International Journal of u- and e- Service, Science and Technology
Vol.10, No. 1 (2017), pp.299-316
http://dx.doi.org/10.14257/ijunesst.2017.10.1.26

A Device Resource-aware Service Discovery Solution for Ad-hoc Mobile Cloud

Dominic Afuro Egbe[1*], Bethel Mutanga Murimo[2] and Matthew O. Adigun[3]

[1,3]*Department of Computer Science, University of Zululand, KwaDlangezwa, South Africa*
[2]*Department of Information and Communication Technology, Mangosuthu University of Technology, Durban, South Africa*
[1]*domafuro@gmail.co*, [2]*mutangamb@mut.ac.za*, [3]*profmatthewo@gmail.com*

Abstract

Ad-hoc Mobile Cloud (AMC) computing represents an opportunistic platform for inexpensive and collaborative resource provisioning. However, service discovery in AMC faces resource limitation and dynamic context challenges. Current attempts to address the impact of context change on service discovery have either not investigated device context or only considered device profile. Meanwhile, service discovery in AMC is resource-sensitive and the dynamic nature of this environment implies the relevance of discovered services is context-dependent. Discovering relevant services therefore requires enough context information, especially resource context, which specifies a device's resource capability. This capability enhances discovery efficiency because having sufficient context information promotes adaptability, which enables discovery mechanisms to tailor services. This paper presented a service discovery mechanism that polled and utilized device context to discover services in a resource-aware manner. We developed a prototype with Node Monitor and Discovery Engine components to gather device context and perform proactive service discovery respectively. With these components, we achieved a service discovery process that utilized device context to measure the resource capability of a client device during service discovery. The results indicated that this approach optimized device resource usage and improved quality of service discovery.

Keywords: *Ad-hoc mobile cloud, resource-aware, device context, service relevance*

1. Introduction

Mobile Cloud Computing has assumed prominence in recent years owing to the proliferation of advanced portable devices. This trend coupled with the increasing drive for ubiquitous computing, has inspired Ad-hoc Mobile Cloud (AMC) computing. The emergence of AMC has also received significant boost from the surge in number of mobile devices with fascinating and advanced features [1-2]. For instance, advances in mobile devices' features with regards to hardware and functionality makes it possible for such devices to take advantage of innovations offered by wireless network technologies to support peer-to-peer resource sharing [3-5].

Another interesting trend is the rapid expansion of e-markets and the corresponding growth in the mobile services consumer base [6-7]. This duo has led to increased research interest aimed at standardizing mobile web services. The advent of mobile web services brings about the prospect of web services being offered from mobile devices, accessed and invoked by peer nodes.

The above highlighted trends richly support the AMC paradigm, which is envisioned to serve as a low-cost and opportunistic mobile web service provisioning platform,

*Corresponding Author

ISSN: 2005-4246 IJUNESST

complementary to the Mobile Cloud. In response to the envisioned prospect of AMC, recent years have witnessed research attention geared towards standardizing mobile web service provisioning with respect to developing mobile hosting platforms and formulating suitable service discovery mechanisms [2-10].

Nonetheless, service discovery in AMC faces many difficult challenges, the major of which are resource limitation and dynamic context. Fundamentally, these challenges are intrinsic characteristics of mobile environments, which can significantly impact on the service discovery process. Consequently, researches in AMC service discovery are concerned with addressing the challenges of adapting to context change and optimizing resource usage [6-12]. Solution to these challenges is critical to achieving efficiency and effectiveness of service discovery, which are measured by determining service relevance and conformance to the resource requirements of mobile devices that constitute AMC environment.

Generally, service discovery mechanisms are required to return relevant services according to users' requirements. That is, retrieving services that fulfill the desired functionality. Nevertheless, in AMC, resources are limited and at the same time, dynamic. In particular, the dynamic nature of AMC resources means that the state of these scarce resources can change unanticipated during discovery operation. Consequently, in AMC domain, fulfilling user's requirements alone is insufficient to guarantee discovering relevant services. This is because the device's context plays a critical role in determining the relevance of discovered services.

The focus of this paper therefore, is to investigate the use of device context to achieve resource-aware-based discovery of relevant services in AMC. When there is a change of state in a device's battery and or memory, the device's context is said to have changed. For instance, if a device's battery is in state 'x' when it first launches a service request and after a period, it executes a similar operation and the battery level drops to 'z', where z = x-k, then there is a change in the device's context - battery. Such dynamic context change has varied implications for service discovery. First, it impedes the discovery of relevant services because discovered services might no longer match the capabilities of the client device in its current context, resulting in resource wastage and low client satisfaction. Second, since mobile peers also acts as providers of service in AMC, the disappearance of a participating node can affect the process of discovering a service. For instance, if a device runs out of limited resources and becomes inaccessible while providing a service, it does so alongside with the service it provide, which suddenly becomes unavailable for discovery. It is therefore highly desirable that service discovery in AMC employ proactive techniques that incorporates resource usage intelligence. Such requirement makes it possible to provide a discovery process capable of adapting to changes in device context and aimed at optimizing resource utilization.

However, current approaches have not fully explored the use of device context to discover services. Although user context: user profile and preferences, and device profile (which consist of device capabilities) have been utilized by current solutions, the resource profile component of device context has not been explored. In this paper, we consider resource profile to consist of active and dynamic device context such as battery and memory. This consideration is imperative because any change in these device resources could lead to discovering a service that may be relevant with respect to functionality but yet not usable due to resource limitations on the consumer device. We therefore argue that it is not enough to consider the relevance of a web service based on whether the service meets user's required functionality while neglecting whether the service matches the client's device capability.

The main contribution of this paper is the development of a service discovery mechanism that takes into account the resource state and unique features of the client device. To realize this goal, we proposed a service discovery algorithm that encompassed a ranking algorithm. The ranking algorithm employed the technique of 'relevancy score'

to sort retrieved services. In other words, each discovered service was assigned a score, which was a weight representing the resource requirement of the service and the total number of supported device features.

Furthermore, the mechanism adapted service request to context change. This service request adaptation was realized by first monitoring changes in the context of device resources and incorporating the context information into service requests as filtering conditions. This approach entailed utilizing a combination of device context information to facilitate discovery of relevant services in a manner that took cognizance of resource limitation in Ad-hoc Mobile Cloud.

In this paper therefore, we proposed a resource-aware service discovery solution that employed the device context model illustrated in Figure 1. The proposed device context model consisted of: resource profile (battery and memory) and device profile (supported device features). The goal was to ensure that discovered services were ranked based on the current resource context and other unique features of the client device.

The rest of this paper is organized thus: section 2 presents related work and the proposed context model is described in section 3. We formulated a resource-aware service discovery and ranking algorithm in section 4 followed by the experimentation and result evaluation in section 5. The paper is concluded in section 6.

2. Related Work

In order to understand the state-of-the-art in AMC service discovery, this section explores current literature in the domain. Ideally, existing mobile cloud service discovery approaches should be reused in AMC since the fundamental techniques of service discovery are the same. Nonetheless, research consensus shows that such techniques are essentially designed for wired environments therefore, may be too resource-intensive for mobile devices [13-14]. Therefore, our focus will be on works specifically designed to address AMC service discovery challenges.

In the context of AMC, literature surveyed in service discovery approaches can generally be classified into three categories covered in sections 2.1 to 2.3:

2.1. The Light-Eeight Technology Approach

Research works in this category focused on the use of light-weight semantic techniques to enhance service discovery. While several semantic technologies and approaches for web service discovery are considered too resource-intensive for mobile web services, a number of optimized semantic technologies have been developed over the years to cater for the unique limitations of mobile devices. Some of the scholarly proposals that seek to exploit the efficiency and accuracy introduced by the use of semantics into relevant service discovery in mobile environments include the ideas reported in [15-19].

Generally, semantic approaches rely on ontology to describe web services. With the help of ontology - a formal explicit specification of shared conceptualization [20], semantic-oriented languages such as Web Ontology Language (OWL), Web Ontology Language for Services (OWL-S), Semantic Web Services Ontology (SWSO), Web Service Modeling Ontology (WSMO), and Semantic Annotation for Web Service Description Language (SAWSDL) *etc.*, can be used to create service abstractions that are meaningful to discovery agents or mechanisms. A major essence of semantic description is to enrich service description through enhanced expressivity. Such enhancements can help to support automation in service discovery and improve the efficiency of discovering relevant web services. Also, by semantically describing a service, it becomes possible for the service to be linked to abstract and cross-platform concepts. This sematic-enabled linking facilitates the possibility of resolving and matching different semantic representations and the mapping between services from different providers, thereby

supporting large-scale service provisioning obtainable in a typical Cloud environment [21].

Although literature generally reports that optimizing service discovery mechanisms via semantic techniques could have rich benefits for mobile environments, it can also be strongly argued that such optimization does not completely eliminate the resource burden posed on resource-limited devices. Such resource burden is created because processing semantically represented knowledge requires significant computational resources [13-22]. More so, it is expected that in an Ad-hoc Mobile Cloud scenario, which is basically a small computing community that is made up of resource-constrained devices, the number of offered web services would be relatively moderate since such devices have limited hosting capacity. Therefore, the use of high-level (semantic) matchmakers is generally not a requirement.

2.2. Offloading of Computational-intensive Operations to the Cloud

Service discovery in mobile environments faces the challenge of resource constraint among other challenges [22]. This challenge is principally due to the inherent nature of mobile environments in general. To bridge the gap between resource-constrained environments and resource-intensive web service discovery operations, researchers have explored the option of pushing resource-intensive tasks of service discovery to the Cloud [7-11] and [22-23]. This complementary approach aims at developing service discovery mechanisms that can achieve relevant service discovery while minimizing the resource burden placed on resource-constrained devices.

Whereas the challenges of resource-intensiveness and context-awareness are adequately addressed by various authors through the above approach, the feasibility of achieving such a Cloud supported approach may not be guaranteed in the AMC scenario where weakness or absence of Internet connection may cause inaccessibility of the Cloud.

2.3. The Ad-hoc Approach

The third category of research interest in mobile web service discovery advocates Ad-hoc or decentralized paradigm. The implication of such approach is that the web service discovery operation takes place between mobile peers. Although the Ad-hoc approach to service discovery is an emerging field and its literature scanty, some early works in this domain indicate that the idea is not new [24]. This paper explores the Ad-hoc approach to service provisioning because it entirely supports the AMC paradigm and the proposed service discovery process. Also, since mobile devices are limited in resources, the ad-hoc approach advocates the use of non-semantic techniques, which are considered too computationally complex and resource-intensive for mobile devices [13-26].

Towards achieving the overall goal of AMC, several scholarly works have pioneered the way, especially in designing service discovery mechanisms. For example, [10] explored the standardization of mobile web service discovery. In the work, the authors proposed a three-layer service discovery framework that utilizes such context information as network characteristics, user preferences, device profile, and available resources. However, being a generic framework, no specific solutions were offered by means of experiments.

Another challenge identified to be crucial in AMC is how to discover services that match the capabilities of client devices. In attempt to address this challenge, a dynamic mobile service discovery solution “MobiEureka” was developed in [5]. The solution adopts a device-aware service discovery approach that utilizes device capabilities captured in the web service descriptions. The work aimed at using these context parameters to help discover, filter and rank relevant services. By this approach, the authors achieved expanded web service descriptions by proposing an extension to WSDL standard called WSDL-M.

Furthermore, in [27] the authors implemented a mechanism for personalized mobile service discovery. In recognition of the impact of context in service discovery, various contexts information such as user preferences, device profile, and environmental parameters were used to realize the proposed mechanism.

Besides the above works, the challenge posed by unanticipated context change to AMC service discovery has motivated other researchers to seek more proactive solutions. This is evident in the contribution of the work reported in [28]. In the work, the authors discussed the development of capacity-driven web services. The idea is to develop web services with proactive capabilities that enable them to adapt to context change.

Interestingly, all the works explored under the Ad-hoc category show a convergence of interest with regards to the use of device context information to enhance service discovery. Nevertheless, our work is motivated by the finding that resource profile, which helps to promote resource-awareness in service discovery has not been explored.

3. AMC Device Context Model

For context information to be utilized, it must first be modeled or represented. Modeling context puts it in a form that can be processed. Thus, since the rationale behind the proposed resource-aware service discovery mechanism is to adapt service requests to current context before discovering services, it becomes imperative to formulate a suitable context model for this purpose. Also, to support such a mechanism, two requirements must be fulfilled. (i) First, the discovery mechanism must be able to enhance the utilization of device context information. In this respect, a Node Monitor component should be used to just poll the client device to get the level of the device resources. (ii) Second, web services must be described in a way that allow the capturing of more non-functional parameters like battery and memory requirements of the web service.

To achieve the above goal, we formulated an expanded device context model as a way of supporting a highly expressive service description approach aimed at engendering resource-friendliness and improving the relevance of services discovered in AMC. The proposed device context model consisted of two components namely, device profile (specific supported device features) and resource profile (state of battery and memory) as depicted in Figure 1. The model captured how device context could be utilized to achieve resource-aware service discovery, suitable for the dynamic nature of AMC environment.

As depicted in Figure 1, a mobile device was associated with a mobile profile (supported features or capabilities) and a resource context (the state of the device's battery and memory at any point in time). There are different standards for representing device capabilities. Example are, Composite Capabilities/Preference Profiles (CC/PP) as defined by the World Wide Web Consortium (W3C), User Agent Profile (UAProf), and the Microsoft-based Mobile Device Profile (MDP). All these standards represent the same specific device features in different ways.

Using this context model will make it possible for service providers to enrich or expand their service descriptions with more web service capabilities or context information. For instance, our approach required that each web service was described in a manner that enabled service providers to indicate the supported device features and resource usage requirement of an offered web service. Therefore, for a web service to be discovered, web service capabilities were matched with device context during matchmaking.

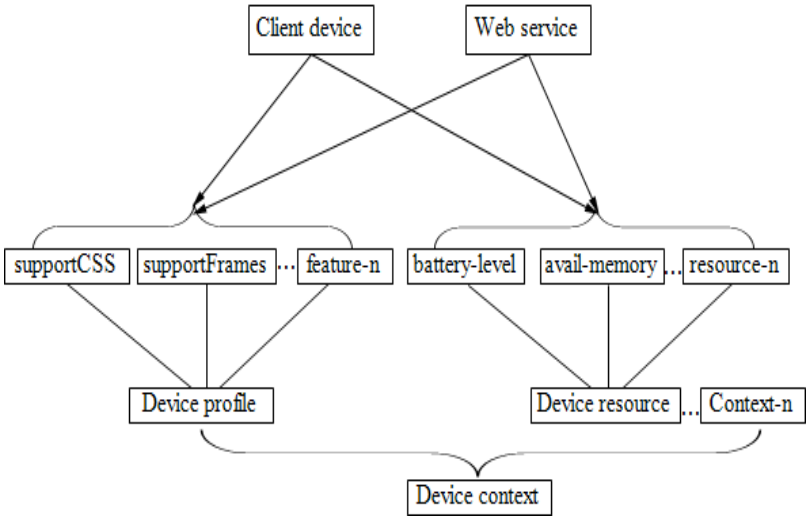

Figure 1. Proposed Device Context Model

In order to utilize the proposed context model, we adapted a WSDL-M based resource-friendly web service description approach from the work reported in [5]. Based on this, we achieved the desired resource-aware web service description approach that incorporated more non-functional or context parameters. Symbolically, in the service descriptions, we used zeros (0) and ones (1) to indicate the resource requirement of a service. Where "0" meant low resource requirement or resource efficient and "1" indicated otherwise. For instance, a hypothetical web service "Ws_1" that was battery and memory efficient was represented as Ws_1: {0, 0}, while another web service "Ws_2", which was battery efficient but memory-inefficient was denoted as Ws_2:{0, 1} *etc.*

4. Resource-aware Service Discovery and Ranking Algorithm

In the web service model, service requests are processed through a matchmaking process (algorithm) that determines which service or services to return to the requester. A keyword-based service matching algorithm is chosen for this work. This choice is intended to minimize resource consumption as against the semantic counterpart. Also, device context is used in the discovery process as a vehicle to drive proactive capability of the discovery mechanism. This is because with device context information, the discovery mechanism is able to adapt service requests in response to context change. Adapting service requests based on context helps to constrain retrieved services to only relevant ones. Consequently, the number of returned services is reduced. The implication of this reduction in number of retrieved services is that with a negligible time complexity that may be introduced into the algorithm as assumed in this paper, processing time or resource burden can also be reduced.

The service discovery and ranking procedure implemented in this paper is formulated based on devices context. The idea is to discover services that match the client device's current context and then rank retrieved services according to the order of their relevance. This discovery and ranking system makes use of two relevancy weighting parameters and their corresponding functions: (i) resource weight (R_w), the value of which depends on resource profile (battery level, available memory) and is generated by the resource weight function: $f_{rw}(\ldots)$ as given in equation (1) and (ii) features weight (F_w), which is the total number of supported device features. This weight is computed by the features weight function: $f_{fw}(\ldots)$ as given in equation (2).

4.1. Algorithm Formulation

Let there be a:

- Set of web services $W = \{w_1, w_2, w_3, \ldots, w_i\}$;
- Set of web service supported features $C = \{c_1, c_2, c_3, \ldots, c_k\}$;
- Set of device profile $P = \{p_1, p_2, p_3, \ldots, p_n\}$ and
- Set of client's device supported features $D = \{d_1, d_2, d_3, \ldots, d_j\}$

Then, utilizing the model of Figure 1, each retrieved web service w_i, can be represented as a matrix [M] = m_{wj}: where "w" (columns) represent individual web services and "j" (rows) is the set of corresponding device features. Such matrix can also be generated for other context parameters. But for the purpose of this work, our resource profile is limited to only battery and memory.

Based on the above context parameters, the resource weight, features weight and the relevancy score are then computed according to (1), (2) and (3) respectively:

$$R_w w_i = f_{rw}(W, P) = \sum_{i=1}^{n} p_i \tag{1}$$

$$F_w w_i = (W, C, D, M) = Mf * \sum_{i=1}^{n} f(c_i, d_i) \tag{2}$$

$$\mathrm{Re}lScore_{wi} = f_{rs}(W) = (R_w w_i + F_w w_i) \tag{3}$$

From (1), the resource weight function assigns weights to web services according to the resource requirements of the service as discussed in section 3. A web service earns a unit score of "1" if it is efficient in any resource. For example, an arbitrary web service 'Ws' described as "Ws":{0, 0} will earn a resource weight of 2 units. In the same vein, the features weight function utilizes the Object Relationship Function, *f* to determine the similarity between web services supported features and device supported features as shown in (2). The function computes objects relation using an arithmetic operation and the Normalized Google Distance – NGD [29]. For instance, the function $f(c_i, d_i)$ generates values in the range k Ɛ [0...1]. If k >= 0.5 then c and d are related while values of k < 0.5 means there are not related.

We reasoned that it is possible for a less resource efficient service with more supported features to be ranked as the best service at the expense of a more resource-efficient service that supports fewer device features. To avoid such flaws, a numeric constant value called minimizing factor "Mf" is introduced in equation (2) to control the features weight value. This constant value is meant to ensure that the resource weight remains a dominant score in the rating scale. For instance, in our experiment, Mf was assigned a default value of 0.05. With such a small value, a less resource efficient web service must support at least 20 more features for it to be ranked better than a resource efficient one, which is infeasible. Following the formulations in (1)-(3), Table 1, shows the proposed resource-aware service discovery and ranking algorithm.

Table 1. Web Service Discovery and Ranking Algorithm

```
Input: Set of web services W; Set of: resource profiles: P;
  Set of web service supported features: C;
  Set of device supported features: D and a Minimizing factor:
Mf
Output: Ranked list of web services (RankdList)
1   Initialize RankdList
2  //compute weighting parameters
3   Foreach wi in W do
4   Call frw (W, P), ffw (W, C, D, M) …(1), ( 2)
5   // Rank web services based on relevancy score
6      Foreach wi Ɛ W do
7        Call frs (wi) ……. (3)
8  //Sort Web services based on their resource score
9        RankdList = SortList(wi)
10     End
11  End
12  Return RankdList
```

5. Experimental Results and Evaluation

For the purpose of analyzing the performance of the proposed service discovery mechanism, a proof-of-concept prototype was developed in this paper. This prototype is an implementation of the context-aware service discovery model formulated in [30]. Two components are central in the prototype: Discovery Engine (DisEn) and Node Monitor (NoM). The DisEn performs service requests adaptation, discovery and ranking therefore, it runs the proposed algorithms. From the client side, the discovery engine is launched whenever a client wants to search for a service. Context information needed to support resource-aware service discovery is polled from the client device by the Node monitor. The prototype was designed to run as a service at the background of the device providing service. And to store web service description documents, the prototype uses SQLite database.

To support the proposed discovery operation, we created a web service description document directory on SQLite database, which was embedded in every Android platform. The directory helped to hold web service description documents with which individual web services was searched for.

Using a non-repeated keyword matching scheme, the matching module measured the similarity between the keyword from a service request and keyword obtained from web service description. The matching module did so by extracting keywords from the description document of individual web services, marked by the tag <wsdl:documentation> in the description document directory. These web service description documents were obtained from online web service directories like WebserviceList, WebserviceX and XMethods. In order to support the proposed service discovery approach, we first extended the obtained web service description files following the adapted WSDL-M standard as suggested in [5-27].

The prototype was developed using Android SDK version 21 integrated with Eclipse Juno 4.2. The coding was done in Java language for ICS 4.1 and above. To enable Ad-hoc nodes to connect to each other and communicate for the purpose of discovering services, we employed the Wi-Fi Direct technology. This technology has the capability to create a P2P Ah-hoc network without need for an access point. Also, by employing the native Java Sockets (java.net.ServerSocket and java.net.Socket) Wi-Fi Direct enables the hosting device to receive, process, and sustain an incoming connection instance (an instance of java.net.Socket pointing to the url of the server device) initiated by a client device.

The experimental setup consisted of five mobile nodes (Sony Xperia E1, Infinix X506, HP-Slat 7 HD tablet: Android-6239, Hisense HS-U939, and Samsung Galaxy X2: Android dea5). The prototype (CaSDiM Engine) was installed in each node as depicted in Figure 2. This was to enable nodes to identify and discover themselves as peer nodes forming Ad-hoc Cloud. We then performed the service discovery process by connecting two of the android devices: 1) Hisense HS-U939 cell phone, with internal and external memory of 1.24GB and 1.27GB respectively and running Android 4.2.2 and 2) HP-Slat 7 HD tablet, with an internal memory of 12.15GB. The Hisense HS-U939 device served as the providing device while the HP-Slat 7 HD device functioned as the client. Our preference for the HP-Slat 7 HD device as the client was based on its screen size, which provideed a better view of the experimental results. Figure 3, shows screenshots of the prototype implementation.

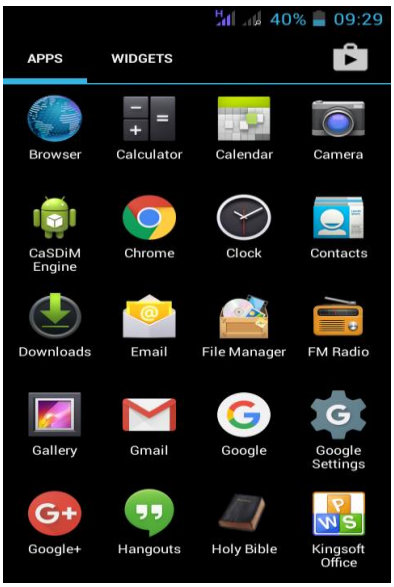

Figure 2. Prototype Installation Screenshot

One thousand web service description documents were used in the experiment. To facilitate easy deployment of the test WSDL files, a web interface (hosted on a free web server) was created with Java and QSL. The web interface employs the WebSocket technology (part of Java EE 7), which defines an API for establishing socket connections between a web browser and a server. This supports creation of a persistent connection for bi-directional and real-time client (Android device hosting services)/server communications. A major advantage of WebSocket technology to AMC in particular is that it supports a single persistent connection, which makes it efficient with regards to resource usage. To be able to receive web service description file that are pushed from the remote server, a client device requires Internet connection to perform a simple process of registration. This registration enlists the client device as a web service subscriber with the web interface.

The experimental analysis focused on the benefits of utilizing device context information to drive proactive service discovery. The goal was to evaluate the effect of such strategy on resource-usage minimization and service discovery efficiency with regards to improving users' experience.

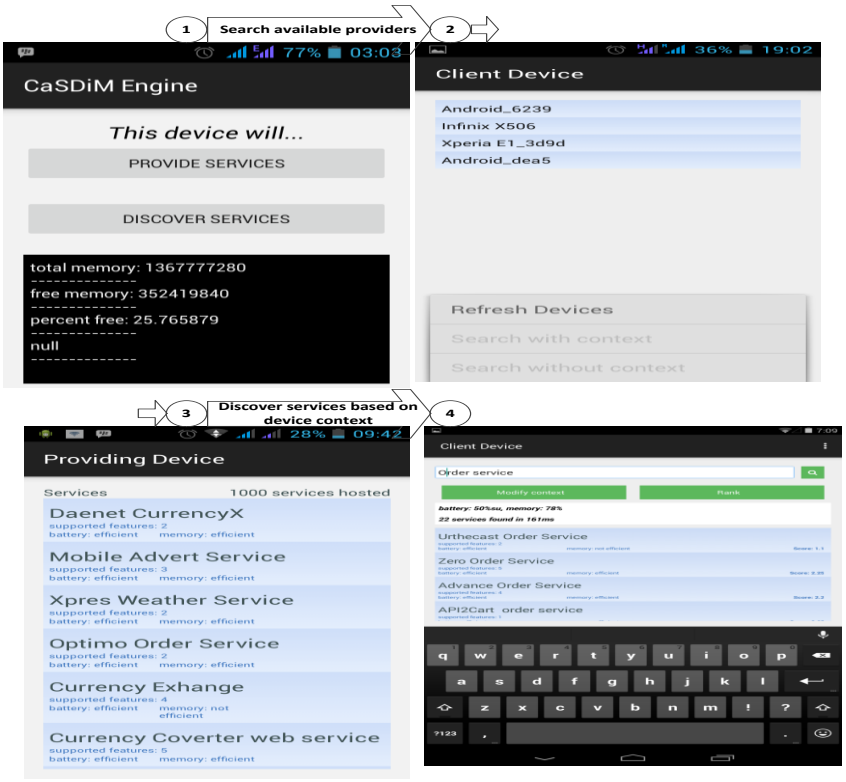

Figure 3. Prototype Implementation Screenshots

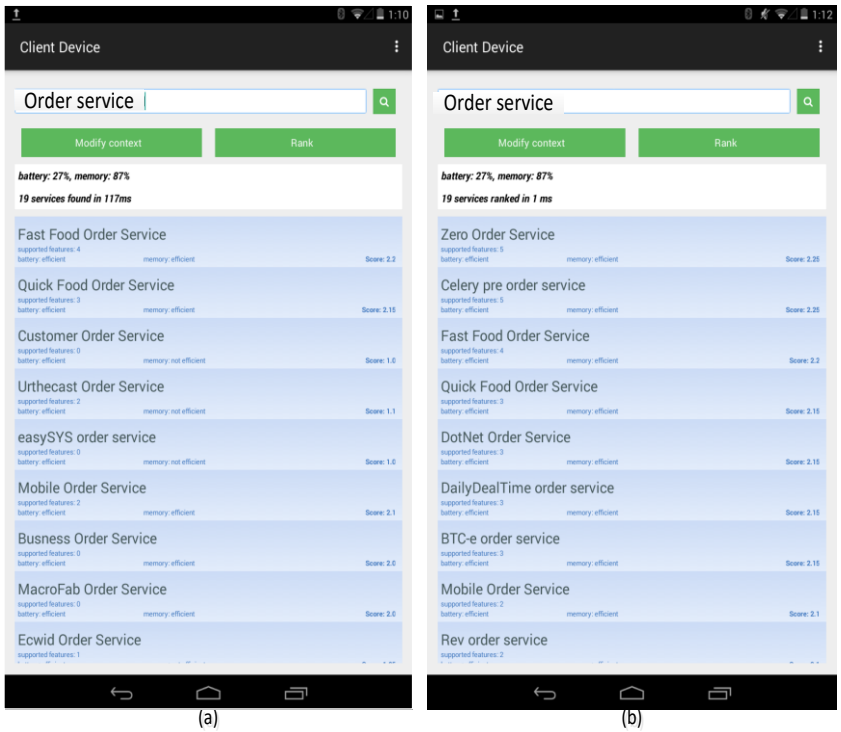

Figure 4. Normal and Context-based Service Retrieval

5.1. Resource Optimization Analysis

This experiment demonstrated how resource consumption could be minimized as a way to enhance effective service discovery in AMC. The experiment showed that proactive and adaptive capabilities would enable AMC service discovery mechanisms to address the issue of limited resources by first being aware of changes in context and then adapting service requests to the current context of client devices before discovering relevant services. In this experiment, four different scenarios were considered as depicted in Table 2. Three of these scenarios represented different device resource states or context, simulated as: i) Critical Resource Level (CRL) - when the device is low in battery and memory *e.g.*, 7%, 11% respectively ii) Normal Resource Level 1 (NRL1) - when the device is only low in memory *e.g.*, 70%, 36% respectively for battery and memory levels and iii) Normal Resource Level 2 (NRL2) - when the device is only low in battery *e.g.*, 41% and 81% for available battery and memory respectively. By independently executing a service request under these different device resource states, we demonstrated that the service discovery mechanism adapted to context change and the time taken to process service requests, while utilizing context information.

The last scenario was the Normal-search. Normal-search referred to service requests that were executed without using device context. The aim of this category of service requests was to enable us know the total number of services that match a particular search key and determine the time taken to process a non-context based service request. With this, we were able to compare the processing times of different service requests.

The number of published services and relevant services were kept constant during this experiment. Out of the relevant published services, nineteen were optimally efficient (both battery and memory efficient), five were only efficient in battery while two were only efficient in memory. With this setup: first, a Normal-search was performed to both determine the number of returned relevant services and the time taken. Second, the same service request (using the same keyword) was repeated twice while the resource context was varied from NRL1 to NRL2. Results of this experiment were captured in Table 2.

Table 2. Services Retrieved Based on the Level of Device Resources

Resource context	Battery and efficient	Only battery efficient	Only memory efficient	Battery and memory inefficient	Filtered-out services	Retrieval time (ms)
CRL: 9%, 11%	19	0	0	0	17	72
NRL1: 70%, 36%	19	5	0	0	12	112
NRL2: 41%, 81	19	0	2	0	15	66
Normal Search	36	x	x	x	0	271

From Figure 5, (derived from Table 2), it was inferred that at CRL, 100% of services retrieved were optimally efficient with regards to resource requirement – that is all returned services met the resource requirement of the client device at its current resource context. To verify this outcome, we conducted a manual check based on the displayed results as presented in Figure 4. The manual check showed that all the services retrieved at CRL state were described to be battery and memory efficient. Though this outcome might be slightly affected by increased number of similar and or relevant services, our result is a reasonable evidence that service discovery efficiency with regards to resource-awareness will remain significantly enhanced. Furthermore, under this same resource

state, a shorter processing time of 72ms was recorded compared to other scenarios. This result pointed to the fact that fewer services were returned at critical resource state.

Interestingly too, results from the experiment on the other two resource states equally generated results that supported the idea of resource optimization. For example, at NRL1 state, out of 24 services retrieved, 79% were optimally efficient and the rest were only efficient in battery. This result meant that no memory inefficient service was retrieved in line with the device's status of 'low in memory'. In contrary, when the resource context was set to NRL2, no battery-intensive service was retrieved. The implication was that at this resource state the device was running low battery therefore, any battery-intensive service was considered not relevant. The significance of this outcome was that since no resource-hungry service was returned coupled with the short processing time recorded when a device's resources were at critical state, it was reasonable to deduce that the overall device resource usage is minimized.

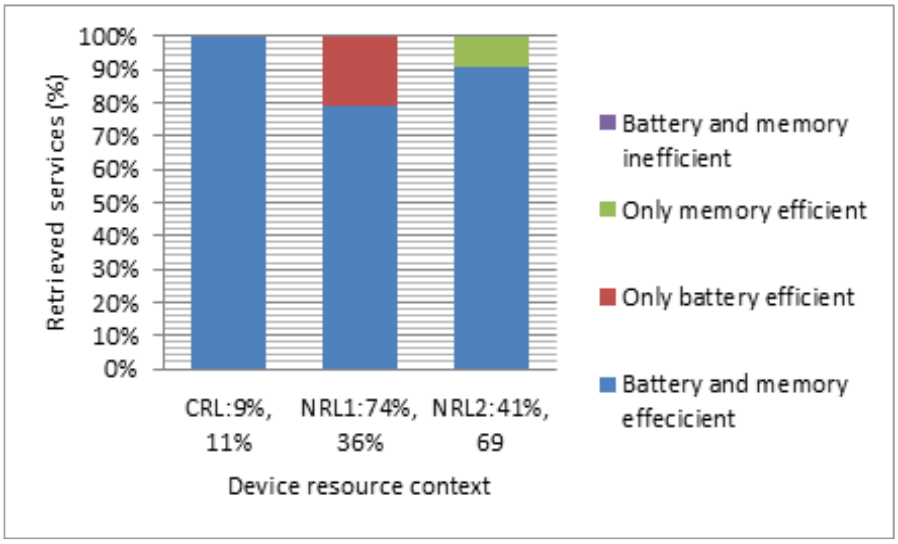

Figure 5. Resource-aware Service Discovery

Another important observation was that the number of filtered-out services had implication for the quality of service discovery (QoSD). In other words, percentage of the ratio of services filtered-out to the total number of services retrieved [27]. Using the results in Table 3, it was deduced that the proposed approach recorded 59% average improvement in the quality of service compared to when device context is not utilized in the discovery process.

5.2. Service Matching Time Analysis

We have reasons to believe that investigating matching time is important; because context overhead can impact on processing time and processing time has a phenomenal implication for computing resources (battery, memory), especially in the context of the Ad-hoc Mobile Cloud. In this experiment six different service requests were independently launched - each under two conditions: i) without device context information and ii) with device context. With the first condition, we can determine the processing time for a service request that is based on a keyword–non-context matching. On the other hand, the search operation under the second condition utilizes device context to adapt service requests and filter retrieved services. The aim was to determine the context matching time.

Table 3. Processing Time with and without Device Context

Retrieved services	Context matching time (ms)	Non-context matching time (ms)
26	10	104
68	17	126
97	28	159
120	33	183
197	51	270
234	94	307

An illustration of processing time is represented in Figure 6a. This graphical illustration was derived from Table 3. From Figure 6a, a proportional relationship between the number of retrieved services and the context matching time was observed. For instance, it took 94ms to retrieve 234 while 26 services required 10ms.

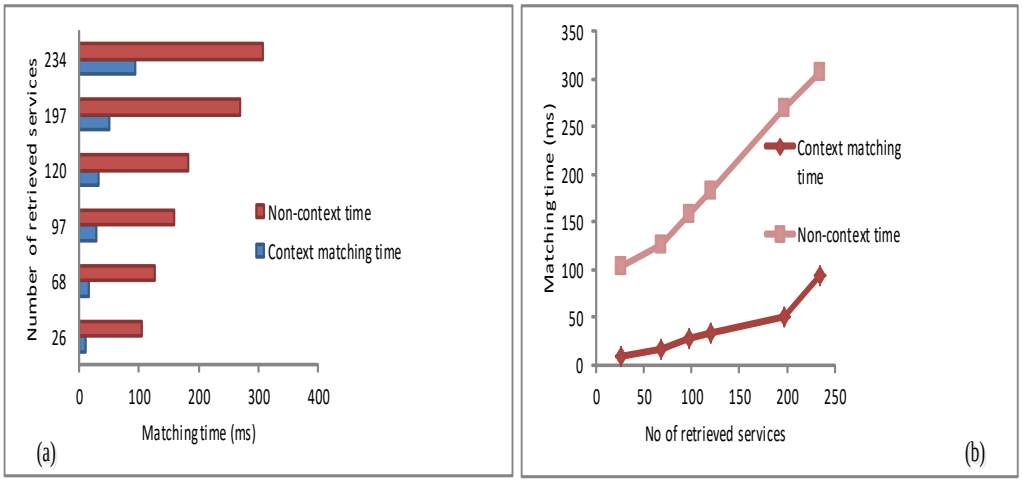

Figure 6. Impact of Device Context on Processing Time

Another significant trend was the fact that the processing time for service requests launched without context was in all cases longer than those executed with device context. Most importantly, we saw a sharp rise in processing time in proportion to the number of retrieved services when device context is not used as shown in Figure 6b. Therefore, it can be reasonably argued that there was a sharp rise in processing time because preconditions (device context) were not attached to the search keyword. Without preconditions, more services were likely to be returned because a large number of similar (but not necessarily relevant) services were retrieved thereby taking more time to process.

However, the reverse was the case when device context was used. Here our approach produced a processing time curve that only rises minimally. This slight rise in processing time suggested that the number of retrieved services did not increase linearly when device context was applied. Having short processing time pointed to minimized resource consumption since long processing time had adverse implications for both battery and memory.

5.3. Service Relevancy Ranking Analysis

This work will not be complete without analysing how relevancy ranking can improve users' experience with regards to the challenge faced by devices with limited display. The results shown in Figure 4a, indicated that it would be problematic for clients using devices with small screens to select the desired service from among returned services. For instance, in Table 4, we labeled retrieved services as "service 1 – 16" according to the

order that they were returned to the client device (unranked) alongside the corresponding relevancy score, as shown in Figure 4a. Similarly, Table 5, shows how the returned services appear on the client's device when relevancy ranking is employed as demonstrated in Figure 4b.

From Table 4, service 1 which tops the list of returned services had a relevancy score of 2.15. However, the fact that retrieved services appeared in the order that they were returned to the client device (unranked) starting from service 1 (the first relevant service to be matched) to service 16 (the last matched relevant service) introduced a problem due to the limited screen size of mobile devices. For example, a manual check on Table 4, showed that the two best or most relevant services (*i.e.*, services with relevancy score of 2.25 and 2.2 respectively) appeared in the 8th and 7th positions on the list of retrieved services.

Table 4. Services Retrieval without Relevancy Ranking

Order of retrieved services (unranked)	Relevancy score
Service 1	2.15
Service 2	1.15
Service 3	2.05
Service 4	1.2
Service 5	2.1
Service 6	2
Service 7	2.2
Service 8	2.25
Service 9	1.1
Service 10	2.1
Service 11	1.1
Service 12	2
Service 13	2.2
Service 14	2.15
Service 15	1.05
Service 16	2.15

Table 5. Service Retrieval with Relevancy Ranking

Order of retrieved services (ranked)	Relevancy score
Service 8	2.25
Service 7	2.2
Service 13	2.2
Service 1	2.15
Service 14	2.15
Service 16	2.15
Service 5	2.1
Service 10	2.1
Service 3	2.05
Service 6	2
Service 12	2
Service 4	1.2
Service 2	1.15
Service 9	1.1
Service 11	1.1
Service 15	1.05

A further manual check on the same table revealed that although four out of the best ten services came under the first ten listed relevant services in the unranked list, only one of them (service 4) appeared before other less relevant services among the first 10 retrieved services. This meant that even if it was assumed that the QoS of the requested service was known beforehand, a client might still have to scroll down several times to pick the right service – this could be daunting in cases where there were several returned services. Consequently, the client might end up invoking a service that did not match the device's context.

However, by adopting relevancy ranking approach which used device context information to rank service as demonstrated in Figure 4b, and Table 5, the result became more user-friendly. For example, retrieved services appeared in their order of relevance with regards to the current resource context of the client device. Looking at Table 5, for instance, service 4 that initially appeared on the 8th position is now top on the list, followed by the second best service that was 7th on the unranked list of retrieved services of Table 4.

This result showed that the proposed service discovery approach offered an easier and probably faster way of discovering and invoking relevant services without necessarily scrolling through the entire list of services returned. This outcome is as a result of the list

of retrieved services being ordered according to the relevance of services to the current context.

4. Conclusion

From our findings, we can conclude that although the use of context aid in the discovery of relevant services, the inherent and unpredictable nature of the Ad-hoc Mobile Cloud environment presented an exceptional scenario. To address this characteristic problem in AMC, it became imperative therefore to design service discovery frameworks supported by algorithms that are adaptive. As a proof of concept, a resource-aware service discovery solution has been presented in this paper for the purpose of exploiting device context information to enhance discovery of services that match the resource capability of client devices.

This paper presented a resource-aware and proactive service discovery strategy achieved by monitoring the battery and memory states of a client device via a Node Monitor component, which dynamically gathered and provided device context data. The prototype strategy through its Discovery Engine uses the context information to perform service requests adaptation, filter, and rank services based on the resource context of a requesting device. The outcome of this work has extended the current discuss on context-based approaches to service discovery in AMC. While others stopped at simply demonstrating the prospect of using context in service discovery, we went further to use the approach to make service discovery adaptive to the requirements of resource-constrained devices. Furthermore, employing context-based approach to realize service ranking helps to generate pertinent output results and shows the tendency to improve users' experience.

Nevertheless, the need to have wide-ranging understanding of the role of device context in improving quality of service discovery in line with the focus of this paper would require extended evaluation. Therefore, conducting further empirical analysis on the effect of device context on critical service discovery parameters such as precision and recall rates and context overhead is an important future direction. Furthermore, the inclusion of multiple filter conditions to tailor services may also come with the likely consequence of introducing time complexity into the algorithm. Hence, there is need to investigate how to achieve a more robust algorithm to minimize time complexity with a view to positively impact on the overall processing time. Realizing the foregoing proposed direction will give credence to the future of AMC service discovery mechanisms that are driven by resource-usage intelligence. With such intelligence, discovered services can be tailored to match the resource capability of devices in addition to fulfilling users' requirements.

References

[1] R. Lacuesta, J. Lloret, S. Sendra and L. Peñalver, "Spontaneous Ad Hoc Mobile Cloud Computing Network," Sci. World J., vol. 19, **(2014)**.
[2] K. S. Wagh and R. Thool, "Web Service Provisioning on Android Mobile Host," Int. J. Comput. Appl., vol. 81, no. 14, **(2013)**, pp. 1–7.
[3] A. Khalifa, "Resilient Hybrid Mobile Ad-hoc Cloud Over Collaborating Heterogeneous Nodes," Collab. ICST, **(2014)**.
[4] D. Mascitti, M. Conti, A. Passarella and L. Ricci, "Service Provisioning through Opportunistic Computing in Mobile Clouds," Fourth Int. Conf. Sel. Top. Mob. Wirel. Netw., vol. 40, **(2014)**, pp. 143–150
[5] 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, **(2010)**, pp. 609–620.
[6] N. A. Saadon and R. Mohamad, "A Comparative Evaluation of Web Service Discovery Approaches for Mobile Computing," Int. Conf. Informatics Eng. Inf. Sci., **(2011)** , pp. 238–252.
[7] 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.

[8] Y. S. Kim and K. H. Lee, "A light-weight framework for hosting web services on mobile devices," Proc. 5th IEEE Eur. Conf. Web Serv. ECOWS 07, **(2007)**, pp. 255–263.
[9] K. Elgazzar, H. Hassanein, and P. Martin, "Personal web services: Architecture and design," Tech. Rep., no. 597, **(2012)**.
[10] 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.
[11] A. Ravi and S. K. Peddoju, "Energy Efficient Seamless Service Provisioning in Mobile Cloud Computing," 2013 IEEE Seventh Int. Symp. Serv. Syst. Eng., **(2013)** March, pp. 463–471
[12] M. Miraoui, C. Tadj, J. Fattahi and C. Ben Amar, "Dynamic Context-Aware and Limited Resources-Aware Service Adaptation for Pervasive Computing," Adv. Softw. Eng., vol. 2011, **(2011)**, pp. 1–11.
[13] L. A. Steller, "Light-weight and Adaptive reasoning for mobile web services, Ph.D thesis, Monash University, Australia," **(2010)**.
[14] 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)**.
[15] L. Steller, S. Krishnaswamy and M. Gaber, "Enabling scalable semantic reasoning for mobile services," Int. J. Semant. Web Inf. Syst., vol. 5, no. 2, **(2009)**.
[16] Y. Kim and K. Lee, "A lightweight framework for mobile web services," Comput. Sci. Dev., (2009).
[17] A. Toninelli, A. Corradi and R. Montanari, "Semantic-based discovery to support mobile context-aware service access," Comput. Commun., **(2008)**.
[18] D. Bianchini, V. De Antonellis, and M. Melchiori, "Lightweight Ontology-Based Service Discovery in Mobile Environments," in 17th International Conference on Database and Expert Systems Applications (DEXA'06), (2007), pp. 359–364.
[19] N. A. Saadon and R. Mohamad, "WSMO-lite based web service discovery algorithm for mobile environment," Int. J. Adv. Soft Comput. its Appl., vol. 5, no. SPECIALISSUE, no. 3, **(2014)**, pp. 75–90.
[20] T. Gruber, "A translation approach to portable ontology specifications," Knowl. Acquis., **(1993)**.
[21] B. Di Martino, G. Cretella and A. Esposito, "Semantic Representation of Cloud Services: A Case Study for Microsoft Windows Azure," in 2014 International Conference on Intelligent Networking and Collaborative Systems, **(2014)**, pp. 647–652.
[22] K. Elgazzar, H. Hassanein and P. Martin, "DaaS: Cloud-based Mobile Web service Discovery," Pervasive Mob. Comput., vol. 13, **(2014)**, pp. 67–84.
[23] K. Elgazzar, P. Martin and H.S. Hassanein, "Empowering mobile service provisioning through cloud assistance," Proc.-2013 IEEE/ACM 6th Int. Conf. Util. Cloud Comput. UCC 2013, **(2013)**, pp. 9–16.
[24] S. N. Srirama, M. Jarke, W. Prinz, S. Birlinghoven and S. Augustin, "Mobile Web Service Provisioning,", **(2006)** February 6.
[25] C. Bobed, R. Yus, F. Bobillo and E. Mena, "Semantic reasoning on mobile devices: Do androids dream of efficient reasoners?," Web Semant. Sci. Serv. Agents World Wide Web, **(2015)** September.
[26] S. Abolfazli, Z. Sanaei, A. Gani, F. Xia and L. T. Yang, "Rich Mobile Applications: Genesis, taxonomy, and open issues," J. Netw. Comput. Appl., vol. 40, **(2014)**. April, pp. 345–362.
[27] K. Elgazzar, P. Martin and H. S. Hassanein, "Personalized Mobile Web Service Discovery," in 2013 IEEE Ninth World Congress on Services, **(2013)**, pp. 170–174.
[28] Z. Maamar, S. Tata, D. Belaïd and K. Boukadi, "Towards an Approach to Defining Capacity-Driven WS," in 09 Int. Conf. on Adv. Info Net & Appl., **(2009)**, pp. 403–410.
[29] R. L. Cilibrasi and P. M. B. Vitanyi, "The Google Similarity Distance," IEEE Trans. Knowl. Data Eng., vol. 19, no. 3, **(2007)**, March pp. 370–383.
[30] 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.

Authors

Dominic Afuro Egbe, holds B.Sc. (Hons) degrees in Computer Science and Computer Science and Information Systems from the University of Calabar, Nigeria (2004) and University of Venda, South Africa (2014) respectively. He is currently studying for his research Masters degree in Computer Science at the University of Zululand, South Africa. His research interest includes ICT for Development and Context-aware Service Discovery in Mobile/Ad-hoc Mobile Cloud.

Murimo Bethel Mutanga, holds a Masters degree in Computer Science from the University of Zululand, South Africa. He is currently doing his PhD with the same institution. His research interest includes wireless ad-hoc networks, Mobile Cloud Computing and the development of low cost technologies for Africa.

Prof M.O Adigun, holds a research PhD in Computer Science from Obafemi Awolowo, University, Ile-lfe, Nigera (1989). Currently he is Professor/Head Department of Computer Science, University of Zululand, South Africa. His research interest includes Software Engineering, Mobile Computing, modeling and Simulation and Performance, Analysis of Computer System.