Dimensioning an OBS switch with partial wavelength conversion and fiber delay lines via a mean field model

Adaptation Strategies for Performance Failure Avoidance

Hong Sun[1], Vincenzo De Florio[1], Ning Gui[1], Raf Hens[2], Bert Vankeirsbilck[2], Bart Dhoedt[2], Chris Blondia[1]

1 *PATS Research Group*
University of Antwerp -IBBT
Middelheimlaan 1, 2020 Antwerp,
{hong.sun, vincenzo.deflorio, ning.gui, chris.blondia}
@ua.ac.be

2 Ghent University - IBBT
Gaston Crommenlaan 8, bus 201 B-9050 Ghent, Belgium
{raf.hens, bert.vankeirsbilck, bart.dhoedt}
@intec.UGent.be

Abstract*—Nowadays, technologies are providing the mobile terminals with much more powerful computational abilities. Such improvement made it possible to run many complex applications on mobile devices. However, many of these new applications are also resources demanding. Lacking sufficient resources would cause performance failures and impact negatively on the users' quality of experience. In order to improve this, it is important to provide the users with an easy access to specify their requirements. It is also crucial to monitor the system resources and make corresponding adaptation immediately according to the user's specifications. In this paper, we propose adaptation strategies that flexibly combine the process of monitoring and adaptation, and provide an easy way to specify user's requirements. By tuning the quality of service, we reduce the applications' demand on system resources, thus decreases the chances of performance failures and improving the users' quality of experience.*

Keywords-Adaptation, OSGi, failure avoidance, Quality of Experience, Aspect Oriented Programming

I. INTRODUCTION

With the development of technologies, the functions of the mobile terminals are becoming more and more powerful; the accesses to pervasive services are available in many areas. Users with powerful mobile terminals are granted with the ability to access information and multimedia services from almost anywhere at anytime.

However, there are still some restrictions impacting on the user's quality of experience (QoE) due to the limitation of the available resources: examples include a high CPU usage impacting the system's ability to support burst computation requirements; or an environment that is changing when the user is roaming, thus affecting the available bandwidth, etc. In order to increase the user's QoE and avoid performance failures (also known as late timing failures [1]), the mobile terminal should intelligently detect the changes of environment (battery capacity, bandwidth usage, CPU usage, etc.), and take proper adaptations promptly.

Many adaptation strategies are already developed in different domains, such as switching transcoding methods, or adjusting the tasks' priorities. However, adaptations are highly application-specific or domain-specific, which may bring conflicts between different application domains. For instance, in the power consumption domain, reducing the power could obtain a longer battery life, while in the communication domain, increasing the power may guarantee a better communication quality. Another feature of these application-specific adaptations is that they are normally hard bound with the application code and lack of flexibility to make reconfigurations in different environments.

Rather than scattering the adaptation logics in different applications and represent them as low-level binary code, architecture-based adaptation uses external models and mechanisms in a closed-loop control fashion to achieve various goals by monitoring and adapting system behavior across application domains. A well-accepted design principle in architecture-based management consists in using a component-based technology to develop management system and application structure [2][3][4]. However, in most of the above mentioned approaches, the configurations of applications and components are generally carried out before runtime; these systems still need to improve their ability to take runtime reconfigurations according to the change of system status.

In this paper, we are proposing an adaptation model to flexibly assemble the monitoring process of system resources, and provide an easy access for the users to specify their preferences. Adaptation logics are separated from applications and controlled by the system, while the reactions are still carried out on applications. Such a framework could flexibly implement, modify, and execute adaptation policies through users' preferences, thus helping avoid the performance failure and increase the users' QoE.

The rest of this paper is organized as follows: section 2 will introduce some design issues of the mobile adaptation frameworks. Our adaptation model will be presented in section 3. Experiments of such an adaptation model will be given in section 4, and conclusions will be made in section 5.

II. DESIGN ISSUES

Many design issues need to be considered in developing adaptation framework for mobile systems, among which, we deem the following aspects to be the most important ones:

When to take the adaptations:

Choices here include deploying the adaptation policies in the installation phase, and adaptations at execution time [5]. Our view is that if the adaptation schemes are solely configured in the setup time, they would lack of flexibility,

while if there are too many operations in the runtime, this will also introduce extra costs such as extra time to switch the policies, and make the system too complex to design. In our design, we set our adaptation policies in the deployment time, however, still provide our users with access to switch adaptation policies and modify adaptation parameters during the runtime. Such runtime modifications can be implemented in a simple way and without introducing an excessive amount of design complexity.

Where to carry out the adaptation:

Three models are widely used to take adaptations: centralized, application-transparent adaptation; decentralized, application-specific adaptation; and integrated model [6]. The centralized model performs adaptation at operating system level, which avoids the competition betweens applications, makes decisions concerning the best adaptation strategy on system level and aids efficiency in resource usage. However, the drawback is that lacking the knowledge from specific applications, adaptation actions are not efficient in such a model. The decentralized model performs adaptation solely within the application. Monitoring and adaptations are glued together in certain applications without any support from the system. Such a solution is effective to solve a particular adaptation facet, however, it is scarcely efficient system wide, and there might be conflicts between different decentralized adaptations. The integrated adaptation model takes the advantage of the two above mentioned approaches: it combines the system management of resources with the application specific knowledge of exactly what is required and how to execute the adaptations. The integrated approach is much more difficult to implement compared with the other two approaches: it requires especially tight cooperation between the applications and the operating system, and existing applications need to be modified to run effectively under such model. In our adaptation model, we use the integrated approach. Monitoring and applications are scattered in the system, however, the monitoring results, adaptation analysis and adaptation instructions are processed in a centralized way. We will explain how we execute the adaptations on applications in later sections.

How to carry out the adaptation:

There are many ways to take adaptations; however, they could be generally divided into two categories: the first one is adaptation of services, e.g. switching the coding method of a video encoder, etc. The second one is setting key parameters, e.g. resetting the frame dropping rate of a mobile video player, etc. In our previous research [7], we built up an adaptation framework combining adaptations on both services and parameters. However, the drawback here is that these two adaptations are carried out separately through SOA and RR variables [8]. In this paper, we will combine the Aspect Oriented Programming and SOA to integrate the above mentioned two approaches together.

III. ADAPTATION FRAMEWORK

In our previous work [7], we have proposed to use a so-called global adaptation framework to realize the adaptation. We deem that in order to make the correct adaptation in the mobile application, there is a stringent requirement for the system to be aware of the environmental changes. The proposed adaptation strategies are thus constructed as an event-condition-action model [9] [10]. The system is context aware, and adaptations are taken when the context changes. The event module detects the changes of the surrounding environment. The detected events trigger the reasoning of the rule engine; depending on the conditions, specific adaptation rules will be fired and the adaptations will be held in the action module.

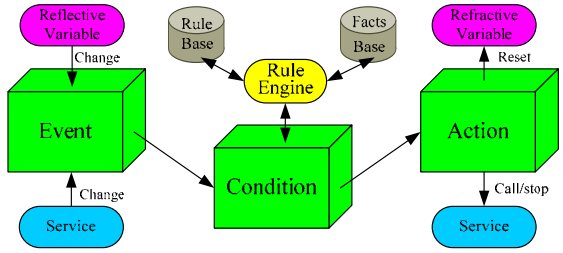

Figure 1. Architecture of Global Adaptation Framework [7]

In Fig. 1, the event module is designed to monitor the environment around the mobile terminal, availability of applications and the system status of the mobile terminal. Services and reflective variables are used to separate different kinds of information. Services are developed as OSGi bundles [11], and changes of the services are detected by the service listener. In order to better indicate the system status, so-called “reflective variables” [8] are used to represent some of the key system state parameters. Adaptation decisions are made in the condition module, where certain rule engines can be implemented, and adaptation decisions are to be deduced based on predefined rules. The action module in the framework executes the adaptations when certain adaptation rules are fired in the condition module. Adaptations in the action module are executed as adjustments on service, such as start/stop/update OSGi bundles; or by setting some system parameters, which is done through so-called “refractive variables” [8].

The above mentioned solution provides an agile way to detect the change of context by the combination of service change detection and the reflective/refractive variables. It also provides fine grained adaptation by the combination of changing services and resetting the refractive variables. The drawback is that firstly, the monitoring and adaptations of services and variables are processed separately by SOA and RR variables, which increases system complexity and reduces efficiency; and secondly, the adaptation policies are all stored as predefined rules in deployment, which lacks of flexibility to carry out runtime changes.

The adaptation strategies we propose in this paper are developed from the structure shown in Fig. 1, however, the above mentioned drawbacks are solved in our proposed

adaptation strategies here. The adaptation model that we propose in this paper is shown in Fig.3.

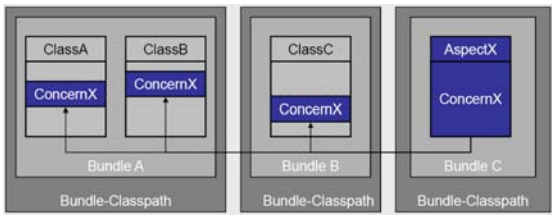

Figure 2. Aspect Weaving for OSGi [13]

The adaptation framework is developed based on the Equinox Aspects [12]. The Equinox Aspects approach combines the advantage of Aspect Oriented Programming (AOP) with those of Service Oriented Architectures. In such a solution, AOP help separate the cross-cutting concerns and take corresponding reactions as “advices”; while the SOA is able to flexibly organize applications as services. Using a load-time weaving extension, AspectJ aspects can be added to bundle-based system just by including them into general OSGi bundles. Figure 2 shows that crosscutting concerns can be expressed between different bundles [13].

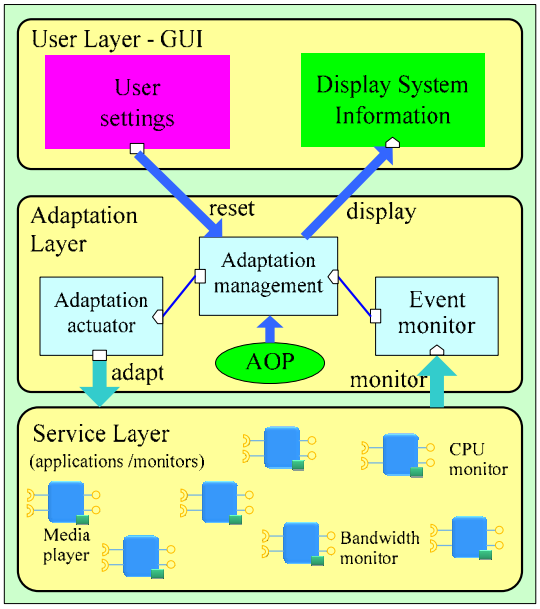

Figure 3. Context Aware Global Adaptation Framework

Figure 3 shows our context aware adaptation framework. Such framework is divided into three layers. In the service layer, applications are wrapped as services to ease adaptations in the later stage. Applications such as media players or different types of monitors are wrapped as service bundles in service oriented approach, or more specifically, with the OSGi Framework.

In the adaptation layer, the event monitor module collects the detected changes from different monitor bundles. Information of the interested resources are collected system-widely: for instance, the value of CPU usage and bandwidth usage are reported by the CPU monitor and bandwidth monitor separately, however, contrarily from the application-specific approach, these monitored values will be collected together by the event monitor module to analyze and trigger the adaptations.

List 1. A Simple Example of Monitor and Adaptation

```
pointcut CPUMonitor(float value) :
    call(float DisplayCPU(float))
    && args(value);

after(float value) : CPUMonitor(value) {
    /*
    Advice, specify adaptation policy here
    */
}
```

Adaptation decisions are deduced in the adaptation management module, based on the adaptation policies and the system resources. Aspect Oriented Programming (AOP) is adopted to express adaptations based on the monitored changes. The relationship between monitoring and adaptation in AOP can be expressed as simplified in List 1. The execution of adaptations is symbolized as advices in List 1, and could be carried out as switching services, or resetting parameters. In real-life applications, the expression of adaptation logics could be very complex, and such adaptations will be executed by the adaptation actuator module.

In order to improve the user’s QoE, we add a user layer to our adaptation framework. In the user layer, we created a graphic user interface for our adaptation framework. The GUI is connected to the adaptation management module in the adaptation layer; the system information received by the monitor module is sent and displayed on the GUI. Most importantly, the GUI provides the users with an access to switching between the predefined adaptation policies, or resetting some key parameters, such as applications priorities, during the runtime. An instance of the GUI is shown in Figure 5, which is used and further explained in the experiment of section.

IV. Experiment

The following application is set up as an experiment to validate how our proposed adaptation framework could monitor the system information and guide the adaptations on video playing.

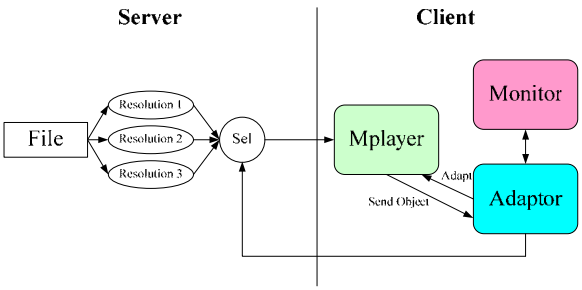

Figure 4. Experiment Structure

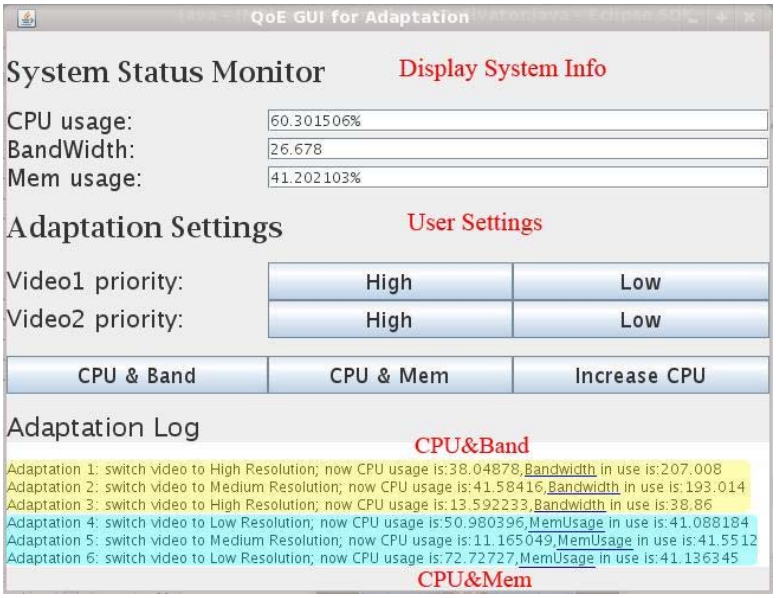

Figure 5. Graphic User Interface

The structure of the experiment is shown in Fig. 4. A videoclip is played by the Mplayer (an open source media player) in the client side as an application. The sources of the displayed videos are stored in the server side, and the server provides the videos with three different resolutions upon decisions from the client side. The monitor module in the client side monitors the information of CPU usage, bandwidth, and memory exchange rate; such information is displayed in the GUI in Fig. 5. The adaptor bundle retrieves the system information from the monitor bundle; meanwhile, it also retrieves MPlayer objects, which describe and control the video that is being played by the MPlayer. According to the system status, the adaptor bundle takes corresponding adaptations on the Mplayer objects, and selects the appropriate resolutions. When the system is meeting restrictions on resources, such as low CPU, or limited bandwidth, etc., it is still possible to avoid performance failures by trading off quality of service that least affects QoE. In this case performance failures are avoided by reducing the resolution of the videos.

The thresholds that we defined to take the adaptations are shown in Fig. 6. There is no particular reason to choose these figures, which are used here only for the proof of concept of our adaptation framework. There are two adaptation policies in our experiment: either consider the combined value of CPU and Bandwidth usage, or consider the combined value of CPU and Memory usage. Users have to choose one of these policies; it is also possible for user to change the adaptation policies in the runtime through the GUI (see the User settings buttons in Fig 5.). In the adaptation log in Fig. 5, it is shown that the adaptations are woven by the change of the values on CPU and bandwidth at the beginning, while later on we switched to adaptations based on the combination of CPU and memory usage.

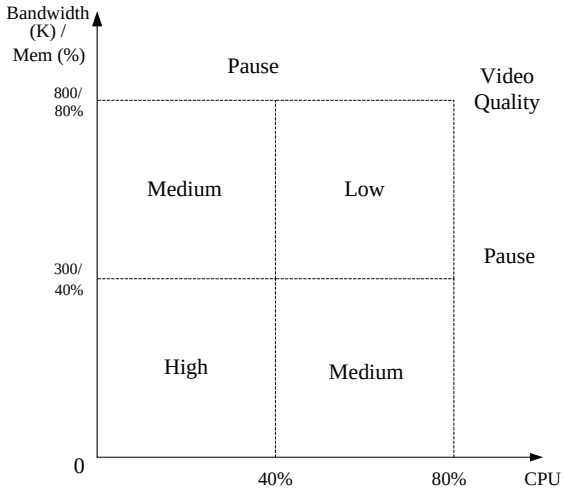

Figure 6. Adaptation Threshold

The results of the experiments proved that our proposed adaptation framework could flexibly organize the adaptations, while also being able to meet the user's requirements, and even change adaptation policies at runtime. Thus it could best meet the users' QoE and reduce the performance failure in mobile computations. In our demonstration, the video could be broadcast with minimal impact of quality of experience by reducing resolutions in stringent computation environment.

Our demo is worked out as a proof of concept to show that our adaptation framework is able to switch adaptation policies at runtime and it is also capable to orchestrate adaptations on different applications. Thus the adaptation policies that we listed above are just a proof of concept of our adaptation framework – finding efficient adaptation policies is out of the research scope of this paper.

V. CONCLUSION

This paper proposed a framework to organize adaptation in mobile terminals with the design goal of improving the user's QoE and reducing the performance failures. The framework is organized as event-condition-action model, and flexibly allows adaptations acting upon the changes of environment context. By coupling the AOP and SoA in an OSGi environment, the architecture proposed in this paper achieves the following improvements:

- -The proposed architecture provides a flexible way to detect the changes and take adaptations. Changes and adaptations include modifications of both services and parameters; however, both are treated in a unified way.
- -Through the GUI of our proposed architecture, users can easily discover system changes, and more importantly, they are able to change the adaptation policies at runtime, thus best improving the users' QoE.

Further work will be carried out on embedding such a framework in the thin-client application architecture [14] to assist the adaptation at the client's side. Applications of such an adaptation framework in homecare environment will be investigated aiming at constructing safety ambient assisted living environments for the elderly people in their surrounding environments.

ACKNOWLEDGMENT

The authors would like to thank IBBT for supporting this work, and also our colleagues in the "End-to-End Quality of Experiences[1]" project for their cooperation.

REFERENCES

[1] F. Cristian, "Understanding Fault-Tolerant Distributed Systems", Communications of the ACM vol.34 no.2, February 1991, pp.56-78.

[2] F. Kon, F. Costa, G. Blair and R. Campbell. The case for reflective middleware: building middleware that is flexible, reconfigurable, and yet simple to use. Communications of the ACM (CACM), Volume 45, Issue 6, 2002, 33-38.

[3] S. Sylvain, B. Fabienne, and P. Noel De, "Using components for architecture-based management: the self-repair case," in Proceedings of the 30th international conference on Software engineering, Germany,2008.

[4] P. Costa, et al. "The RUNES middleware for networked embedded systems and its application in a disaster management scenario," 5th Annual IEEE Int. Conf. on Pervasive Computing and Communications, 2007.

[5] N. Houssos, V. Gazis and N. Alonistioti, A Novel Mechanism for Mobile Service Adaptation, In 57th IEEE Vehicular Technology Conference (VTC 2003 Fall), 2003.

[6] T. Edmonds, A. Hopper, S. Hodges, Pervasive Adaptation for Mobile Computing, in proceedings of 15th International Conference on Information Networking (ICOIN'01), 2001.

[7] H. Sun, V. De Florio, N. Gui & C. Blondia, Global Adaptation Framework for Quality of Experience of Mobile Services, in proceedings of the 2007 IEEE Three-Rivers Workshop on Soft Computing in Industrial Applications, 2007

[8] V. De Florio & C. Blondia. Reflective and Refractive Variables: A Model for Effective and Maintainable Adaptive-and-Dependable Software, in the Proceedings of the 33rd Euromicro Conference on Software Engineering and Advanced Applications (SEEA 2007), Luebeck, Germany, 2007.

[9] R. Etter, P. Costa and T. Broens. *A Rule-Based Approach Towards Context-Aware User Notification Services.* In: Proceedings of the IEEE International Conference on Pervasive Services 2006, 26-29 Jun 2006, Lyon, France. pp. 281-284.

[10] V. De Florio & C. Blondia. A System Structure for Adaptive Mobile Applications, In Proceedings of the IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2005 (WOWMOM 2005), Giardini Naxos, Italy, 2005.

[11] Open Service Gateway initiative (OSGI), OSGI Service Platform Version 4, http://www.osgi.org, May, 2007.

[12] E quinox Aspects Project:

http://www.eclipse.org/equinox/incubator/aspects, 2009

[13] M. Lippert**,** Aspect weaving for OSGi, in proceedings of Conference on Object Oriented Programming Systems Languages and Applications, Companion to the 23rd ACM SIGPLAN conference on Object-oriented programming systems languages and applications, 2008.

[14] P. Simoens, L. Deboosere, D. De Winter, F. De Turck, B. Dhoedt, P. Demeester: Optimization Models for Application Migration to Support Mobile Thin Clients. EuroNGI Workshop 2006.

[1] Official website of End-to-end Quality of Experience: https://projects.ibbt.be/qoe/