International Journal of Digital Information and Wireless Communications (IJDIWC) 4(3): 284-291
The Society of Digital Information and Wireless Communications, 2014 (ISSN: 2225-658X)

An Active Middleware for Secure Automatic Reconfiguration of Applications for Android Devices

S. Kami Makki
Department of Computer Science, Lamar University, Texas, USA
kami.makki@lamar.edu

Karim Abdelrazek
Department of Computer Science, The City College of New York, New York, USA
kabdelr00@ccny.cuny.edu

Shui Yu
School of Information Technology, Deakin University, Victoria, Australia
syu@deakin.edu.au

***Abstract*—With the prevalence of smart phones and the role they play in the lives of consumers, the demand for high performing mobile computing is apparent. Although smartphones today are feature-rich, they are still resource-scarce; they are limited by their memory, energy, and processing power. These limitations constrain the ability of these devices to perform intensive computational tasks without compromising the consistency of mobile device performance. As such, the development of a dynamic and intelligent mobile middleware solution can ameliorate these constraints through the utilization of surrogate computing methodologies. In this paper, we present an intelligent and active middleware solution for secure automatic reconfiguration of applications for android devices. This middleware offers efficiency and enhances the conservation of resources for these devices.**

***Index Terms* - Application offloading; Context awareness; Feature models; Mobile middleware; Reconfiguration; Surrogate computing**

I. INTRODUCTION

A smartphone is a mobile device, which offers PC-like functionality. A variety of operating systems exist for smartphones today; examples of this include Apple's iOS, Android, Symbian, and Windows Mobile. Each of these operating systems allow for the execution of smartphone applications, which range from gaming to productivity applications.

These days, smartphones are an integral part of many people's lives. As a recent report indicates, the number of people that have mobile phone is growing rapidly [1]. Out of the five billion mobile phones available, approximately one billion smartphones are in use worldwide. The United States alone has over 91 million smartphone users and it is projected to grow much more [1].

As users become increasingly dependent on smartphones for their daily life (e.g. banking, maps, ticketing, etc.), the need for mobile devices with rich computing capabilities is apparent. However, smartphones have several limitations: they have finite energy resources, contain highly variable network connections, and are resource-poor compared to standard desktop computers. With these limitations, smartphones generally cannot perform computationally-intensive tasks without compromising mobile performance. Thus, these constraints lower the quality of interaction between mobile devices and their users.

Constraints facing mobile devices can be alleviated in a variety of ways. Manufacturers of mobile devices produce smartphones with improved phone specifications, such as faster processors and bigger batteries. However, generating high-end hardware for these devices affects manufacturing and device

This material is based upon the work supported by the National Science Foundation under Grant No. 0851912.
Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

costs while limiting the product in other ways, such as memory size and weight. While this approach makes some improvements on mobile device resources, it still leaves the devices with significant limitations.

Another approach involves offloading computational tasks to remote servers in order to conserve mobile resources. This approach is generally described as “Cyber Foraging”, or “application offloading”. This approach is a key step in realizing the pervasive computing vision [21]. Solving the problem of resource constraints on mobile devices is an important achievement for the user and for mobile computing. In order to achieve this, an intelligent and active mobile middleware is necessary.

The outline of this paper is as follows: In the next section, we discuss relevant background information pertaining to surrogate computing, automatic reconfiguration, remote servers, and mobile middleware. Section III summarizes and highlights the key features of our proposed middleware. Section IV presents the conclusion and future work.

II. Related Work

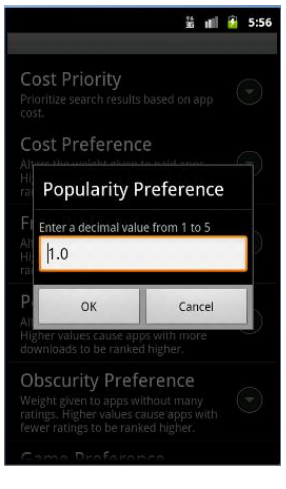
A. Mobile Applications

A mobile application is a program designed for mobile devices. There are a plethora of mobile applications (apps) available today; for instance, the iPhone’s App Store is home to over 500,000 mobile applications [2]. Similarly, Android’s Google Play contains approximately the same amount of mobile applications [3]. Mobile applications range from games to navigation apps; each application put different strains on the resources of a mobile device. For example, a movie streaming application heavily consumes networking resources, while a game with detailed 3D graphics consumes processor resources.

B. Offloading

For many years, researchers have investigated the issues of conserving mobile resources through the offloading of computationally intensive tasks. Rudenko et. al. [4] was among the first to introduce the idea of remote execution to conserve energy of mobile devices. The authors in [5, 18, 20, 22] reintroduced the concept of remote execution in pervasive environments, coining the term “Cyber Foraging.” Cyber Foraging revolves around mobile devices offloading computational tasks to resource-rich “surrogates” in order to conserve mobile resources such as memory, and battery life. This approach has been investigated [6] and shown to be a viable solution [6, 7, 8] for minimizing the consumption of resources of mobile devices. Recent research revolves around the utilization of cloud resources for offloading [7, 8, 9, 10].

To show how the surrogate computing can be utilized easily for computation, we have developed an application which extends the Google Play’s search functionality. Google Play provides users with access to hundreds of thousands of applications. However, it is a challenge for users to find applications that they may like, since they have been given limited search options. That is because Google Play only allows searching for apps by keyword or popular apps featured, while users may want to search for apps with different criteria. Our developed application gives users the ability to set a variety of preferences (such as app type, popularity, and cost) in order to tailor search results to their interests.


5:56

Cost Priority
Prioritize search results based on app cost.

Cost Preference

Popularity Preference
Enter a decimal value from 1 to 5
1.0

Higher values cause apps with more downloads to be ranked higher.

Obscurity Preference
Weight given to apps without many ratings. Higher values cause apps with fewer ratings to be ranked higher.

Figure 1: User Preference Screen

As shown in Figure 1 each preference is given a value from 1 to 5, with 5 indicating the highest priority for that particular preference. Once the user makes a search query for a particular type of app, this query is offloaded to a private server. The server sends the query to Google Play for retrieving the search results. The server calculates a score for each result based on the user preferences. Then, it organizes the results so only the apps relevant to the user's interests appear, and finally the server sends the results to the user. Without using a surrogate, this process is unacceptably slow, as memory and processing resources of a mobile device are limited. Our experiment demonstrates the benefits of offloading to minimize mobile device resource usage, highlighting the importance of having automatic reconfiguration in middleware.

C. Remote Server

In order for tasks or applications to be offloaded to servers remotely, servers must have the following features to run effectively:

- *Communication Manager* **-** A communication manager is needed to operate data to and from mobile devices.
- *Virtual Machine Manager***-** This section of a remote server manages the creation, execution, and storage of the virtual machine within the server.
- *Decryption Service* – A decryption service is required to decrypt data exchanged between severs and mobile device.

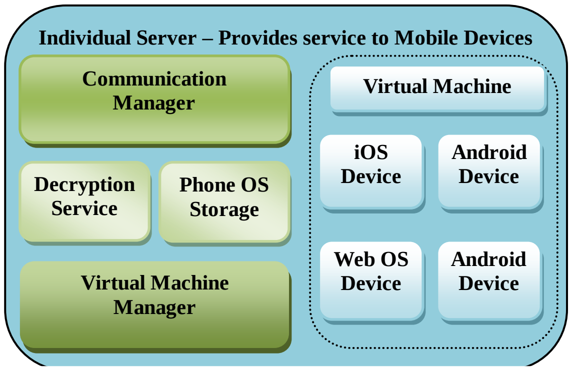


Figure 2: Remote Server Design

Figure 2 illustrates the design of such a remote server [11]. The server houses multiple virtual machines, with each one executing different applications for mobile devices. It is important to note that hosting multiple virtual machines is only possible if the server is scalable. The operating system of the virtual machine is determined by the operating system of the mobile device. As previously mentioned, the communication manager coordinates all communication between the mobile device and remote server; it also accesses the decryption service to decrypt incoming information, and encrypt outgoing data.

D. Middleware

Middleware is the management layer that manages service, monitors access, plans the communication between local and remote services and efficiently executes the operations required for switching services [12]. As such, it is responsible for the reconfiguration of applications for purposes of offloading. There are three main types of reconfigurations [11]:

Static Reconfiguration – With static reconfiguration, applications are always offloaded whenever they are executed on the mobile device. This process occurs at the time of installation, and before execution of the application, allowing the user to change preferences for static reconfiguration of a specified application.

Dynamic Reconfiguration – Unlike static reconfiguration, dynamic reconfiguration is a process of offloading based on the state of the device. This occurs when there is a noticeable change in performance. In order to determine when a dynamic reconfiguration should take place, cost benefit formulas are used. These formulas take into account several factors, such as battery level, user preferences, bandwidth description, and network strength. The formulas are calculated individually and compared. If the benefit of offloading is higher than the cost, the middleware will offload the application to a remote server.

Contextual Reconfiguration – Contextual reconfiguration shares similarities with Dynamic configuration, with the exception of where applications are offloaded. With contextual reconfiguration, only nearby servers are used to host applications. The term originates from the use of the device's context, specifically its location. Utilizing contextual reconfiguration may reduce the latency of offloading from the mobile device to the remote server.

Criteria for Good Middleware

In the context of being able to perform efficient automatic reconfiguration of applications, a good mobile middleware solution must generally meet one or more of these five criteria:

i. **Complete Service** – Along with reconfiguration, middleware must offer other features, such as fault solution systems and intelligent knowledge-base.
ii. **Application Portability** – Middleware compatibility with applications is essential for providing efficient automatic reconfiguration.
iii. **Offloading Pieces of Application** – Small portions of an application or the application itself may need to be offloaded if they are computationally intense.
iv. **Triggering of Reconfiguration.**
v. **Contextual Reconfiguration** – Contextual reconfiguration allows for data to be sent across shorter distances, allowing for more efficient offloading.

E. *Offloading*

Related work in this area has touched upon different aspects of mobile middleware development and mechanisms for offloading.

Xiaohu Gu et. at.'s work describes an "adaptive offloading" mechanism which relies on a distributed offloading platform and an offloading inference engine to compute small portions of an application remotely [13].

Similarly, Gonzalo Huerta-Canepa et. al., worked on a version of this offloading technique that revolved around monitoring application behavior and offloading based on a mathematical model [15].

Mararasu et al.'s work describes a service which must be created and configured by developers [12].

Byung-Gon Chun et. al., discussed the design and implementation of the "CloneCloud" system [7, 8], which enables mobile applications to seamlessly off-load threads from mobile devices onto device clones operating in a computational cloud. This is done through the utilization of static analysis and dynamic profiling to partition applications in a manner which optimizes application execution. Implementation of this system was shown to lead to a 20 times execution speedup for certain applications [8].

III. Middleware Design

We base the design for our system on our earlier design [11]. Our propose system provides an intelligent mobile middleware solution offering context-awareness, an intelligent knowledge-base to handle errors, and automatic reconfiguration of applications. This middleware strives to give developers a way to make their applications reconfigurable for offloading with little application re-development. This section summarizes how our active middleware works, as well as, highlights its key components.

In addition to offering a solution for all the criteria listed in Section II (sub-section D), our system also contains the following features:

- *Transparency* - Tasks performed by middleware should be inconspicuous; the user should be oblivious to anything but the content being received on the mobile device.
- *Modularity* - Modularity is necessary in order to allow for easy modification of the middleware system.

These features are necessary for a well-developed middleware solution, and ensure efficient performance on mobile devices.

A. *Application Compatibility*

Applications with our system have three requirements:

- XML Information File - An XML information file is necessary to tell to our proposed middleware whether the application is able to be offloaded and if static reconfiguration is preferred.
- Separation of Interface and Computation – Applications need to have an interface which interacts with the user and a computation portion which manipulates data.
- Transparent RPC - Transparent Remote Procedural Calls (RPC) is needed for transferring data between various portions of the application. This would utilize a serialization format to communicate between the two portions. When this information is shared with the application, the application would translate this data into the data used by the application.

B. *Reconfiguration*

The proposed middleware supports dynamic, static, and contextual reconfiguration:

Dynamic Reconfiguration - Our system utilizes the cost benefit using the formulas shown below:

Cost of offloading = Network Type + Battery Level + Bandwidth Restrictions + User Preferences+ Current Network Strength + Application Currently Reconfigured (1)

Benefit of offloading = Processor Utilization + Memory Utilization (2)

Static Reconfiguration - As mentioned in Section II, with static reconfiguration the application is always offloaded. However, this type of reconfiguration is done only at the user's request, or if a developer places a set of conditional rules for which static reconfiguration must occur in order for the application to run successfully on the mobile device.

Contextual Reconfiguration - With contextual reconfiguration, applications are only offloaded to nearby servers. When the user permits contextual reconfiguration, the procedure that follows is similar to that of dynamic reconfiguration (e.g., utilizing the cost-benefit formulas to determine whether offloading is necessary); however, transferring of the application would have to occur much faster. The middleware would seek out available servers nearby, and then offload the application once one is found. The process is only triggered when the users move further away from their current locations.

To demonstrate the performance of our proposed middleware a tic-tac-toe game application was developed in Makki et. al., [14]. In this application, once the middleware has decided that offloading computation of a program is beneficial to the mobile device, the middleware searches for a server. After it locates a remote sever, the middleware pauses the selected application and offloads it. Meanwhile, the server creates a virtual machine for the device and receives the application. After the application is received, the server executes the application and the middleware resumes the interface portion of the application; these two portions then communicate over transparent RPC. With this constant communication, the application has no knowledge that it has been reconfigured, and continues to run smoothly.

C. *Intelligent Knowledge Base*

The middleware's knowledge-base is a fault repository, a collection of previously encountered errors along with their solutions. If the middleware encounters a new problem, it first attempts to use a relevant solution. If that fails, the middleware will try to solve the problem through trial and error [11]. When multiple failures are accumulated, they are recorded into the fault repository, as well as, their attempted solutions. The knowledge base consists of three columns: there is one for storage of problem description (stored as keywords), one for attempted fixes, and the third is for problem solutions. Figure 3 illustrates how the knowledge base works.

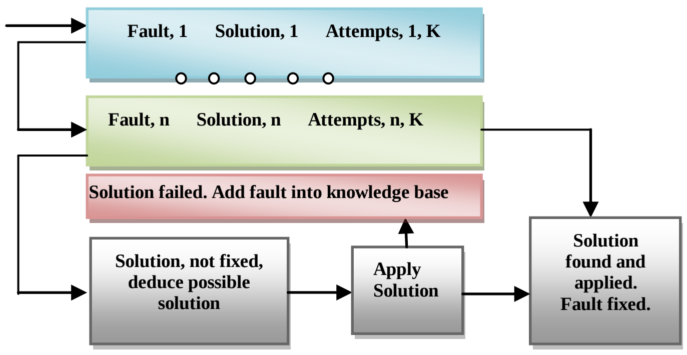

Fault, 1
Solution, 1
Attempts, 1, K
Fault, n
Solution, n
Attempts, n, K
Solution failed. Add fault into knowledge base
Solution, not fixed, deduce possible solution
Apply Solution
Solution found and applied. Fault fixed.

Figure 3. Intelligent Knowledgebase Operation: K = number of previous faults, n = number of faults

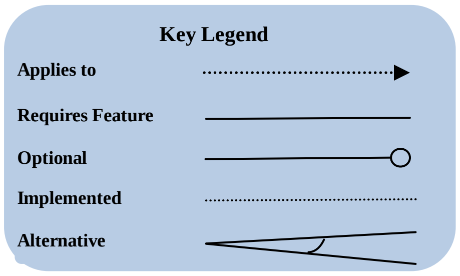


Figure 4. Feature Model Key/Legend

D. *Feature Model*

The feature model as shown in Figure 4 provides design information about our proposed system, which allows us to identify all of the commonalities that would exist between different versions of the middleware on different devices [11]. The feature model also allows the middleware to be easily visualized, as shown in Figures [5-8].

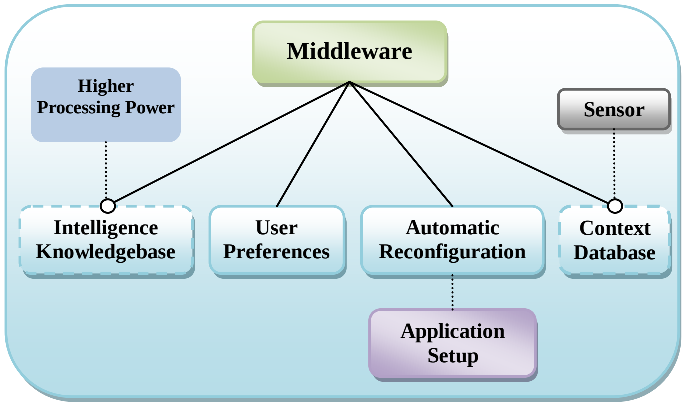


Figure 5. Architecture Overview

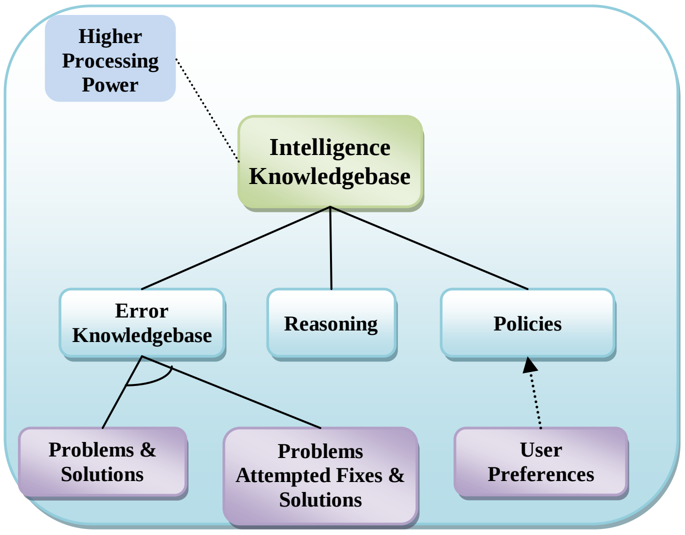


Figure 6. Intelligence Knowledgebase Feature Model

Our system has two main features: automatic reconfiguration, and user preferences as shown in Figure 5. It also has optional features such as Context Database and Intelligence Knowledgebase as shown in Figure 5. As shown in Figures [6-8], there are many alternative sub-features, which together implement the super-feature, or make up some portions of it. This highlights the modularity of our propose system, allowing for the creation of a mobile middleware suited for several mobile devices.

E. Data Serialization Format

As mentioned before, automatic reconfiguration is a key component of our intelligent and active mobile middleware solution. Data communication is extremely important in the process of offloading application. Therefore, a reliable serialization format is essential for an effective middleware. Sumaray and Makki, compared the efficiency of data serialization formatting such as: XML (eXtensible Markup Language), JSON (JavaScript Object Notation), ProtoBuf, and Thrift for a mobile platform [16].

They showed that the faster formatting for data serialization and deserialization is the Binary formatting (ProtoBuf and Thrift). However, when taking into consideration other factors such as human readability and adaptability, JSON proven to be the best overall data serialization format [16]. Therefore, we propose to use JSON as the default serialization format for our proposed middleware. This will provide the middleware with an efficient means of data communication, especially in transparent RPC (discussed earlier). It will also decrease the size of data being transferred to remote servers, which is important since smartphones typically have a limited capability for transferring large amount of data.

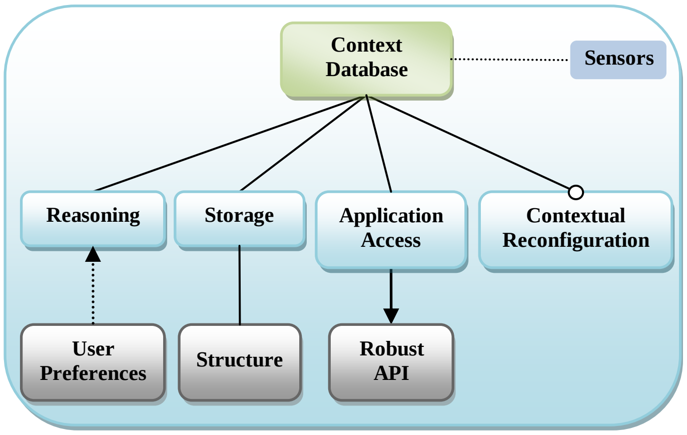


Figure 7. Context Database Feature Model

F. Fidelity Adaptation

Fidelity Adaption is another approach to conserving a mobile device's resources. Fidelity refers to an application's metric of quality [17] that can be modified to conserve energy resources. For example, if a user runs a visually intensive application on his/her mobile device and observes a low frame rate (because the device cannot handle the intense computations), then the user can lower the visual quality of the application so it can run smoothly.

Therefore, the fidelity adaptation approach is ideal as it provides a method of minimizing resource usage in the absence of surrogate computers. Furthermore, it can be combined with "Cyber Foraging" techniques to optimize mobile device efficiency [18, 19]. For example, this combination would allow a lower quality visual application to be offloaded to a surrogate which leads to a smaller amount of data being communicated to a remote server. This results in a lower latency between the server and the mobile device because smaller data can be parsed quickly.

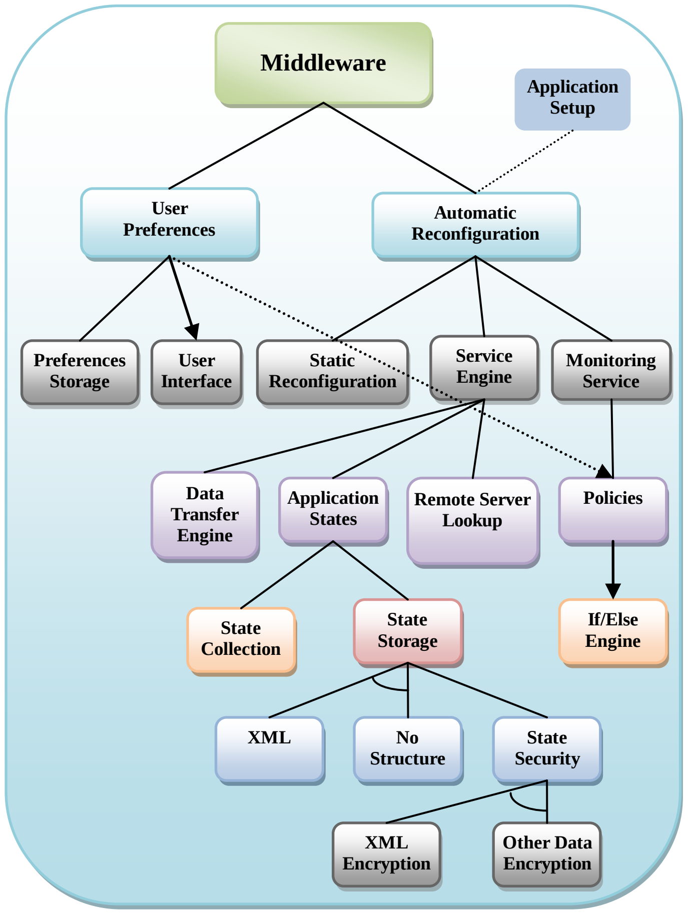


Figure 8. Automatic Reconfiguration and User Preferences Feature Model

Adding this feature to our system enhances its capabilities and it can lead to performance gains on mobile devices. In order to make it work with the middleware system, the XML information file described in sub-section A of section III would need to have additional information specifying whether the application quality can be adjusted. If it can, then the middleware will lower its quality and offload it when necessary to conserve energy of mobile devices.

IV. CONCLUSION

In this paper, we have investigated information pertaining to middleware technology and surrogate computing [9]. These technologies improve conservation of resources of mobile devices and improve their capability. Therefore, we introduced a new middleware system, which enables automatic reconfiguration of applications. It is a complete mobile middleware system containing key features such as full application offloading, and an intelligent knowledgebase for fault tolerance. What distinguishes our propose middleware system from other existing middleware is the utilization of fidelity adaptation to optimize mobile device resource conservation. This feature gives mobile devices the ability to save energy in the absence of surrogate computers, and also facilitate further conservation of resources when combined with application offloading. All these proposed features make our system a fast, versatile, and developer-friendly middleware solution.

ACKNOWLEDGMENT

The authors wish to thank Matthew Williamson, a Lamar University student, who has assisted in programming tasks of this project.

V. REFERENCES

[1] Go-Gulf.com, Infographic SMARTPHONE USERS AROUND THE WORLD – STATISTICS AND FACTS (prepared on 2nd January 2012), http://www.go-gulf.com/blog/smartphone.
[2] Apple, Apps for iPhone, http://www.apple.com/iphone/apps-for-iphone/
[3] Google, Android Apps on Google Play, https://play.google.com/store/apps?hl=en.
[4] Alexey Rudenko and Peter Reiher. Saving Portable Computer Battery Power through Remote Process Execution. In Proceedings of Mobile Computing and Communications Review, Vol.2, No.1, pp.19-26, 1998.
[5] M. Satyanarayanan. Pervasive Computing: Vision and Challenges, IEEE Personal Communications, Vol.8, No.4, pp.10-17, 2001.
[6] K. Kumar and Y. H. Lu. Cloud Computing for Mobile Users: Can Offloading Computation Save

Energy. In Proceedings of Computer, Vol.43, No.4, pp.51-56, April 2010.

[7] C. Byung-Gon and M. Petros. Augmented Smartphone Applications Through Clone Cloud Execution. In Proceedings of 12th workshop on Hot Topics in Operating Systems (HotOS XII), May 2009, Monte Verita, Switzerland.

[8] Byung-Gon Chun, Sunghwan Ihm, Petros Maniatis, Mayur Naik, and Ashwin Patti. CloneCloud: Elastic Execution between Mobile Device and Cloud. In Proceedings of the 6th international conference on Computer systems (EuroSys '11), pp. 301-314, 2011, New York, USA, DOI=10.1145/1966445.1966473, http://doi.acm.org/10.1145/1966445.1966473.

[9] Roelof Kemp, Nicholas Palmer, Thilo Kielmann, and Henri Bal. The Smartphone and the Cloud: Power to the User. In Proceedings of the MobiCloud 2010.

[10] Xinwen Zhang, Anugeetha Kunjithapatham, Sangoh Jeong, and Simon Gibbs. Towards an Elastic Application Model for Augmenting the Computing Capabilities of Mobile Devices with Cloud Computing. In MONET 16(3): pp. 270-284, 2011.

[11] Setfan Ashmore and S. Kami Makki. IMISSAR: An Intelligent, Mobile Middleware Solution for Secure Automatic Reconfiguration of Applications, Utilizing a Feature Model Approach. In Proceedings of the 5th International Conference on Ubiquitous Information Management and Communication (ACM SIGKDD-SIGAPP ICUIMC 2011), pp. 21-23 February, 2011, Seoul Korea.

[12] Alin F. Murarasu and Thomas Magendanz. Mobile Middleware Solution for Automatic Reconfiguration of Applications. In Proceedings of the 6th International Conference of Information Technology: New Generations (ITNG 09), pp.1049-1055, 2009, Las Vegas USA.

[13] Xiaohui Gu, Alan Messer, Ira Greenberg, Dejan Milojinic, and Klara Nahrstedt. Adaptive Offloading for Pervasive Computing. In IEEE Pervasive Computing Magazine, Vol.3, No.3, pp. 66-73, 2004.

[14] S. Kami Makki, Narasimha B. Srirangam, Venkata S. Aiswarya, and Shui Yu. Utilizing Intelligent Middleware for Reconfiguration of Applications on Android. In Proceedings of International Conference on Convergence and Hybrid Information Technology (ICHIT'11), pp. 81-89. Springer, Heidelberg, September 2011, Daejeon, Korea.

[15] Gonzalo Huerta-Cánepa and Dongman Lee. An Adaptable Application Offloading Scheme based on Application Behavior. In Proceedings of the workshop of the 22nd International Conference on Advanced Information Networking and Applications, pp. 387-392, March 2008.

[16] Audie Sumaray and S. Kami Makki. A comparison of data serialization formats for optimal efficiency on a mobile platform. In Proceedings of the 6th International Conference on Ubiquitous Information Management and Communication (ICUIMC 2012). ACM, DOI=10.1145/2184751.2184810, .New York, NY, USA, 2012.

[17] Rajesh Krishna Balan. Powerful Change Part 2: Reducing the Power Demands of Mobile Devices. In IEEE Pervasive Computing, Vol. 3, No. 2, pp. 71-73, IEEE Press, 2004.

[18] Jason Flinn and M. Satyanarayanan. Energy-aware adaptation for mobile applications. SIGOPS Operating System Rev. Vol. 34, No. 2, pp. 13-14, April 2000. DOI=10.1145/346152.346170 http://doi.acm.org/10.1145/346152.346170.

[19] Eyal DeLara, Dan S. Wallach, and Willy Zwaenepoel. Puppeteer: Component-based Adaptation for Mobile Computing. In Proceedings of USITS, pp. 14-25, 2001.

[20] Eduardo Cuervoy, Aruna Balasubramanianz, Dae-ki Cho, Alec Wolmanx, Stefan Saroiux, Ranveer Chandrax, and Paramvir Bahlx. MAUI: Making SmartPhones Last Longer with Code Offload. In Proceedings of MobiSys, 2010.

[21] M. D. Kristensen. Scavenger: Transparent Development of Efficient Cyber Foraging Applications. In Proceedings of IEEE International Conference on Pervasive Computing and Communications (PerCom 2010), pp.217-226, 2010.

[22] Rajesh Balan, Jason Flinn, M. Satyanarayanan, Shafeeq Sinnamohideen, and Hen-I Yang. The Case for Cyber Foraging. In Proceedings of the 10th workshop on ACM SIGOPS European workshop (EW 10). New York, NY, USA, pp. 87-92, 2010. DOI=10.1145/1133373.1133390. http://doi.acm.org/10.1145/1133373.1133390.