Journal of Theoretical and Applied Information Technology
15th July 2018. Vol.96. No 13

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

RESOURCE-AWARENESS: A STRATEGY FOR RESOURCE OPTIMIZATION AND RELEVANT SERVICE DISCOVERY IN AD-HOC MOBILE CLOUD

[1]DOMINIC AFURO EGBE, [2]BETHEL MUTANGA MURIMO

[1]Department of of Information and Communication Technology, Durban University of Technology (DUT), South Africa

[2]Department of Information and Communication Technology, Mangosuthu University of Technology, Durban, South Africa

E-mail: [1]domafuro@gmail.com , [2]mutangamb@mut.ac.za,

ABSTRACT

The challenges of limitation in devices' resources and dynamic context is an inherent characteristic of mobile environments. These challenges have strong implication on service discovery efficiency. While service discovery operations may create resource-burden on mobile devices, service-relevance is impacted by changes in device context. Generally, discovery mechanisms aim to discover services relevant to consumers' requirements hence service-relevance is based on service functionalities. However, due to changing context in mobile environments, service functionalities alone are insufficient to address service-relevance with regards to resource capability. Consequently, discovered services may fail to match the resource capabilities of client devices, leading to resources wastage. Addressing this challenge requires proactive discovery mechanisms that can adapt to context change based on devices' resource capabilities and service functionalities. In this paper we designed and prototyped an adaptive service discovery mechanism. The mechanism monitors client devices to collect context data used to adapt to changing resource-context before discovering services. This approach recorded high precision and recall rates and reduced processing time, while relative quality of service discovery was significantly enhanced - meaning resource usage in optimized.

Keywords: *Resource-Awareness, Adaptive, Resource-Efficient, Service Discovery, Relevant Services.*

1. INTRODUCTION

The increasing quest to achieve ubiquitous computing and the surge in the number of e-service providers has resulted in Ad-hoc Mobile Cloud Computing (AMC) gaining significant research attention in recent years [1].

The widespread use of mobile devices and increasing availability of fast wireless mobile Internet connectivity are, among others, the driving factors behind AMC computing paradigm [2]. This paradigm exploits advancements in the capabilities of current mobile devices to create low-cost, ad-hoc or opportunistic resource provisioning platform. Over the last decade, mobile devices have evolved and gotten better in terms of hardware and functionality. Nonetheless, these devices are still subject to the impediment of resource limitation that centres on limited memory, storage capacity and battery dependence [3] [4].

To address this characteristic resource challenge, Mobile Cloud Computing (MCC) has been the ideal computing infrastructure to support mobile devices [4]. The idea takes advantage of the Cloud infrastructure to provide scalable server-side processing. This kind of processing helps to relief client devices of computationally complex and resource-intensive operations. MCC also leverages the potentials of mobile devices by providing remote data storage, access to vast online resources, and computational capabilities [5].

Although MCC provides efficient mechanism for e-services delivery, the cost of IT infrastructure makes it very expensive to fully realize in remote areas. Issues such as inaccessibility of the Cloud due to weak or absence of Internet connectivity, high Internet subscription charges, high energy consumption (due to wireless uplink connection from mobile devices) and latency are other potential research challenges facing mobile MCC [4], [6].

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Furthermore, service relevance is influenced by the dynamic context of mobile devices while discovery efficiency hinges on the resource capability of the device consuming the service. These issues are still open in the MCC domain [7]. In attempt to address these challenges, AMC computing model proposes a network of mobile devices that acts as Cloud provider by enabling each device to provide its resources to others within the network [1], [8]. Essentially, this model has offered leverage for the evolution of the concept of mobile web services, which advocates the need to enables mobile devices not only as conventional web service requesters but as providers as well [9]. These web services are provisioned from mobile devices, accessed and invoked by peer nodes [10].

Nevertheless, being a service offering platform, discovering relevant services is, in addition to other benefits, a fundamental requirement that defines the overall usefulness of AMC. Unfortunately, this vital requirement for service discovery efficiency faces two unique challenges:

1.2 Unpredictable Context Change

Generally, mobile environments are prone to unanticipated changes in local context, which may include user preferences, environmental variables and hardware resources (available battery and memory). Due to the unique nature of AMC, hardware resource context plays a vital role in service discovery. For instance, if a device's battery is in say level "at the first launch of a service request and after a period, when a similar operation is performed the battery level drops to say x-k, then there is a change in resource context (battery) of the device. Such unanticipated context change can impede the discovery of relevant services, because discovered services might no longer match the capabilities of the device in its current resource state, resulting in resource wastage and low client satisfaction [11]. To address the impact of dynamic context on service discovery, context-aware applications are emerging to exploit the dynamism in mobile environments to provide new and innovative services to mobile consumers [12].

1.3 Resource Limitation

Resource limitation of mobile devices poses another critical challenge because service discovery is generally considered a resource-intensive process, especially in mobile domains [13]. The challenge of inadequate resources is overwhelming in AMC environment because resource-constrained devices act as service providers in addition to running other conventional tasks. Another challenge is that, resources are not only scarce, they are dynamic as well. Addressing these challenges makes it imperative to advocate the development of adaptive service discovery mechanisms that can weigh the resource capability of client devices and use the weight to determine which services would be relevant to the device. The lack of such techniques can lead to a situation where a device may run out of limited resources while consuming and or providing a service.

Generally, current literature shows several scholarly proposals that offered various service discovery techniques. However, most of these techniques are traditionally designed for Cloud and Mobile Cloud discovery mechanisms. Therefore, these techniques do not adequately consider the challenges with regards to the peculiarity in AMC [14], [15], [16]. On the other hand, current state-of-the-art in AMC has not fully explored hardware resource as a component of device context in determining the relevance of discovered services. Whereas the relevance of a service is measured not only by the service's ability to fulfil the required functionality but also by matching the resource capabilities of the target device [17]–[19].

Above all, the fact that in AMC hardware resources are limited coupled with the resource-intensive nature of service discovery operations, emphasizes the necessity to design resource-aware mechanisms for discovering services in AMC. Such mechanisms will introduce a balance between discovering relevant services and efficient resource utilization.

Broadly, there are two schools of thought or paradigms in the solution space of AMC service discovery. The first leverages on the resource-rich Cloud to develop agent-based or broker-based service discovery frameworks. The idea is to push resource-intensive service discovery operations to the Cloud to minimize resource burden placed on mobile devices.

Example, in [17] the authors developed a complex semantic-based service discovery process mediated by a Cloud broker. Similarly, [10] proposed a cloud-based framework for mobile web service discovery for resource constrained environments.

The framework adopts the typical Cloud model to offer discovery-as-a-service – DaaS. Also, in [20], [21] context-aware Offloading Decision and Discovery Algorithms were implemented to determine when it is beneficial to offload energy-

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

intensive execution to the Cloud and discover the right service respectively.

The second school of thought advocates efficient service discovery in pure AMC settings, focusing on using device context to determine service relevance. In exploring this idea, [22] highlighted the core requirements for a reliable mechanism for efficient service discovery in AMC. While [11] developed a device-aware service discovery mechanism and a limited resource-aware service adaptation for pervasive environments is proposed by [23]. In addition, other authors have proposed a service discovery mechanism that utilizes user preferences, user rating and device profile to achieve relevant service discovery and rating [24].

While the first paradigm does not fully reflect the ideal AMC (infrastructure-less), proposals in the second school of thought are not focused on studying the impact of device resource capability on relevant service discovery.

To bridge the gap between the above two schools of thought, this paper focuses on achieving an Ad hoc service discovery platform. The aim is to develop a resource-aware service discovery mechanism for AMC environments. To achieve this aim, we consider resource capabilities of devices as vital constituent of device context. The emphasis is on utilizing device context as a tool to enhance relevant service discovery in an adaptive and resource-efficient manner.

In the remainder of the paper, web service and web service discovery are used interchangeably with service and service discovery respectively.

The remainder of this paper is organized as follows: section 2 describes three levels of resource usage optimization in AMC. While section 3, presents the proposed solution approach, which includes formulation of resource-aware service discovery and ranking algorithms. The architecture of the proposed mechanism is then described in section 4. Prototype results and analysis are presented in section 5. The paper is concluded in section 6.

2. RESOURCE OPTIMIZATION IN AMC SERVICE DISCOVERY

A good service discovery mechanism for AMC should be lightweight or not resource-intensive. That is, it should adopt techniques that do not weigh down the resources of mobile devices. Generally, resource-intensiveness is introduced using semantic or ontology techniques, which are usually employed to enhance service discovery efficiency [25]. Basically, there are two levels in the service discovery process where resource-intensiveness can be minimized as discussed in this section:

2.1 Web Service Description Level

Web service description is a fundamental part of the web service development cycle because it makes it possible to group, discover and invoke web services. Web services employ the Cloud concept of software as a service (SaaS) to deliver software entities over the network. Therefore, for a web service to be discovered and subsequently invoked, it must first be specified or described. Web Service Description Language (WSDL) is the de facto standard for specifying web services.

A web service description technically consists of the information model, functional capabilities, non-functional parameters, and the technical specifications of a service. Essentially, the technique for web service specification has the tendency to introduce resource burden, especially in AMC, where resource scarcity is a critical challenge [19]. Consequently, resource usage optimization in AMC must begin from service description level, by adopting a lightweight service description approach.

2.2 Web Service Matchmaking Level

Matchmaking operation is a core component in the service discovery task because it that determines the relevance of returned services with respect to requesters' need. This discovery component deals with the intelligent decision making of comparing users' requirement against available services. Playing a decision-making role denotes that knowledge is being processed at this stage to enable discovery mechanisms to decide which services to retrieve or leave out. In any case, web service matchmaking requires computational resources. Consequently, it is imperative that the design and implementation of matchmakers for AMC environments take into consideration resources scarcity.

However, increasing demand for high performance in service discovery with regards to robustness, accuracy, and efficiency has intensified research efforts enhance matchmaking in a manner that does not favour mobile devices. Therefore, the state-of-the-art in Cloud and Mobile Cloud computing advocates semantic enhancements to service matchmaking [13], [15]. Unfortunately,

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

though such semantic enhancements bring huge improvement, they equally create additional computational burdens [13], [26].

The above underscores the fact that optimizing resource utilization at the service matchmaking level cannot be overlooked in the context of achieving efficient service discovery in AMC.

3. RESOURCE-AWARE SERVICE DESCRIPTION AND MATCKMAKING

Resource-awareness plays key role in AMC service discovery. This role is, in this paper, considered key to determining the relevance of discovered services. Current solution approaches in the literature have predominantly used other context parameters such as QoS, user ratings and preferences, device features etc. [11], [24]. However, we argue that the definition of service relevance should be extended beyond meeting users' requirements to also matching the client device's resource capabilities. To establish the forgoing argument, we propose the incorporation of device context into web service description and matchmaking processes in place of semantic enhancement. This approach is aimed first, at enhancing the discovery of relevant services. And second, minimizing the huge computational demand associated with semantic techniques.

As revealed in section 2, resource usage optimization can be ensured at two critical levels in the web service discovery process – service description and matchmaking levels. Consequently, this section presents two approaches of utilizing device context to achieve optimal resource usage in AMC service discovery.

3.1 Context-based Service Description

Lack of expressiveness and resource-intensiveness are respectively the weakness of syntactic and semantic service description approaches. That is, while syntactic service description can only capture limited non-functional service parameters, their semantic counterparts require huge computational resources, which is not suitable for AMC environment. Achieving a balance between these constraints for enhancing relevant service discovery in AMC environment is challenging.

In this section, a resource-friendly approach to service description based on Web Service Description Language Mobile (WSDL-M) is introduced [11]. This approach uses WSDL-M to provide a light-weight and more expressive service description that captures device context as non-functional parameters. Effectively, by this concept, web service providers are enabled to specify the resource requirement, among other non-functional parameters of offered services in their service description. A graph-based context model as depicted in Figure 1 is utilized in this service description approach. From Figure 1, in each web service description, providers specify the types of device features support by the offered web service as well as its battery and memory requirements.

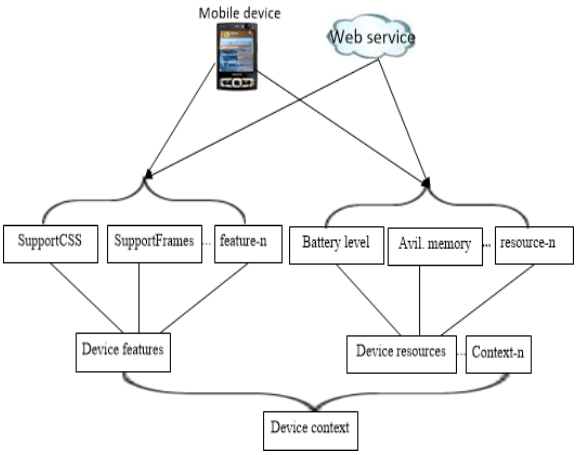

Figure 1: AMC Device Context Model

One of the fundamental service discovery challenges in AMC environment is how to find and invoke services that would function within the capabilities of the client device. Nevertheless, the need for a highly customized web service discovery process that considers the current diversity of client devices has been neglected by many discovery solutions. For instance, different mobile devices support varying features such as frames, callback, cookies etc. Therefore, the rendering or functioning of a web service may be impaired if consumed by a client device that lacks some capabilities that enhances the web service's performance. The proposed context model incorporates device features to help address the above problem. For instance, employing device profile and device resource context can enhance discovery of services that are tailored to the capability of individual devices. Some of the commonly used device features include, but not limited to screen pixel height, maximum href height, screen pixel width,

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

supportCSS, frames, cookies, and support body colour as listed in [11].

With this proposed lightweight and expressive web service description approach, web service capabilities such as battery and memory requirement represented with numeric scales (0, 1). That is, the numeric values symbolize the impact a web service may have on the battery and memory of a client device. These values simply tell whether a service is resource-hungry or not. Example, zero (0) symbolizes that the service is less battery or memory intensive, while one (1) implies the contrary. With this information built into web service descriptions, it becomes possible for web service capabilities to be matched with the client device's resource capabilities during matchmaking. Such device capability-based matching can boost the prospect of realizing a discovery process that optimizes resource usage in addition to enhancing relevant service discovery.

3.2 Context-based Matchmaking and Service Relevancy Ranking

Service matchmaking is one of the most crucial tasks in the service discovery process. On the other hand, in AMC environment, the task is considered the most resource-intensive operation [13], [19] To achieve minimal resource burden, this section presents the discussion of a proposed resource-friendly relevant service discovery algorithm. First, it involves no semantic and ontology techniques, which are the chief cause of computational bottlenecks. Second, it utilizes device context to makes the discovery process adapt to context change. This discovery or matchmaking process is composed of two operations: i) keyword-based matching and ii) web service resource-based relevancy ranking.

3.2.1 Keyword-based Matchmaking Algorithm

Keyword-based matchmaking algorithm is considered appropriate in AMC because it is envisioned that this environment mobile providers will constitute small computing communities with common needs. Therefore, the volume of web services offered through AMC platform can be assumed to be relatively small, compared to other conventional service provisioning platforms like Cloud or Mobile Cloud. The above assumption makes keyword-based matchmaking more ideal compared to the semantic counterpart. This high preference for keyword approach is informed by the fact that semantic techniques are usually necessary in domains with vast number of complex web services. So, deploying semantic in AMC domain will amount to resource waste apart from creating unnecessary computational burden to client devices. Therefore, this paper employs a keyword-based service matchmaking algorithm adapted from [9] as shown in Table-1.

The idea in the algorithm is to measure the semantic distance between the synonyms of a service request keyword and an actual web service. The smaller the semantic distance between them, the closer their relation is. The algorithm therefore returns all web services that are closely related to the keyword used in the service request. This technique aims at reducing the volume of services retrieved by first matching them against the current device's context.

Resource wise, the advantage of this discovery approach is that from the client's side, smaller number of returned services can be viewed as reduced processing time, which corresponds to less resource burden. In addition, the integration of device context parameters into the algorithm enhances that capability of the discovery process to be adaptive to context change. This makes is possible to still discover relevant services when there is a context change because the discovery mechanism tracks such changes and adapts the discovery process accordingly.

3.2.2 Resource-based Matchmaking Algorithm

Ranking of retrieved web services have recently been utilized by several discovery mechanisms to measure the degree of relevance of a service to a client. However, current AMC service discovery mechanisms predominantly adopted the user-centered approach. Such approaches disregard or partly consider the device-centered dimension while focusing mainly on the web service functionality. The problem with this approach is that it is possible for a web service to satisfy a requester's requirement but not match the resource capability of the client's device. Contrary to this, our approach considers a dual perspective where a web service is said to be relevant if it fulfils users' requirement and matches the current context of the target device. Essentially, in this paper, device context refers to the combination of hardware resource capabilities and supported device features.

Journal of Theoretical and Applied Information Technology
15th July 2018. Vol.96. No 13

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Ultimately, the idea of utilizing device context here attempts to strike a balance between fulfilling clients' requirements and not jeopardizing the service provisioning function in an AMC. To achieve this goal the proposed relevancy ranking algorithm depicted in Table 2 is based on device context. The algorithm employs two context measures (relevancy parameters) to compute the relevancy score of each web service retrieved. These parameters are:

Resource weight (R_w): this weighting parameter derives its value from the resource context (battery level, available memory) of the target device. The proposed discovery mechanism assigns resource weights to web services according to their resource requirement as specified in the service description.

For example, a web service with battery and memory requirement status of [0, 1] is assigned a weight of 1 unit. While another service with battery and memory specified as [0, 0] is assigned a weight of 2 units; where zero (0) is interpreted as being resource-efficient with a corresponding score of 1 unit. Similarly, one (1) means resource-inefficient and is assigned a score of 0 units. Therefore, the resource weight (R_w) of a web service is the sum of its battery and memory requirement ratings.

Features weight (F_w): each web service earns a unit score (1) for every device feature that it supports. So, the feature weight is the sum of the score of device features supported. The feature weighting idea enabled us to define the relevance and consequent ranking of a web service as function of its resource and feature weight.

Following the proposed context model of Figure1, it is assumed that there exists:

$$\wp = \{s / s \in \text{published web services}\}$$
$$W = \{w / w \in \text{services that match keyword}\}$$
$$C = \{c / c \in \text{service supported features}\}$$
$$P = \{p / p \in (x, y) \text{ set of device profile}\} \text{and}$$
$$D = \{d / d \in \text{client's device supported features}\}$$

Table 1: Keyword-based Service Matching Algorithm Orininally from [9]

Input: a keyword (w), device context
Output: list of Web services matching the keyword
Take a keyword input w Ɛ W,
where W is a set of provided web services
Form a set N of the synonyms of the keyword entered
Foreach n_i Ɛ N
  Retrieve n if (name match keyword)
  Exclude stop words
  Store the retrieved web services into set R
End
Foreach r Ɛ R
  Filter our services do not match device context
  Store remaining service into set T
End
//Calculate the semantic distance between "n" and "r"
Foreach n_i Ɛ N
  Foreach r_j Ɛ R
    $d(n_i, r_j) = \sum_{k=1}^{|F|} \left(p(t_{ik} | s_i) * \log \left(\frac{p(t_{ik} | s_i)}{p(t_{ik} | r_j)} \right) \right)$
  End
  Add r_j to list of retrieved web services (L)
  if r_j is closely related to n_i
End

Table 2: Service Relevancy Ranking Algorithm

Input: Set of web services W Ɛ L
  Set of: resource profile:P
  Set of web service supported features: C
  Set of device supported features: D and
  Minimizing factor: M
Output: Resource-aware ranked list of web services (RaRnk)
1 Initialize RaRnK
2 //compute resource and features weight for
3 // each web service using (2) and (3)
4 **Foreach** w_i in W do
5 Call resource weight function (R_wW_i)
7 Call Features weight function (F_wW_i)
8 **End**
9 // Rank web services with priority
10 // to resource profile
11 **Foreach** w_i Ɛ W do
12 Sort (w_i: R_ww_i, F_ww_i) = RanK
13 **End**
14 RaRnK = RaRnK + RanK
15 Return RaRnK

The resource and features weights are generated by their corresponding functions; the resource weight function $f_{rw}(...)$ and the features weight function $f_{fw}(...)$ of (1) and (2) respectively.

Therefore, given a web service wi Ɛ W, the resource and feature weights can be

Journal of Theoretical and Applied Information Technology
15th July 2018. Vol.96. No 13

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

computed based on equations 2 and 3 respectively.

$$R_w w_i = f_{rw}(W, F) \quad (1)$$

$$F_w w_i = f_{fw}(W, C, D, M) = \sum_i w_i \left(f(c_k, d_j) \times M \right) \quad (2)$$

From equation (2) the features weight function $f_{fw}(\ )$ calls an Object Relationship Function, f in equation (4), which computes the relation between two objects using arithmetic operation and the Normalized Google Distance – NGD [9]. The function $f(x, y)$ generates values in the range $k \in [0...1]$. Where $k \geq 0.5$ implies that x and y are related while $k < 0.5$ means they are not related.

$$\int(x, y) = \begin{cases} \overline{y} & \text{if x, y are} \\ sim & \text{if x, y are strings} \end{cases} \quad (3)$$

From (2), we introduced a minimizing factor (M) meant to ensure that the resource weight remains a dominant score in the rating processes. That is, to avoid the chances of having to rank a less resource-efficient web service that supports more device features as being the best at the expense of a more resource-efficient counterpart. To achieve this, "M" is set to a default value of 0.05. The implication of this small value of "M" is that based on equation (2), a less resource-efficient web service must support at least 20 more features for it to be ranked better than a resource-efficient one. However, we assume that no single web service that will support up to 20 device features, which makes the above condition infeasible.

4. AMC RESOURCE-AWARE SERVICE DISCOVERY ARCHITECTURE

The architecture of the proposed resource-aware relevant service discovery mechanism (RaRSDiM) is shown in Figure 2. The architecture consists of two layers:

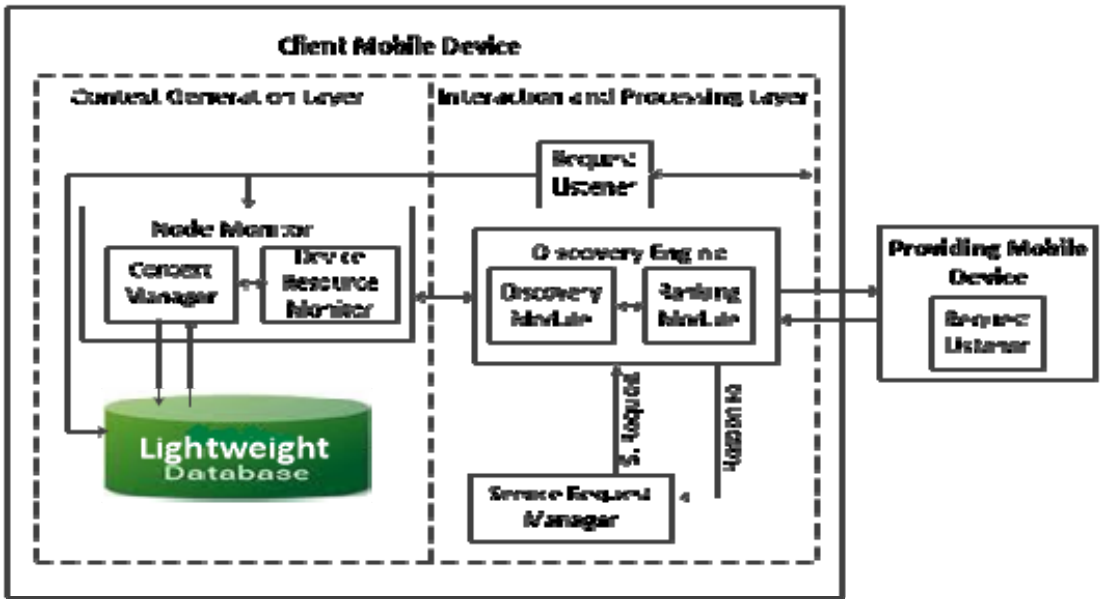

Figure 2: AMC Service Discovery Architecture

the context generation layer (CGL) and the interaction and processing layer (IPL). These two layers consist of three main components with modules that run in them or within the layer. The three main components are the Node monitor, Lightweight database, and the Discovery engine.

4.1 Context Generation and Storage Layer

An important consideration for the proposed mechanism is the utilization of static context (device profile) and dynamic context (resource profile) to enhance service discovery in AMC environment. This goal is achieved in this

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

mechanism via the context generation and storage layer. This layer is made up of components that provide or store context data relevant to the discovery process. There are two components in this layer – node monitor and lightweight database.

4.1.1 The Node Monitor (NoM) Component

Dynamic device context can impede relevant service discovery in AMC domains. This inherent characteristic of mobile environments poses the challenge of discovering relevant services [23]. In addressing this challenge, the node monitor component of the proposed mechanism monitors device context and provides the relevant context data used in the service discovery process. Two modules run in the NoM component – Device Resource Monitor and the Context Manager. The Device Resource Monitor monitors the dynamic context (also referred to as resource profile) of a mobile device.

For example, whenever a service request is launched the device resource monitor collects information about the current battery level and the available memory of the requesting device as discussed in section 3. The context manager on the other hand interacts with the device resource monitor and the lightweight database to make context data available. During this interaction the context manager retrieves dynamic context and static device context from the device resource monitor and lightweight database respectively for use in performing discovery operation.

4.1.2 The Lightweight Database Component

A lightweight database (SQLite) embedded in Android devices can used to host mobile web services as well as store static contexts that may be required for the discovery system. That SQLite has a self-contained library with no server component, no need for administration, a small code footprint, and limited resource requirements makes it highly suitable for resource-constrained environments like AMC.

4.2 Interaction and Processing Layer

This layer consists of the discovery engine and other modules that facilitate both internal and external interactions. Internal interaction happens between a service requester and his mobile device while external interaction refers to the communication between a client device and providing/host device. The first kind of interaction occurs when a mobile user initiates a service request. On the other hand, the operation that involves a client device requesting a service from a host is termed external interaction.

4.2.1 Service Request Manager Module

Service requests are constructed through the request manager and subsequently used to query nearby AMC nodes for required services. Basically, the service request manager performs two main roles: provides user interface that allows users to enter a word or phrase representing the name of desired web service; constructs an XML or JASON message using the word or phrase entered by requesters. The constructed message is then forwarded to the discovery engine for processing by the same module.

4.2.2 The Discovery Engine (DiEn) Component

DiEn is a principal component of the proposed service discovery mechanism. Principally, the proposed service discovery and ranking algorithms are implemented by the DiEn component. To function, DiEn interacts with the node monitor component to obtain context data.

Basically, DiEn performs service discovery and ranking functions through its discovery and ranking modules:

4.2.2.1 Discovery Module (DisM)

Responsible for adapting service requests based on the provided context information to discover relevant services. DisM performs its function by receiving the XML or JASON service request message from the Request Manager and then incorporates device context data extracted from the Node Monitor component. These contexts information now serves as additional request parameters. The processing of expanding the original service request by adding device context is termed service request adaptation. The adapted service request is forwarded to the providing device.

4.2.2.2 Ranking Module (RnkM)

Retrieves services and ranks them according to their relevancy weight as described in section 3. The ranked list of web services is then returned to the service request manager.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

A distinctive feature of RaRSDiM is that at any time a service request is launched, the requesting device's current context is, in addition to the search keyword, used as constraint parameters to filter and rank returned services. Potentially, this feature: enable the discovery process to be proactively adapts to changes in device context, reduce the number of retrieved services, ensure that clients discover services that match their resource capability, and makes invocation of retrieved services easy and user-friendly by returning matched services in a ranked list. These benefits sum up to fulfil two aims: (i) discovery of relevant services and (ii) optimization of device resource usage.

Service request Listener Module: Since AMC peers communicate through a network, the service request listener module listens and responds to incoming service requests from clients. Such interaction which represents sending and receiving service request is mediated by the service request listener module. In addition to intercepting service requests from potential clients, the request listener directly communicates with the lightweight database to access hosted web services or other static context data.

Though each AMC node has a request listener module, the module is only in active mode from the service provider's side. This design is aimed at avoiding redundancy, which can lead to energy waste.

5 EXPERIMENTATION AND EVALUATION

To investigate service discovery based on the proposed approach, a web service description document directory was created using SQLite database, which is embedded in Android platforms. That is, the mobile device acting as service provider uses SQLite database to host web service documents of the services it provides. Using SQLite DB to hold descriptions of sample web services enabled the experimentation of service discovery without having to implement real mobile web services.

These web service description documents were obtained from WebserviceList, WebserviceX and XMethods, which are online web service directories. To effectively test the proposed service discovery approach, these sample service description files were first extended following the WSDL-M standard as discussed earlier. Other notable proposals in the literature adopted the same approach [11], [24]. To perform discovery operations, the proposed Discovery Engine first extracts keywords (representing individual web service names) from the service description files found in the description document directory marked with the tag <wsdl:documentation> and their corresponding hardcoded capabilities.

5.1 Setup of Experiment

The experimental setup consisted of six android mobile devices. These devices included a Hisense HS-U939 smartphone, an HP-Slat 7 HD tablet. The other four were G-TiDE E77, Sony Xperia E1, Infinix X506 and Samsung Galaxy X2 smartphones, all running Android 4.2.2. In each of these devices the RaRSDiM engine prototype was installed, which enables them to discover themselves as peer nodes forming an Ad-hoc Cloud.

In the experimentation, Hisense HS=U939, Infinix X506, Sony Xperia E1, G-TiDE E77 and Samsung Galaxy X2 served as the providing devices. This was done by setting RaRSDiM prototype TO run on provider/host mood. On the other hand, the HP-Slat 7 HD device acted as a client – that is, it client mood was activated on it. The preference for HP-Slat 7 HD device as the client was based on its screen size, which helps to provide a better view of the experimental results.

To achieve inter-node connection and communication, Wi-Fi Direct technology with the capability to create P2P ah-hoc network was utilized. With this technology, a host device automatically functions as a hotspot provider. And by using native Java Sockets (java.net.ServerSocket and java.net.Socket), the host device receives, processes, and sustains incoming connection instances (an instance of java.net.Socket pointing to the url of the server device) initiated by a client device.

The experiments were carried out with 1000 modified (based on the WSDL-M standard) web service description documents obtained from online web service directories as explained in section 5.

For easy deployment of web service description (WSDL) documents into the host device, we created a web interface (with Java and QSL and hosted on a free server). The web

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

interface utilizes the WebSocket technology (part of Java EE 7), which defines an API for establishing socket connections between a web browser and a server and creates a persistent connection for bi-directional and real-time client/server communications. With this interface, we were able to remotely push WSDL files to the host device. To receive pushed services, the host device requires Internet connection to be able to perform an uncomplicated process of registration that enlists it as a web service subscriber with the web interface.

5.2 Performance Evaluation

The proposed mechanism is evaluated based on recall and precision rates, relative quality of relevant service discovery (RQoRSD), processing time and resource optimization as depicted in Figures-3 to 7. The main essence was to ascertain the impact of using device context information in driving resource usage optimization in the discovery of relevant services in AMC environment.

These evaluation parameters are defined in [24], [27] as:

$$\Pr e = \frac{\text{Re}\,turned\ \ \text{Re}\,levantServ\ s}{\text{Re}\,turnedServ\ s} \quad (4)$$

$$RRate\ = \frac{\text{Re}\,turned\ \ \text{Re}\,levantServ\ s}{Published\ \ \ \text{Re}\,levantServ\ s} \quad (5)$$

$$RQoRSD = \frac{FilteredOutServs}{\text{Re}\,turned\ \text{Re}\,levantServs} * 100 \quad (6)$$

5.2.1 Precision and Recall Rates Analysis

A service in this experiment is either relevant or not relevant depending on its resource requirement and supported device features. A total of one thousand web service description documents (representing 1000 web services) were used in this experiment: the number of published relevant services was kept constant while varying the number of published services by 200 services. Published relevant service and published services were represented by their respective service descriptions in the web service description document directory.

We varied the number of published services to influence the independent variables (retrieved services and retrieved relevant services). Five service requests were performed under two scenarios: 1) with the context module disabled and 2) with the context module enabled. The aim was to study the RaRSDiM's performance with regards to recall and precision. Data obtained in the recall and precision experiments are shown in Tables 3 and 4 respectively.

Results as presented in Figure 3 indicate that RaRSDiM performs at peak recall rates of 39% and 72% for operation with and without device context respectively. Higher recall rate here explains the fact that without device context, more similar services are retrieved because the constraint condition is whether a service matches the keyword or not. However, when the same experiment was conducted while using device context information as additional constraint conditions for service retrieval, result obtained indicated the opposite. That is, fewer services are retrieved, which is why we have lower recall rate. Retrieving fewer services is advantageous to AMC scenario due to the limited resource capability of mobile devices.

Table 3: Recall Rate Outcome.

Published services	Recall without device context	Recall with device context
0	0	0
200	0.39	0.19
400	0.51	0.21
600	0.57	0.29
800	0.59	0.29
1000	0.72	0.35

Table 4: Precision Rate Outcome

Published services	Precision without device context	Precision with device context
0	0	0
200	0.25	0.56
400	0.21	0.52
600	0.18	0.53

On the other hand, as depicted in Figure 4, when RaRSDiM is evaluated with regards to precision rate, its performance revealed a reverse trend as against that obtained when evaluated based on recall rate. That is, RaRSDiM recorded higher precision rate of 56% when device context is utilized as against 25% without device context. This result is indicative of the fact that the service

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

request adaptation technique adopted in the proposed discovery mechanism helps to narrow down the number of retrieved services to only those that meet the client device's current context, thereby increasing precision rate.

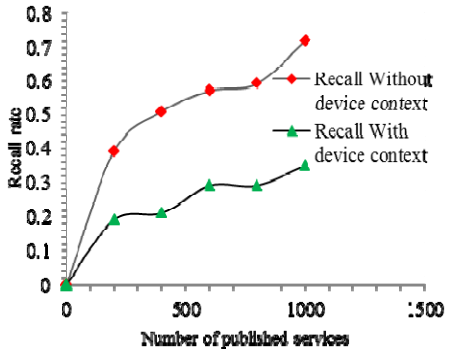

Figure 3: Recall Rate

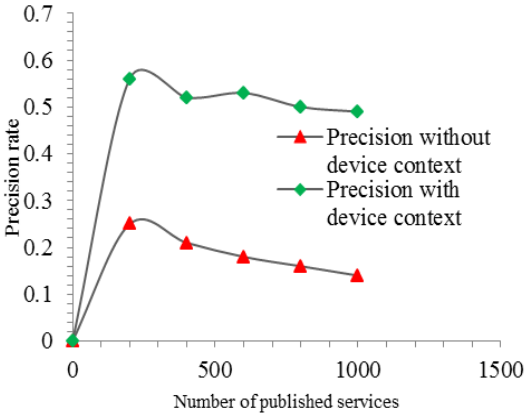

Figure 4: Precision Rate

It is understandable that having high precision and low recall has the inevitable implication of possibly leaving out some relevant services. However, it can be argued that although some relevant services may be left out, they may not necessarily be relevant in terms of meeting the current resource requirement of the client device – which is the idea behind the filtering approach advocated in this paper. Therefore, the proposed solution approach can deliver an effective service discovery system for resource-constrained environments.

5.2.2 Relative Quality of Relevant Service Discovery

As defined in equation (6), RQoRSD is a vital parameter used to measure the impact of device context information on the quality of service discovery. In Figure 5, results obtained from RaRSDiM representing the RQoRSD for six different service requests are shown.

On the average, RaRSDiM demonstrates a 73% improvement in relevant service discovery. This performance enhancement is contributed by the proposed context-aware service discovery strategy, which reduces the number of retrieved relevant services. This reduction invariably increases the number of filtered-out services. Consequently, the precision rate is positively impacted, resulting in improved quality of service discovery.

In the same experiment, it was observed that there was irregularity in RQoRSD across the six service requests. This lack of a defined pattern in the outcome of RQoRSD can be attributed to the dynamic nature of AMC environment, which makes changes in device contexts not to occur in any distinct pattern. This result is achieved through the Node monitor feature, which extracts the current context of the client device whenever a service request is initiated.

5.2.3 Processing Time Evaluation

As much as the use of context information can help improve service discovery, especially in AMC domain, context overhead may impact on processing time and processing time has a phenomenal implication for computing resources (battery, memory). This consideration makes it necessary to evaluate CaRSDiM with regards to processing time.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

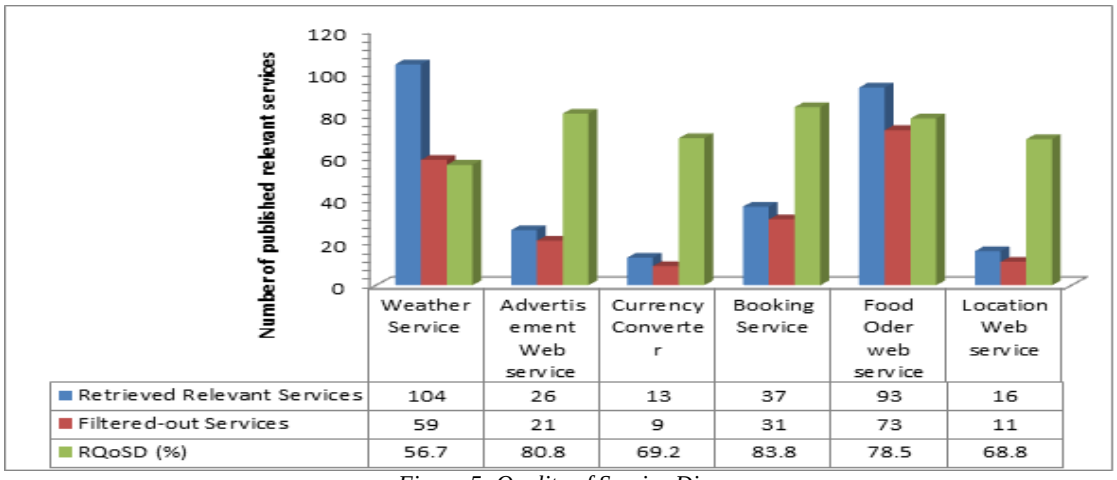

Figure 5: Quality of Service Discover

In this experiment, first, we launch service request via RaRSDiM request manager interface and recorded the number of relevant services retrieved and the overall response time, which accounts for communication, service matching, and context matching time. Second, the same service request was re-launched, but in two steps: a) searching and retrieval and b) filtering and ranking. The first operation in the second stage involves initiating a service request with RaRSDim context module disabled. With this operation, web services matching the entered keyword are retrieved and the time taken to complete the processes (service matching time without context) is reported. The second operation then utilises device context to filter the services retrieved in the first stage. In this step, the experiment is only concerned with determining the context matching time.

Figure 6 is based on Table 5 and demonstrates the effect of device context on processing time. From the result, RaRSDiM attained a shorter maximum processing time of 94ms in all three services requests executed with device context. While without device context, processing time increased by a maximum of 213ms.

Table 5: Processing Time Outcome

Matching time (ms)	Number of retrieved services						
	0	26	68	97	120	197	234
	0	104	126	159	183	270	307
Non-context-based	0	10	17	28	33	51	94
Context-based	0	26	68	97	120	197	234

Basically, a short processing time was because retrieved services were tailored to current context of the target device. Therefore, there was a reduction in the total number of returned services. The benefit of this outcome is that having short processing time implies that RaRSDiM minimizes resource consumption since long processing time has adverse effect on both battery and memory.

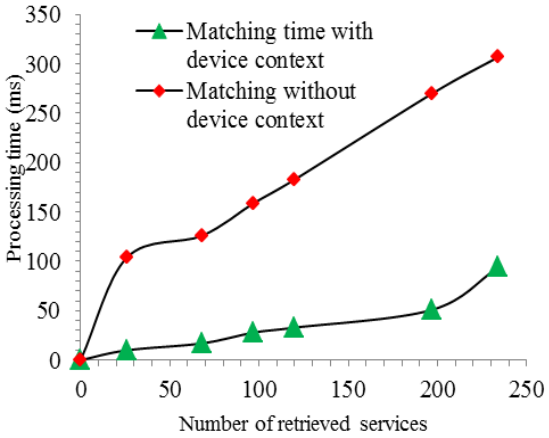

Figure 6: Processing Time

5.2.4 Resource Optimization Evaluation

Due to resource scarcity in AMC environment, one key objective of this paper is to optimize resource utilization, to enhance effective service discovery. To address this challenge, RaRSDiM adopts an adaptive approach that first incorporates the current resource context of client devices into service requests before discovering services.

In this experiment, device resource context was categorized into two states: critical and moderate. Critical state is when a client device is low in both battery and memory.

Journal of Theoretical and Applied Information Technology
15th July 2018. Vol.96. No 13

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

In the other hand, when a client device is either low in memory or battery but not in both, it is said to be at normal device resource context state. The moderate resource state was further divided into two sub categories – moderate 1 and moderate 2 as exemplified in Table 6. For each service request executed, device resource context is varied from moderate to critical. The purpose of this alteration is to evaluate the effect of resource-aware service discovery on resource usage optimization. With this evaluation, we attempt to illustrate how RaRSDiM adapts to changes in device resource context (battery, memory) during service discovery.

Figure 7 is derived from Table 6 and depicts RaRSDiM's performance in this regard.

Table 6: Adaptive service discovery

Service requests	Resource context	Percentage of retrieved services (%)		
		Battery and memory efficient services	Only battery efficient services	Only memory efficient services
Request 1	Critical (low in battery and memory)	61.5	71.4	68.0
Request 2	Moderate 1 (low in battery)	73.2	75.6	65.2
Request 3	Moderate 2 (low in memory)	54.4	78.6	62.4

From Figure 7, it was observed that although the number of published relevant services was kept constant at 30 services for the three executions of the same service request, the number of retrieved relevant services varied in all three scenarios (Critical, Moderate 1 and Moderate 2). For instance, at critical resource state, fourteen (14) relevant services were returned, which is also the smallest number of services retrieved in all the three scenarios considered. These number of services amounts to 63% cumulative relative quality of relevant service discovery.

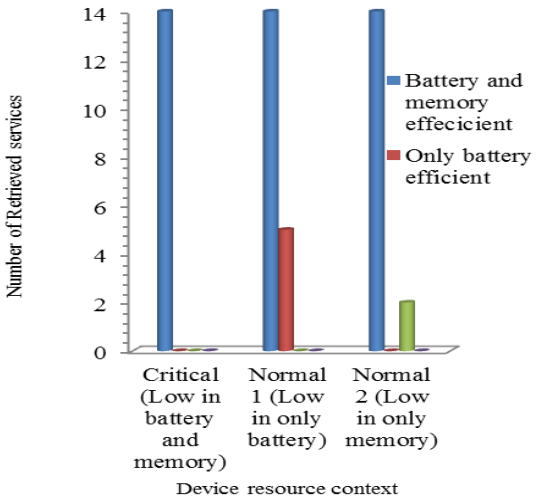

Figure 7: Resource Optimization

Furthermore, a manual inspection of the retrieved services and their resource requirements shows that only optimally efficient services (efficient in battery and memory) were returned at critical resource state. This outcome coupled with the least number of services retrieved at critical resource level implies that RaRSDiM can adapt to changes in device resources in a manner that promotes efficient resource usage.

In the same vein, considering the Moderate 1 and Moderate 2 scenarios, the same service request generated sixteen (16) and nineteen (19) relevant services respectively. This result indicates that both scenarios returned higher number of services each compared to that returned at critical resource state. Following a manual check, on all the service returned at the moderate resource states, it was revealed that only services that are most efficient in the depleted resource or both were retrieved in each case. The implication is that when a device requesting a service is deficient or low in a resource, say battery, the discovery mechanism only returns battery-efficient or non-power-hungry services. The same thing happens when the client device is low in memory. However, if the client device's resources are in critical state, that is both battery and memory are low, RaRSDiM returns only services that are both battery and memory efficient. These results were achieved using a combination of the node monitor and the discovery engine to monitor device resource and adapt service requests to suit a client device's current context.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

6 CONCLUSION

To take advantage of changing context to determine relevant services, a prototyped discovery mechanism was designed in this paper. The mechanism leverages on mobile devices' resource capabilities and web service functionalities to provide proactive service discovery in AMC.

Based on our findings, we can draw the following conclusions: (i) In AMC environment, the issues of service relevance and resource capability of client devices are interwoven because context consists of device resources. (ii) Incorporating resource context information into AMC service discovery process would introduce such flexibility that can equip discovery mechanisms with adaptive capability to proactively respond to changing context while discovering services. (iii) With regards to resource-efficiency, optimal resource utilization is critical to achieving service discovery efficiency in AMC domains. More so, that although the use of context information can enhance discovery of relevant services in AMC environment, the unpredictable nature of the environment due to dynamic context presents an inherent resource challenge.

Addressing this fundamental challenge, requires a more comprehensive context model that considers the state of device resources in addition to other context variables. More so, the design of service discovery frameworks with the capacity to adapt to context change also becomes imperative. Considering a resource-aware service discovery mechanism that achieved adaptive and resource-efficient discovery of relevant services was presented. The mechanism exploits the relationship between service-relevance and resource capability of devices. With this approach, it was demonstrated that service discovery mechanisms with resource-awareness and adaptive capabilities can enhance resource-usage optimization and relevant service discovery in AMC environment.

Experimental results indicated that the proposed approach improves discovery efficiency with regards to discovering relevant services and minimizing resource consumption.

REFERENCES:

[1] D. Renzel, D. Kovachev, and R. Klamma, "Mobile Community Cloud Computing: Emerges and Evolves," *2010 Elev. Int. Conf. Mob. Data Manag.*, pp. 393–395, 2010.

[2] A. Khalifa, M. Azab, and M. Eltoweissy, "Resilient Hybrid Mobile Ad-hoc Cloud Over Collaborating Heterogeneous Nodes," *Collab. 2014 Proc. 10th IEEE Int. Conf. Collab. Comput. Networking, Appl. Work.*, 2014.

[3] R. Lacuesta, J. Lloret, S. Sendra, and L. Peñalver, "Spontaneous Ad Hoc Mobile Cloud Computing Network," *Sci. World J.*, vol. 19, 2014.

[4] H. Dinh, C. Lee, and D. Niyato, "A survey of mobile cloud computing: Architecture, Applications, and Approaches," *Wirel. Commun. Mob. Comput.*, vol. 13, no. 18, pp. 1587–1611, Dec. 2013.

[5] S. Abolfazli, Z. Sanaei, E. Ahmed, A. Gani, and R. Buyya, "Cloud-Based Augmentation for Mobile Devices: Motivation, Taxonomies, and Open Challenges," *IEEE Commun. Surv. Tutorials*, vol. 16, no. 1, pp. 337–368, 2014.

[6] E. E. Marinelli, "Hyrax : Cloud Computing on Mobile Devices using MapReduce," *M.Sc. thesis, Sch. Comput. Sci. Carnegie Mellon Univ. Pittsburgh*, vol. 389, no. September, 2009.

[7] D. Afuro, P. Mudali, M. O. Adigun, A. Akingbesote, and M. B. Mutanga, "Ad-hoc Mobile Cloud Service Discovery Based on Device Resources," *South. Africa Telecommun. Networks Appl. Conf. 2015*, pp. 273–278, 2015.

[8] K. Dejan, Y. Cao, and R. Klamma, "Mobile cloud computing: A comparison of application models," *arXiv Prepr. arXiv1107.4940*, 2011.

[9] Z. Zhao, X. Huang, and N. Crespi, "A system for web widget discovery using semantic distance between user intent and social tags," *Soc. Informatics 4th Int. Conf. SocInfo2012*, pp. 1–14, 2012.

[10] K. Elgazzar, H. Hassanein, and P. Martin, "Mobile Web Services : State of the Art and Challenges," *Int. J. Adv. Comput. Sci. Appl.*, vol. 5, no. 3, 2014.

[11] E. Al-Masri and Q. H. Mahmoud, "MobiEureka: An approach for enhancing the discovery of mobile web services," *Pers. Ubiquitous Comput.*, vol. 14, no. 7, pp. 609–620, 2010.

[12] D. A. Egbe, M. B. Mutanga, and M. O. Adigun, "SERVICE DISCOVERY IN AD-HOC MOBILE CLOUD : CONTEMPORARY APPROACHES AND FUTURE DIRECTION," *J. Theor. Appl. Inf. Technol.*, vol. 90, no. 1, pp. 101–117, 2016.

[13] L. A. Steller, "Light-weight and Adaptive

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

reasoning for mobile web services, Ph.D thesis, Monash University, Australia." 2010.

[14] G. O. Cortazar, J. J. S. Zapater, and F. G. Sanchez, "Adding semantics to cloud computing to enhance service discovery and access." pp. 1–6, 2012.

[15] M. Rodríguez-García, "Creating a semantically-enhanced cloud services environment through ontology evolution," *Futur. Gener. Comput. Syst.*, no. 32, pp. 295–306, 2014.

[16] A. V. Paliwal, B. Shafiq, and J. Vaidya, "Semantics-Based Automated Service Discovery," *IEEE Trans. Serv. Comput.*, vol. 5, no. 2, pp. 260–275, Apr. 2012.

[17] N. A. Saadon and R. Mohamad, "Cloud-based Mobile Web Service Discovery framework with semantic matchmaking approach," in *2014 8th. Malaysian Software Engineering Conference (MySEC)*, 2014, pp. 113–118.

[18] D. Bianchini, V. De Antonellis, and M. Melchiori, "Lightweight Ontology-Based Service Discovery in Mobile Environments," in *17th Int. Conf. on Database and Expert Systems Applications*, 2007, pp. 359–364.

[19] K. Elgazzar, H. Hassanein, and P. Martin, "DaaS: Cloud-based Mobile Web service Discovery," *Pervasive Mob. Comput.*, vol. 13, pp. 67–84, 2014.

[20] A. Ravi and S. K. Peddoju, "Energy Efficient Seamless Service Provisioning in Mobile Cloud Computing," *2013 IEEE Seventh Int. Symp. Serv. Syst. Eng.*, pp. 463–471, Mar. 2013.

[21] B. Zhou, A. V. Dastjerdi, R. N. Calheiros, S. N. Srirama, and R. Buyya, "A Context Sensitive Offloading Scheme for Mobile Cloud Computing Service," 2015.

[22] K. Elgazzar, H. Hassanein, and P. Martin, "Effective Web service discovery in mobile environments," in *2011 IEEE 36th Conference on Local Computer Networks*, 2011, pp. 697–705.

[23] M. Miraoui, C. Tadj, and J. Fattahi, "Dynamic Context-Aware and Limited Resources-Aware Service Adaptation for Pervasive Computing," *Adv. Softw. Eng.*, pp. 1–11, 2011.

[24] K. Elgazzar, "Discovery , Personalization and Resource Provisioning of Mobile Services, Ph.D Thesis," *Queen's Univ. Kingston, Ontario, Canada*, no. August, 2013.

[25] L. Sun, H. Dong, and J. Ashraf, "Survey of Service Description Languages and Their Issues in Cloud Computing," in *2012 Eighth Int. Conf. on Semantics, Knowledge and Grids*, 2012, pp. 128–135.

[26] M. Ruta, F. Scioscia, and E. Di Sciascio, "A Mobile Matchmaker for Resource Discovery in the Ubiquitous Semantic Web," in *2015 IEEE Int. Conf. on Mobile Services*, 2015, pp. 336–343.

[27] A. Gunawardana and G. Shani, "A Survey of Accuracy Evaluation Metrics of Recommendation Tasks," J. Mach. Learn. Res., vol. 10, no. Dec, pp. 2935–2962, 2009.