CARMM: Context-Adaptable Reflective Middleware over Mobile Environment

Khaing Lae Lae Soe
Faculty of Information and Communication Technology
University of Technology (Yatanarpon Cyber City)
Pyin Oo Lwin, Myanmar

Thiri Haymar Kyaw
Faculty of Information and Communication Technology
University of Technology (Yatanarpon Cyber City)
Pyin Oo Lwin, Myanmar

Abstract

Today people widely use mobile devices as versatile tools for their everyday works. End users desire their mobile applications to use more effectively and efficiently with the maximize use of their resources in mobile environment. In this paper, we propose CARMM (Context-Adaptable Reflective Middleware over Mobile environment) to adapt application behavior over changing mobile device's resources such as processor usage, battery, memory and available bandwidth. CARMM is a reflective middleware that respond suitable application version based on client device's characteristics. Previous reflective middleware solutions adapt application behavior based on user's preferred Quality of Service (QoS). In our proposed system, application must be learned to make response the best output regardless of user's preference. Multiple linear regression is used to learn application and stored obtained rules in application profiles. The system can keep the balance of resource utilization in mobile devices and appropriate service for users.

1. Introduction

Common Object Request Broker Architecture (CORBA), the Distributed Component Object Model (DCOM), and Java Remote Method Invocation (RMI) are early stage middlewares that abstract the low-level TCP/IP communication details and replace the communication interface with a local procedure call or function invocation. However, traditional middleware is limited in its ability to support adaptation and its limiting capability in mobile computing environment due to its overhead.

On the other hand, message-oriented and transaction-oriented middleware are also not suitable for mobile setting because the former requires large amount of memory and the latter wants high computational load [1]. These middlewares hide implementation details in middleware layer and cannot be seen from both end-users and developers. Hiding implementation details means that all the complexity is managed internally by the middleware layer; middleware is in charge of taking decisions on behalf of the application, without letting the application influence this choice [1].

To guarantee the best quality of service, computationally heavy weight system with large amount of code and data is needed. Heavyweight systems cannot however run efficiently on a mobile device as it cannot afford such a computational load. Variations in resource availability, network connectivity, and hardware and software platforms impact greatly the performance of user applications. It is desirable for the applications to adapt their behaviors to resource limitations and variations. Hence, the adaptation task is best coordinated by a middleware that is able to cater for individual application's need on a fair ground, while maintaining optimal system performance.

There are two types of adaptation provided by adaptive middleware: static and dynamic. Static adaptation can occur during compile or startup time, and dynamic adaptation occurs only after startup time. Reflection is employed to formulate dynamic context adaptation. Therefore context-aware reflective middleware can monitor real-time contextual information and adapt the application

behaviors to the context changes. Thus, it provides a powerful reconfiguration approach to build adaptive applications [2].

Reflection generates significant advantages over lively mobile situation, i.e., change in context (such as the device location), change in network conditions (such as variation in bandwidth and latency) and change in available resources (such as device's battery and memory level) [3]. Computational reflection enables middleware to inspect the application specification, reason about, and adapt itself at run time. This process is called absorption whereas application can alter middleware behavior, known as reification.

Middleware takes place conflicts to decide which policy is taken up to deliver the service. In our proposed system, mobile application must be first learned to produce appropriate rules for each its version. Our design utilize machine learning approach of multiple linear regression to be resolved these conflicts. Although learning stage makes computational overhead for middleware, it is offline operation and it brings about lightweight calculation by storing predefined rules in its profile.

The rest of the paper is organized as follows. Section 2 expresses related work and the concept of reflective middleware is explained in Section 3. Section 4 presents our proposed system and Section 5 describes how to learn rules in middleware with locally weighted regression. Section 6 is about our case study application. Finally, we conclude this paper in Section 7.

2. Related Work

Mobile middleware research has focused on addressing the key problems, namely: unpredictable network connections, poor network Quality of Service (QoS), ad-hoc interaction, and limited end-system resources [3]. The concept of reflection was first introduced by Smith in 1982 [4] as a principle that allows a program to access, reason about and alter its own interpretation in the procedural programming language. After ten years later, reflection has been applied to the field of operating systems [5] and after that in distributed systems [6]. The principle of Reflection has often been used to allow dynamic reconfiguration of middleware and has proven useful to offer context-awareness [1].

DynamicTAO and OpenORB are reflective middlewares that comprise components enabling on-the-fly reconfiguration [7]. Middlewares that exercise both reflection and context-awareness are CARISMA and ReMMoC. CARISMA is a mobile computing middleware which exploits the principle of reflection to enhance the construction of adaptive and context-aware mobile applications. CARISMA proposes how context changes should be handled using policies. The conflicts of policies are resolved using a microeconomic approach that relies on a particular type of sealed-bid auction to resolve the service policy confliction at execution time [8].

The ReMMoC project examines the use of reflection to accommodate heterogeneity requirements imposed by both applications and underlying device platforms. It can dynamically adapt its underlying behaviour between different concrete middleware implementations [9]. Other implementation MARCHES separates adaptation concerns from other constructions of the application and leads to simplification for application development, low-overhead and good robustness for mobile computing devices [2]. MUSIC enables the self-adaptation of mobile and ubiquitous applications in the presence of *Service-Oriented Architectures* (SOA). It focuses on changes in the service provider landscape in order to plug in interchangeably components and services providing the functionalities defined by the component framework [10].

All previous reflective middlewares acquire user's preferences to achieve desired quality of applications. No reflective middleware solution exists that makes automatically adaptation of application according to available resources. However, FSAM, context-aware mobile computing middleware, formulate the service adaptation process by using fuzzy linguistic variables and membership degrees to define the context situations and the rules for adopting the policies of implementing a service [11].

On the other hand, logistic regression has been already used for gird computing [12] and we wishes to use this method because it can handle continuous input streams very well and current prediction is done by local functions which are using only a subset of the data. We use locally weighted learning to find out the mapping between input context and output quality. After that multiple

linear regression is employed to predict suitable application version based on current contexts of client's mobile device.

3. Reflective Middleware

"Reflection is the integral ability for a program to observe or change its own code as well as all aspects of its programming language - even at runtime" [16]. Reflective middleware moves reflection to the middleware level. For this purpose, a reflective system maintains a representation of itself that is causally connected to the underlying system that it describes. This is known as *CCSR (Causally Connected Self Representation)***.** The CCSR is often referred to as the *meta level*, and the system itself *the base level.* System and application code can use meta-interfaces to inspect internal configuration of the middleware reconfigure it to adapt to changes in the environment [3].

Context-aware reflective middleware actively measures the application interested contexts and adapts to them automatically and predicatively to meet the adaptability demands of distributed applications [2].

Making some aspects of the internal representation of the middleware explicit and hence accessible from the application, through a process is called ***reification***. The process where some aspects of the application are altered or overridden is called ***absorption***. In middleware platforms, two (complementary) styles of reflection have been used, namely structural and behavioural reflection.

Structural reflection is concerned with the underlying structure of objects or components, e.g., in terms of interfaces supported, another way, the ability to adapt the structure of an object (e.g., to add new behaviour at run-time). Meta-data or context can be viewed as a form of structural reflection, providing additional (meta) information about the underlying system, e.g. physical location, current battery levels or performance of the network.

Behavioural reflection is concerned with activity in the underlying system, e.g., in terms of the arrival and dispatching of invocations. Typical mechanisms provided include the use of interceptors or dynamic proxies that support the reification of the process of invocation and the subsequent insertion of pre or post- actions [3], [13].

4. Proposed System

The increasing use of mobile devices, such as mobile phones and personal digital assistants, has caused designers to find out many ways for end-users to use mobile applications that exploit flexible and convenient methods or techniques.

These devices have scare resources and exposed to variation of context. Context is the set of environmental states and settings that either determines an application's behavior or in which an application event occurs and is interesting to the user [17]. Contexts such as location, devices status, and bandwidth and network state frequently changes in mobile world. Therefore, mobile applications must be adaptable to these changing contexts.

To adjust these dynamic contexts, middleware platforms for mobile computing must be capable of both deployment-time configurability and run-time reconfigurability. In consequence, a reflective middleware is proposed for service adaptation in order to response to erratic execution contexts. In reflecting middleware behavior according to contexts, it can be seen few solutions for self-adapting although there are some attempts to preferred QoS of users. Therefore, end-users or application developers always specify their desired specification of applications.

The proposed middleware architecture allows the best possible quality of application based on available contexts. The overview architecture of the system is depicted in Figure 1. The system has five attributes to decide QoS of application_ network bandwidth, network latency, processor speed, memory and battery status. Mobile application is trained to the system to output suitable version of application to the user by applying these five attributes. For each attribute, the dependency of application on it is calculated by using Eq. (2). After evaluating every attributes, the most appropriate quality of application is determined by exploiting Eq. (1). Policies are defined after that. These steps take place in learning phases of server and are offline processes. In responding client requests, the server gives suitable version of application based on currently contexts of client (i.e. resources available of clients) with the

utilization of pre-stored polices. If the contexts of client's device change after running the application, it notify to the server and the server adapts the running application according to the changing contexts.

We will implement a prototype of our reflective middleware on android platform and Restaurant Finder application will be put into service as case study. eXtensible Markup Language (XML) is used to encode application profiles. XML is flexible meta-languages that can improve interactions between middleware and applications and its signification can be easily manipulated and understandable by both machines and humans.

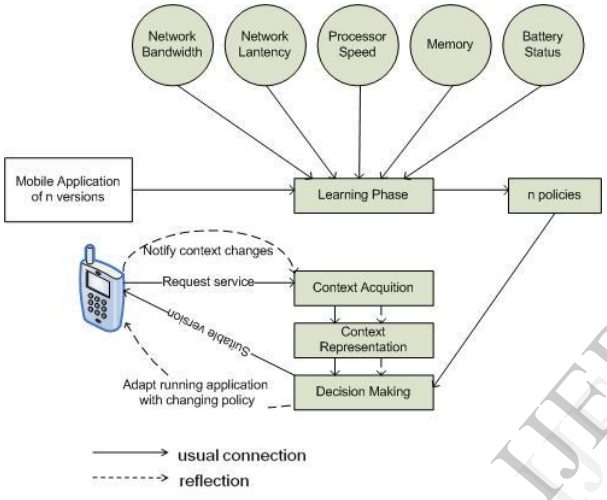


Figure 1. Overview of CARMM

5. Learn Rules with Locally Weighted Regression

Although there are many contexts that can affect mobile device behavior, in this place, we select network bandwidth, latency, processor speed, memory and battery level as contexts of our applications. These contexts are available for the system as parameters of locally weighted regression. As more than one context is required to predict QoS of application, we use multiple linear regression to approximate the appropriate quality.

$$E(Y|c) = \beta_0 + \beta_1 c_1 + \cdots + \beta_p c_p + e_i \qquad (1)$$

where β_0 is called the constant term and β_o to β_p are the coefficients relating to each context c. e_i is the error term. β_j is the importance of c_j on the application. Based on these function, we can estimate the expected quality of the application. Multiple linear regression is the linear combination of explainatory variables [14].

In above equation, For each c, it should have its own function f(c) to calculate its solely influence on application. Each parameter is inputted to the learning stage to discover the relationship of desired output and this parameter using simple linear function. Simple linear regression analysis brings into play to predict the value of the dependent variable *y*, given the value of the explanatory variable, x. In this model, we can write down the following form:

$$f(c) = \beta_0 + \beta_1 x_i + e_i \qquad (2)$$

where β_0 the *intercept* and β_1 is the *slope* of the line. For example, the higher the bandwidth, the higher the quality of image (i.e. the higher the value of xi, the higher value of $f(c)$ will be obtained).

In spite of concerning all resource usages such as battery, CPU loading, bandwidth and memory, some context are more important for some applications. For example, email application is more interesting in higher bandwidth rather than larger memory space. So the weights of these contexts are different and application must assign relevant weight or score for better result.

Server makes decision by matching calculated results of contexts and predefined rule. As an example,

Training weights of contexts are here:

weight for Bandwidth = 3, weight for CPU = 2, weight for Memory =1

c_1 = bandwidth, c_2 = CPU and c_3 = Memory

The result is $1*c_1+2*c2+3*c3=x$.

And in our Restaurant Finder application, there are three versions, says v_1, v_2 and v_3. If x is in the range of predefined policy 1, version1 (v_1) is sent to user.

p_1 = predefined range of policy1 (such as 0-20)

p_2 = predefined range of policy2

p_3 = predefined range of policy3

The procedure is as follow:

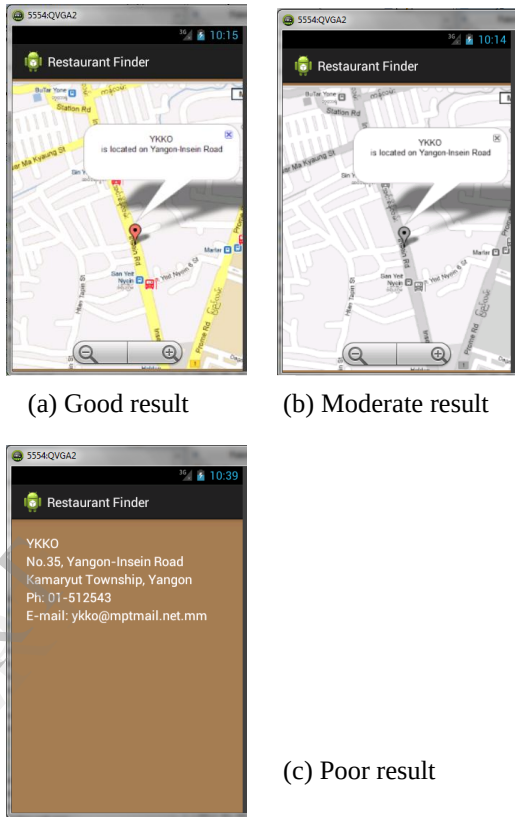
If ($\min(p_1)<x<\max(p_1)$)
    v_1 is sent to client's device.
Else If ($\min(p_2)<x<\max(p_2)$)
    v_2 is sent to client's device.
Else
    v_3 is sent to client's device.

The range values of p_1, p_2 and p_3 are categorized using locally weighted logistic regression as a classifier.

6. Case Study Application

In our Restaurant Finder application, users can find the restaurant he desire by the name of restaurant. The server can respond location of the restaurant with map view and some information of it. Server store application of three versions - full-colour map, gray- colour map and description texts only without photo as shown in Figure 2. When user invokes service from the server, it sends response the version convenient with the current context of the user's mobile phone. The server makes decision which policy should be adopted by matching predefined rules and acquired contexts' calculation. In Figure 2(a), the server give the result of good quality map in the time of client's resources maximize. However, in figure 2(b), the facility of application is not great because the client' contexts degrade. Figure 2(c) is the poorest result returned from server because of exhausting one or more resources of device.

As contexts change quite frequently, application must be fixed its own profile for all the time of changing contexts. Middleware must be provided an initial profile to be allowed the application to dynamic access. Applications can read their own profiles (introspection of middleware behaviour), and dynamically modify the meta-data that is encoded in the profile (adaptation of middleware behaviour) [8]. For example, in deciding to display map of restaurant, an application may alter the use of the 'FullColour' policy when it is assigned the time where bandwidth is high to 'GrayColour' policy when available bandwidth is considerable slow.



(a) Good result (b) Moderate result

(c) Poor result

Figure 2. Map results returned from the server

7. Conclusion

Due to the increasing popularity of mobile devices, end users desire their mobile applications to use more effectively and efficiently with the maximize use of their resources in mobile environment. Moreover, they want to experience stable and flexible applications. CARMM attempts to balance user's needs and resource constraints by using reflection and locally weighted regression. In addition, the system provides some other desirable properties, including simplicity, competitive accuracy, capability of extrapolating, and confidence interval because it utilizes logistic regression.

References

[1] C.Mascolo, L. Capra, W. Emmerich, *"Mobile Computing Middleware",* Advanced lectures on networking, pp. 20-58 Springer-Verlag New York, Inc., 2002.

[2] S.Liu, L.Cheng,*"A context-aware reflective middleware framework for distributed real-time and embedded systems",* Journal of Systems and Software, Volume 84, Issue 2, pp. 205-218, February 2011.

[3] P. Grace , G. Blair*, "Reflective Middleware"* , In Handbook of Mobile Middleware, A. Corradi and P. Bellavista eds. (invited book chapter), CRC Press, 2006.

[4] B. Smith. *"Reflection and Semantics in a Procedural Programming Language",* Phd thesis, MIT, Jan. 1982.

[5] Y. Yokote, *"The Apertos reflective operating system: The concept and its implementation"*, In Proceedings of OOPSLA'92, pp. 414-434. ACM Press, 1992.

[6] J. McAffer, *"Meta-level architecture support for distributed objects"*, In Proceedings of Reflection'96, pp. 39-62, San Francisco, 1996.

[7] F. Kon, F. Costa, *"The Case for Reflective Middleware",* Magazine of Communications of the ACM – Adaptive middleware, Volume 45 Issue 6, pp. 33-38, June 2002.

[8] L. Capra, *"Reflective Mobile Middleware for Context-Aware Applications",* PhD thesis, University of London, 2003.

[9] P. Grace, *"Overcoming Middleware Heterogeneity in Mobile Computing Applications"*, PhD thesis, Lancaster University, 2004.

[10] R. Rouvoy, P. Barone, Y. Ding, F. Eliassen, S.Hallsteinsen, J. Lorenzo, A. Mamelli, and U.Scholz, *"MUSIC: Middleware Support for Self-Adaptation in Ubiquitous and Service-Oriented Environments",* B.H.C. Cheng et al. (Eds.): Self-Adaptive Systems, LNCS 5525, pp. 164–182, 2009.

[11] J. Cao, N. Xing, A.T.S Chan, Y. Feng, B. Jin, *"Service Adaptation Using Fuzzy Theory in Context-aware Mobile Computing Middleware*",* Proceedings of the 11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'05), 2005.

[12] J. Khan, F. Shafiq, M.K. Khan, *"A Statistical Approach for the Assessment of QOS and Performance in Grid Computing Environment",* International Journal of Computer Applications (0975 – 8887) Volume 19– No.3, April 2011.

[13] L. Capra, G.S. Blair, C.Mascolo, W. Emmerich, P.Grace, *"Exploiting Reflection in Mobile Computing Middleware",* Mobile Computing and Communications Review, Volume 6, Number 4.

[14] CCSR: The Cathie Marsh Centre for Census and Survey Research, http://www.ccsr.ac.uk/publications/ teaching/ mlr.pdf, LastAccess-(9-1-2013).

[15] J. Malenfant, M. Jacques, F.-N. Demers, *"A Tutorial on Behavioral Reflection and its Implementation",* Reflection 96' Conference Proceedings, 1996.

[16] G. Chen, D.Kotz, *"A Survey of Context-Aware Mobile Computing Research",* Darmouth Computer Science Technical Report TR2000-381.