we focus on tasks (ii.), (iii.) and (iv.) For tasks (ii.) and (iv.), the MN needs to select a network that provides high bandwidth, low packet losses and delay. For task (iii.), the MN needs to select the best cloud that efficiently processes sensory data and determines user activity in a timely manner.

A. Prototype Implementation and Experimental Setup

We considered a scenario where a user using his/her MN roams seamlessly in HANs while an AR application running on the MN collects sensory data from the user and sends it to the best cloud for processing. Once the user activity(s) is recognized on the cloud, the results are then sent back to the MN. To validate M^2C^2, we developed a system prototype and a testbed consisting of several components (see Fig. 1) including: the implementation of M-MIP mobility management protocol, the cloud probing service (CPS) running on anchor point (AP), the cloud ranking service (CRS) running on AP, the AR application running on MN, and the AR service running on several cloud instances. The M-MIP prototype was developed using C++ to handle multihoming in HANs. In the current version of M^2C^2 prototype, the AP runs the Home Agent (HA) service. It runs the CPS to probe all the cloud services, where users' applications are deployed. It also runs the CRS to determine the best cloud. As mentioned in the previous section, the CPS and CRS can also be deployed as a separate network entities providing dedicated cloud probing and cloud ranking functionalities.

The CPS was implemented for both public as well as the local clouds. For public clouds, we considered Amazon EC2 micro instances running in "eu-west-1c" region. To probe the Amazon EC2 instances, we considered the Apache jclouds multi-cloud toolkit Java APIs [2]. The Apache jclouds API supports several cloud providers and can be used to programmatically orchestrate cloud operations such as starting/stopping/terminating the virtual machines. It can also be used to gather QoS stats for the Amazon cloud instances.

We chose jclouds API over Amazon's CloudWatch API [1] as it supports multiple cloud providers. For gathering QoS stats from the local cloud, we considered SIGAR Java API [4] that provide statistics such as CPU utilization and memory usage. The CPS was running as a RESTful web service on all clouds. The CRS sent the GET calls to these web services to retrieve the QoS stats from all clouds.The AP also runs the CRS to compute cloud rank $\Re_k \forall K$ (using Eq. 6 to 8) to determine the single best cloud, k where users' AR should be performed. We implemented the Situation- And Context-Aware Activity Recognition (SACAAR) algorithm [15] as a RESTful web service on the clouds (local and public) to determine users' activities. The MN subscribes to the CRS and retrieves the URL of k^{th} cloud and then sends the sensory readings to it for performing AR.

For experimentation, we setup a testbed (see Fig. 1) incorporating WiFi and 3G wireless networks as well as cloud instances running on both local and public cloud infrastructure. In particular, for the local cloud, we considered an Apple Macbook Pro computer with 16 GB RAM and an Intel i7 2.8 GHz processor running the AR web service. On public clouds, we ran two AR web services on two separate Amazon EC2's micro instances in the "eu-west-1c" region (Public cloud 1). Each running instance had its own URL and a public IP address. Our Java based AR application running on the MN then registers the URLs of both local and public clouds in its local database. The AR application then uses these URLs to recognize user activities based on the selected network i as well as the cloud instance k.

B. Experimental Results

Experiment 1: To validate the proposed system, we performed extensive experimentation and considered several scenarios. As the first experiment, we studied the overall latency of performing AR on both local and public clouds. The aim of this experiment was to benchmark the best and worst case scenarios while the user is constantly connected to the Internet via WiFi or 3G networks using M-MIP. Real-time applications, especially those involving humans such as augmented reality, virtual reality, cognitive assistance and activity recognition are extremely sensitive to latency [13], [16]. Ha *et al.* [13] suggested that for these class of applications, the overall latency should not be more than few tens to few hundred milliseconds. As mentioned previously, in this paper, we consider the case of AR where overall latency of performing AR on clouds is the summation of: time taken to send the data from MN to the AR service running on clouds; the time taken by the AR algorithm to determine the user activity(s); and the time taken to get the results back from the AR service to the AR application running on the MN. Fig. 2 shows the cumulative distribution function of the overall latency of performing AR on local and public clouds while using WiFi and 3G networks, respectively. We considered four cases: (A) Local cloud using 3G network; (B) Local cloud using WiFi network; (C) Public cloud using 3G network; and (D) Public cloud using WiFi network. In this experiment, for each case, an average of 3353 ($N = 13412$) API calls were made from the MN AR application to AR services running on clouds.

We can clearly observe from Fig. 2 and table 1 that on an average the overall latency of performing AR on the local

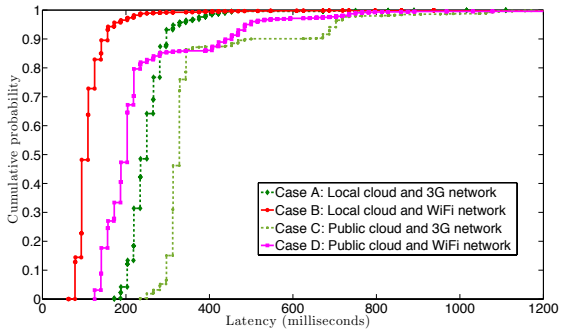

Fig. 2. Cumulative distribution function of overall latency corresponding to performing activity recognition on local and public clouds.

cloud is significantly lower than the public cloud considering both WiFi and 3G networks. One of the major factor contributing to the overall latency was the type of network used where WiFi offers significantly lower round trip times (RTTs) compared to the 3G network. For instance, the average latency of performing AR is more than twice as high in case of local cloud using 3G network (case A) compared to local cloud using WiFi network (case B) (see table I). Secondly, the latency of performing AR on a public cloud using 3G network (case C) is nearly on an average, 130 ms higher than using public cloud using WiFi (case D). This is due to the distance between a MN and the cloud which contributes to the increase in overall WAN latency. From this experiment, we conclude that the local cloud performs approximately 60% better than public cloud for AR considering both 3G and WiFi networks. These observations lead us to assert that the computation should mainly be performed at local clouds using WiFi networks, assuming there is sufficient compute resources available at local clouds.

Experiment 2: After benchmarking both local and public clouds using HANs, we performed experiments to determine whether M^2C^2 can select the best cloud and the best network under stochastic network and cloud conditions such as network congestion, handoffs and variation of CPU utilization of cloud resources. As the second set of experiments, we setup the Anchor Point (AP) to run Home Agent (HA), Cloud Probe Service (CPS) and Cloud Ranking Service (CRS). To validate CPS and CRS, we deployed our AR service on two micro instances on a public cloud (Amazon EC2). We randomly generated a synthetic workload using the stress utility [5] on one of the public cloud instances and imposed a load of 100% on the CPU cores. We did not put additional stress on the other cloud instance. Fig. 3 shows the screenshot of the Glassfish server log running the CPS and the CRS. The figure shows that two public cloud instances with IP addresses: 54.77.183.180 (cloud 1) and 54.77.218.113 (Cloud 2) are probed for CPU utilization. As the cloud 1 had lowest average CPU utilization, it was correctly selected by the CRS for AR. As soon as the MN requested the CRS for best cloud for performing AR, the IP address of cloud 1 was provided by the CRS (see Fig. 4).

Fig. 4 shows the output log of AR application running on the MN. As can be observed from the figure, the URL for AR web service consists of the IP address of cloud 1. The sensor reading were then sent to the AR web service listening on this URL. The AR web service then inferred the user activity and

TABLE I. STATISTICS REGARDING OVERALL LATENCY (L) WHILE PERFORMING ACTIVITY RECOGNITION ON LOCAL AND PUBLIC CLOUDS

Case	Avg. L. (ms)	Std. Dev. L. (ms)	Min. L. (ms)	Max. L.(ms)
A	252.27	57.74	172	1110
B	113.90	64.81	62	3063
C	366.52	149.51	234	1844
D	236.04	150.60	125	2407

```
INFO: The total number of Clouds listed in our account are: 2
INFO: The start time is :Tue Sep 09 15:11:51 CEST 2014        1. Cloud probing by CPS
INFO: The end time is :Tue Sep 09 15:16:51 CEST 2014
INFO: Cloud with IP: [54.77.183.180]CPU utilization statistics: 0.67% (avg), 1.69% (max), 0% (min)
INFO: The start time is :Tue Sep 09 15:11:52 CEST 2014
INFO: The end time is :Tue Sep 09 15:16:52 CEST 2014
INFO: Cloud with IP: [54.77.218.113]CPU utilization statistics: 100% (avg), 100% (max), 0% (min)
INFO: Retrieving the best URL...
INFO: The Best cloud URL: [54.77.183.180]                     2. Cloud selection by CRS
INFO: [54.77.183.180]
```

Fig. 3. Logs showing the output of Cloud Probing Service and the Cloud Ranking Service.

sent it back as the response to the MN. The MN then computed the overall latency of performing AR on cloud 1 while using a particular access network (WiFi in this case). We repeated this experiment more than hundred times by varying the CPU load randomly on both cloud instances and established that both CPS and CRS worked as expected.

Experiment 3: In experiment 3, we studied the impact of mobility on AR while the user was on-the-move in HANs (3G and WiFi). The aim of this experiment was to test whether MN using multihoming capabilities can facilitate low latency handoffs with minimal packet losses without adversely affecting the AR application. In this experiment, once the MN started, it received the care-of-address (CoA) from the Anchor Point (AP). We then started the AR application on the MN. The AR application probed the CRS for the best cloud instance, k (see Fig. 3 as an example) using our implemented APIs. At the same time, the MN probed the HA running on AP to select the best network, i by computing the RNL metric $\forall I$. Fig. 5 shows the RNL values for both WiFi and 3G networks. It can be observed from Fig. 5 that the RNL values for WiFi network were less stable compared to the 3G network. At time $t = 13\,sec$, the MN made a successful handoff without packet loss to the 3G network and stayed with that network for approx. 4 seconds and made another handoff to the WiFi network. Then at $t = 23\,sec$, the MN performed another successful handoff from WiFi to 3G network. Fig. 6 shows the results of these handoffs on the latency of performing AR. As can be observed from Fig. 6, the latency of performing AR on a 3G network is higher than WiFi network. *Most*

```
Mobile Cloud Application is running...
INFO: Getting sensor readings:
Sep 09, 2014 4:23:11 PM com.ltu.mobilecloud.mobilecloudapplications.MobileCloudApplications
performActivityRecognition
INFO:
http://54.77.183.180:8080/ActivityRecogntionAppService/webresources/getactivity?type=xml&x=4
43&y=701&z=659
Sep 09, 2014 4:23:12 PM com.ltu.mobilecloud.mobilecloudapplications.MobileCloudApplications
performActivityRecognition
INFO: The recognized activity is: <InferredActivity>sitting</InferredActivity>
Sep 09, 2014 4:23:12 PM com.ltu.mobilecloud.mobilecloudapplications.MobileCloudApplications
performActivityRecognition
INFO: The end-to-end latency for performing AR on cloud:  54.77.183.180:8080 is: 176.0 milliseconds
```

Fig. 4. Logs showing the output of activity recognition (AR) application running on the mobile node. The output shows the URL of the AR web service running on the selected cloud instance. Inferred activity, for example, "sitting" as well as the latency of inferring the activity is also shown.

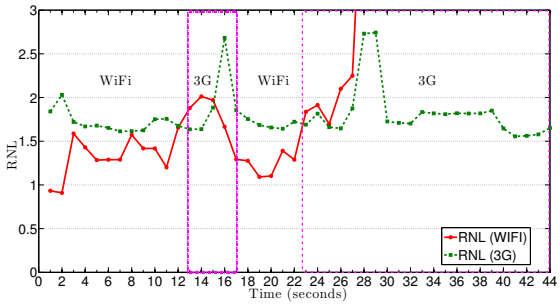

Fig. 5. Figure showing the variation in RNL values as well as handoffs for WiFI and 3G networks. The first handoff between WiFi and 3G occurs at time 13 second and the second handoff occurs at time 23 second.

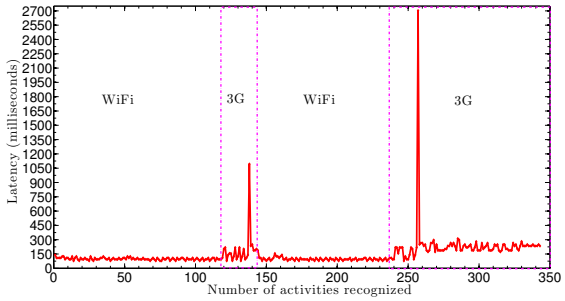

Fig. 6. Figure depicting the overall latency of performing AR on local cloud. The latency becomes higher between 13 and 17 seconds and between 23 and 44 seconds when the MN was connected to 3G network. As can be seen, no AR responses were lost due to M-MIP.

importantly, we concluded from these experiments that during the handoff process, the MN did not observe any packet loss and no API calls to the AR web service for inferring user activities (between the MN and the AR web services running on the public clouds) were lost. Similarly, no responses regarding the inferred activity between the AR web service and the MN were lost.

Our results validate that M^2C^2 can successfully support user-oriented applications in HANs while the users are on-the-move. M^2C^2 successfully tracked the QoS parameters of all clouds K as well as wireless networks I, to determine the best cloud k and network i for processing user requests in a hybrid cloud and heterogeneous access network scenario.

C. Discussion and lessons learnt

Our experiments demonstrated that M^2C^2 efficiently supports mobility in MCC. Our system is built using RESTful principles and provides cloud and network probing, and cloud and network selection APIs to the applications running on MN. We validated M^2C^2 by considering an AR application as a case study. However, the proposed system can easily be extended to include several other applications such as optical character recognition, virtual reality and speech synthesis. From experimentation 1 we learnt that in even in the best case scenario (see Fig 2. Case B), the average latency can still be high for certain type of applications (e.g., virtual reality and speech recognition) thereby creating the need for preprocessing data on the MN itself. We intend to investigate this issue as a future work. For experiment 2, we considered CPU utilization as the QoS parameter. This is based on the fact that SACAAR algorithm [15] is primarily CPU intensive and does not consume significant memory and disk space.

However, with the introduction of new applications with M^2C^2 as a future work, we would also like to include additional parameters such as disk I/O and memory utilization. These parameters can easily be supported within our architecture. In regards to this, we noticed that public cloud providers such as Amazon and Rackspace do not provide all QoS metrics such as memory utilization and supports different APIs. We partially mitigated this challenge by considering the jclouds [2] API. However, we may need to integrate methods [6] that can obtain other QoS parameters directly from the virtual machines running on public clouds.

In all, M^2C^2 is a fully integrated system that offers several novel mechanisms missing in the state-of-the-art research [11], [13], [14]. These include multihoming, cloud and network awareness, and cloud and network selection in HANs. We foresee the usage of our system in several areas including (but not limited to): mobile health care and emergency management and cognitive assistance.

IV. Related Work

MCC enables computation and storage offloading to the clouds with an aim to maximize MNs battery, storage and processing capacity. Thus, making MCC extremely valuable in the areas such as mobile healthcare and battlefield. In the recent years, systems such as MAUI [11], CloneCloud [10] and Cloudlets [16] were proposed to facilitate MCC. These systems use techniques such VM migration and code offloading to the local or remote clouds with an aim to increase MNs battery lifetime and reduce task processing latency. The results from these systems have established that code offloading and VM state migration from the MN to clouds (local or public) can be extremely beneficial. However, [10], [11], [13], [16] have not considered the impact of user mobility on applications running on the MN. For example, while roaming in HANs, a MN may handoff from one access to another leading to high latency and packet losses. Further, we assert that these works [10], [11], [13], [16] do not consider the challenge of cloud and network selection based on monitored QoS parameters such as CPU utilization and network load [12], [17]. These factors may significantly impact applications running on the MN. To support mobility in MCC, the MN needs to be aware of the stochastic network and cloud conditions. Further, the MN should select the best cloud and the network while the users are on-the-move in HANs. Compared to the state-of-the-art research in the area of MCC [10]–[13], [16], [17], in this paper, we have built and validated a cloud and network probing mechanism for the selection of the best avaliable network and cloud resources while enabling mobility in HANs.

V. Conclusion and Future Work

This paper presented M^2C^2, a novel system for mobility management in mobile cloud computing systems. The proposed system enables users to roam seamlessly in heterogeneous access networks while accessing the best possible resources from both local and public clouds. Our proposed system supports multihoming and selects the best network and the best cloud by probing and ranking available cloud and network resources for stochastic conditions such as network congestion and variation in cloud workloads. For validating M^2C^2, we developed a system prototype and performed extensive experimentation. Our results show that our system can easily support applications such as activity recognition that require low-latency and minimal packet losses. To the best of our knowledge, M^2C^2 is the first such system to support mobility management in mobile cloud computing systems by offering capabilities such as multihoming, cloud and network probing, and selection of best cloud and network.

As future work, we aim to integrate a larger spectrum of applications within M^2C^2, for example, applications supporting health care, aged care, safer mining operations and emergency management.

References

[1] Amazon cloudwatch, http://aws.amazon.com/cloudwatch/, [online] access date: 26/02/14.

[2] Apache jclouds multi-cloud toolkit, https://jclouds.apache.org/, [online] access date: 09/09/14.

[3] Itu-t: Ict facts and figures 2014, https://www.itu.int/en/itu-d/statistics/documents/facts/ictfactsfigures2014-e.pdf, [online] access date: 1/09/14.

[4] Sigar−system information gatherer api, "http://www.hyperic.com/products/sigar", [online] access date: 01/10/13.

[5] Stress−workload generator for posix systems, http://people.seas.harvard.edu/ apw/stress/, [online] access date: 01/08/14.

[6] K. Alhamazani, R. Ranjan, K. Mitra, P.P. Jayaraman, Z. Huang, L. Wang, and F. Rabhi. Clams: Cross-layer multi-cloud application monitoring-as-a-service framework. In *Proceedings of the 11th IEEE International Conference on Services Computing*. IEEE, 2014.

[7] K. Andersson, D. Granlund, and C. Åhlund. M4: multimedia mobility manager: a seamless mobility management architecture supporting multimedia applications. In *Proceedings of the 6th international conference on Mobile and ubiquitous multimedia*, pages 6–13, 2007.

[8] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. Above the clouds: A berkeley view of cloud computing. Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley, Feb 2009.

[9] P. Bahl, R. Y. Han, L. E. Li, and M. Satyanarayanan. Advancing the state of mobile cloud computing. In *Proceedings of the third ACM workshop on Mobile cloud computing and services*, pages 21–28. ACM, 2012.

[10] B. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti. Clonecloud: Elastic execution between mobile device and cloud. In *Proceedings of the Sixth Conference on Computer Systems*, EuroSys '11, pages 301–314, New York, NY, USA, 2011. ACM.

[11] E. Cuervo, A. Balasubramanian, D. Cho, A. Wolman, S. Saroiu, R. Chandra, and 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, pages 49–62, New York, NY, USA, 2010. ACM.

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

[13] K. Ha, Z. Chen, W. Hu, W. Richter, P. Pillai, and M. Satyanarayanan. Towards wearable cognitive assistance. In *Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '14, pages 68–81, New York, NY, USA, 2014. ACM.

[14] M.R. Rahimi, N. Venkatasubramanian, and A.V. Vasilakos. Music: Mobility-aware optimal service allocation in mobile cloud computing. In *Cloud Computing (CLOUD), 2013 IEEE Sixth International Conference on*, pages 75–82, June 2013.

[15] S. Saguna, A. Zaslavsky, and D. Chakraborty. Complex activity recognition using context-driven activity theory and activity signatures. *ACM Trans. Comput.-Hum. Interact.*, 20(6):32:1–32:34, December 2013.

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

[17] Y. Xu. and S. Mao. A survey of mobile cloud computing for rich media applications. *Wireless Communications, IEEE*, 20(3):46–53, 2013.