Architectural Ideas for the Support of Adaptive Context-Aware Applications

Keith Cheverst, Christos Efstratiou, Nigel Davies and Adrian Friday
Distributed Multimedia Research Group, Lancaster University,
e-mail: kc@comp.lancs.ac.uk

1. INTRODUCTION

Mobile applications are required to operate in environments characterised by change. More specifically, the availability of resources and services may change significantly during a typical period of system operation. As a consequence, adaptive mobile applications need to be capable of adapting to these changes to ensure they offer the best possible level of service to the user. Our experiences of developing and evaluating adaptive context-aware applications in mobile environments has led us to believe that existing architectures fail to provide the necessary support for such applications. Moreover, we believe that current research in this area is also failing to address the core requirements of this class of application, i.e. the need to support coordinated action between application and system components and the resolution of conflicts caused by the need to adapt to multiple contextual triggers.

Current adaptive mobile applications are built using one of two approaches: either the adaptation is performed by the system which underpins the application (in an attempt to make transparent the effects of mobility) or the application itself monitors and adapts to change. In some cases, these approaches are combined as, for example, in the MOST system [1] where the middleware platform adapts the operation of the network protocol in the face of changes in QoS and, additionally, reports these changes to the application to enable application level adaptation.

An analysis of current approaches to supporting adaptation reveals two important facts. Firstly, support for adaptation is often fragmented with a range of mechanisms being used to notify applications of changes in different environmental and contextual attributes [2]. Secondly there is a clear, yet incomplete, set of directed information flows within the system. More specifically, applications typically notify the system of their requirements and invoke internal (to the application) mechanisms to achieve their objectives. Correspondingly, the system is usually able to notify the application of changes in attributes that are of relevance to the application. However, the authors believe that these information flows are not sufficient. An additional flow is required, that of control messages from the system to the applications. In this paper, we explore the need for a unified architecture which can support multiple contextual attributes coupled with a control flow mechanism.

2. DRAWBACKS OF CURRENT APPROACHES

Mobile systems need to be capable of adapting to a wide range of attributes such as network bandwidth, location, power etc. In general, current mobile systems provide support for adaptive applications by notifying applications when certain 'interesting' changes in attributes occur, e.g. when bandwidth falls below some specified minimum level. It is then the responsibility of the application to adapt in an appropriate way, e.g. by reducing its bandwidth requirements. However, this approach can lead to inefficient solutions because of the lack of support for enabling coordination between the adaptation policies of multiple applications that may co-exist on the same system. In addition, when using this approach in isolation, there can be insufficient control over the implications of having multiple, and possibly conflicting, attributes triggering adaptation. In the following scenarios, we illustrate the kind of problems that can occur when relying upon a simplistic notification based approach and isolated, uncoordinated, application adaptation.

2.1 The Need for Coordinated Application Adaptation for Power Management

This scenario illustrates the need for coordination in order to achieve efficient power management on a mobile system. One existing approach for handling power management, i.e. the ACPI model, is to enable the operating system to switch hardware resources into low power mode when not in use, e.g. spinning down the hard-disk. This approach requires that applications leave hardware resources in an idle state for sufficient periods of time to make the transition between idle and active states worthwhile. Although this approach is suitable when only one application is running on a mobile device, the approach can prove ineffective when multiple applications or system services are sharing hardware resources. In more detail, the lack of coordinated access to hardware resources can result in poor utilisation of the shared resource and therefore sub-optimum power management. For example, consider the case of multiple applications that implement an auto-save feature. In the absence of any coordination between applications each application may choose to checkpoint its state to the disk at an arbitrary time, without considering the state of the disk (i.e. spinning or sleeping). In contrast, if applications are able to coordinate their access to the hard-disk then access to the disk can be clustered, allowing longer periods of inactivity. This latter approach is clearly more power efficient than the situation in which usage of the hard-disk is completely arbitrary and uncoordinated.

2.2 The Problem of Conflicting Adaptation

In this scenario, we illustrate the potential problems that can occur in a system that utilises separate adaptation mechanisms for different attributes. We consider a hypothetical mobile system that utilises two independent adaptation mechanisms, one for managing power and the other for managing network bandwidth. The two mechanisms can conflict with one another as the following example illustrates. If the system needs to reduce power consumption, the power management mechanism will request those applications that are utilising network bandwidth to postpone their usage of the network device in order to place the network device in sleep mode. As a consequence of applications postponing their use of the network, the available network bandwidth increases. However, the network adaptation mechanism will detect this unused bandwidth and notify applications to utilise the spare bandwidth. In this way, the request to utilise available bandwidth is in direct conflict with the request to postpone network usage. In this example if conserving power was the system's primary goal then the instruction given to applications to utilise more bandwidth should have been withheld. Clearly appropriate platform support needs to enable coordination or harmonisation in order to detect and avoid potentially conflicting adaptation mechanisms.

2.3 Using Remote Services

In this scenario, we consider a client-proxy-server example (e.g. web browsing) in which the proxy object is required to adapt its behaviour based on the context of the client. In this example, the client communicates with the proxy using a wireless link but communication between the proxy and the server is via wired networks. Traditionally, where the bandwidth between the client and the proxy is limited, a common solution is to compress the data at the proxy and then perform decompression at the client. However, the limited processing power on the mobile device constrains the extent to which processor-intensive decompression can occur on the client and therefore the potential for reducing the bandwidth of the transmitted stream. Furthermore, this implies that the processing demands of other applications on the mobile host has a direct impact on the potential for reducing bandwidth requirements. Consequently, if gaining a reduction in the bandwidth requirements of the wireless link has a high priority then this could be achieved by maximising the available processing power for decompression on the client, possibly by reducing the allocation of processing time to other processes.

An additional complication to this scenario is based on the fact that the client will not necessarily have sufficient information to determine whether or not the use of a proxy object is the appropriate solution for dealing with the problem, i.e. a perceived lack of network bandwidth. Indeed, the actual cause behind the perceived low network bandwidth could be due to a problem at the server or on the wired network. In either of these cases, the interposition of a proxy object would not help address the bandwidth problem and would in fact add further delay into the system. This example highlights the desirability of adopting an end-to-end approach towards adaptation and, as such, requires the monitoring of all components involved in the interaction.

2.4 Utilising Alternative Location Sensing Mechanisms

This scenario considers the case of supporting multiple services for providing similar contextual information. In this case, we consider a location aware system that is capable of sensing its current location through two different mechanisms: a local GPS device and via beaconing in a cell-based wireless network. Using the latter mechanism, the system can identify the current cell in which it operates and thus specify its location. Although both mechanisms provide the same contextual information they have significantly different characteristics. In more detail, the GPS mechanism is typically the more accurate (with accuracy in the region of 5m) but does require extra power to operate. Alternatively, the network-based solution is generally less accurate (depending on the size of a cell, e.g. approximately 200m for WaveLAN) but has little or no addition power consumption requirements given that the network device is already in use by the system. It follows that if accurate location information was required (and concern over additional power usage was not an important issue) then the mobile system would select to use the GPS based solution. Alternatively, if the lifetime of the system's batteries (and therefore operation) was more important than achieving greater location accuracy then the network location mechanism would be more appropriate. Should more than one application on the host require location information, then it would be preferable if they all utilised the same mechanism, so that resources would not be wasted.

The adaptive strategy that would be most appropriate depends on both the user's requirements and the context of other attributes, such as power. In order for the system to make such decisions there is a basic requirement for system-wide adaptation policies. Without such policies, coordinated adaptation on the use of alternative context retrieval mechanisms is difficult because each application relies on a single mechanism without being able to identify the implications of its operation on other system resources and, consequently, other applications.

3. ANALYSIS

Current context-aware applications handle context in an improvised fashion with no support for achieving coordinated adaptation to context changes. Our belief is that any general platform for supporting context-aware application needs to be capable of addressing the problems of coordinated adaptation.

The situation is complicated still further by the fact that the adaptive behaviour triggered by one attribute can cause side-effects on other attributes. These side-effects could, in-turn, trigger adaptation requests to other applications that result in conflicting actions (as illustrated in the conflicting adaptation scenario in section 2.2). Moreover, current research [2,3] has identified the need to provide adaptation solutions based on the combination of different attributes. However, existing architectures do not provide the necessary support to enable programmers to construct applications that can adapt to multiple attributes and both identify and cope with conflicts in adaptation strategies. Instead, current mobile systems that provide support for adaptive applications tend to rely heavily on integrating QoS feedback and adaptation with network bindings. Examining the architecture of such systems allows us to identify a framework for analysing the architectural model of existing adaptive systems. The framework comprises two layers, the upper application layer and the lower representing the adaptation support platform. Between these two layers we can identify four distinct flows of control and information (see figure 1).

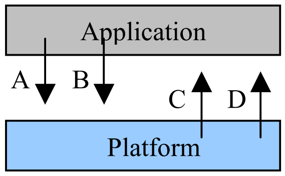

Figure 1: Directed flows between applications and platform.

Flow A: Represents the requirements set by the application, e.g. the network QoS requirements, concerning the resources or attributes supported by the underlying infrastructure.

Flow B: Represents the ability of the application to control the functionality of the underlying infrastructure. In the case of accessing a GPS device this could represent, for example, the control of the device by the application.

Flow C: Represents an information flow from the platform to the application. This could be used, for example, as a notification mechanism to inform the application when certain requirements cannot be met.

Flow D: Represents the ability of the underlying platform to actually control the operation of the application. More specifically, this flow represents an explicit request from the system for the application to perform a specific adaptive behaviour. For example, the application might be requested to reduce its demand for network bandwidth or disk usage.

Consideration of this framework enables a classification of current systems according to the types of flows supported. For example, network based adaptive systems such as Odyssey [5] and MOST [1] support flows A and C. Alternatively, Context-aware applications such as GUIDE [4] are based on flows B and C where flow B represents access to context-sensors and flow C represents information flowing from the sensors to the application.

4. CONCLUSIONS

The key conclusion of this paper is that the information flows supported by current platforms are insufficient. Indeed, an additional flow, that of control messages from the system to applications, is required to enable the development of a more unified architecture, capable of supporting coordinated and system wide adaptation based on multiple contextual attributes. In more detail, coordination is required in order to (1) avoid conflicts between the adaptation policies of multiple applications that may co-exist on the same system and (2) make optimum use of available resources.

References

1. Friday A., N. Davies, G. Blair and K. Cheverst. "Developing Adaptive Applications: The MOST Experience". Journal of Integrated Computer-Aided Engineering, 6(2), pp 143- 157.

2. Davies N., A. Friday, S. Wade and G. Blair. "L^{2}imbo: A Distributed Systems Platform for Mobile Computing". ACM Mobile Networks and Applications, Special Issue on Protocols and Software Paradigms of Mobile Networks, 3(2), pp 143-156, 1998.

3. Flinn J. and M. Satyanarayanan. "PowerScope: A Tool for Profiling the Energy Usage of Mobile Applications". In Proceedings of the Second IEEE Workshop on Mobile Computing Systems and Applications, 1999.

4. Cheverst, K., N. Davies, K. Mitchell and A. Friday, "Experiences of Developing and Deploying a Context-Aware Tourist Guide: The GUIDE Project". To appear in Proc. of MOBICOM'2000, Boston, ACM Press., 2000.

5. Noble B., M. Satyanarayanan, D. Narayanan, J. E. Tilton, J. Flinn and K. Walker. "Agile Application-Aware Adaptation for Mobility", In Proceedings of the 16th ACM Symposium on Operating System Principles, 1997.