University of Kent

Kent Academic Repository

Efstratiou, Christos, Cheverst, Keith, Davies, Nigel and Friday, Adrian (2000) ***Architectural Requirements for the Effective Support of Adaptive Mobile Applications.*** **In: IFIP/ACM International Conference on Distributed Systems Platforms and Open Distributed Processing (Middleware 2000). . pp. 15-26. Elsevier, Middleware**

Downloaded from

https://kar.kent.ac.uk/38867/ The University of Kent's Academic Repository KAR

The version of record is available from

This document version

UNSPECIFIED

DOI for this version

Licence for this version

UNSPECIFIED

Additional information

Versions of research works

Versions of Record

If this version is the version of record, it is the same as the published version available on the publisher's web site. Cite as the published version.

Author Accepted Manuscripts

If this document is identified as the Author Accepted Manuscript it is the version after peer review but before type setting, copy editing or publisher branding. Cite as Surname, Initial. (Year) 'Title of article'. To be published in *Title of Journal* , Volume and issue numbers [peer-reviewed accepted version]. Available at: DOI or URL (Accessed: date).

Enquiries

If you have questions about this document contact ResearchSupport@kent.ac.uk. Please include the URL of the record in KAR. If you believe that your, or a third party's rights have been compromised through this document please see our Take Down policy (available from https://www.kent.ac.uk/guides/kar-the-kent-academic-repository#policies).

ARCHITECTURAL REQUIREMENTS FOR THE EFFECTIVE SUPPORT OF ADAPTIVE MOBILE APPLICATIONS

Christos Efstratiou, Keith Cheverst, Nigel Davies and Adrian Friday

Distributed Multimedia Research Group,
Department of Computing,
Lancaster University,
Bailrigg,
Lancaster,
LA1 4YR
U.K.

telephone: +44 (0)1524 594462
e-mail: most@comp.lancs.ac.uk

ABSTRACT

Mobile applications are required to operate in environments that change. Specifically, the availability of resources and services may change significantly during typical system operation. As a consequence, 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 adaptive applications have led us to believe that existing middleware fails 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 adaptive mobile systems. In this paper we present a set of requirements for future mobile middleware which have been derived by considering the shortcomings of existing approaches and the needs of application developers. Key among these requirements is 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. The paper concludes with the presentation of an architectural framework within which middleware researchers can deploy solutions to the problems identified.

Full paper submission

ARCHITECTURAL REQUIREMENTS FOR THE EFFECTIVE SUPPORT OF ADAPTIVE MOBILE APPLICATIONS

ABSTRACT

Mobile applications are required to operate in environments that change. Specifically, the availability of resources and services may change significantly during typical system operation. As a consequence, 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 adaptive applications have led us to believe that existing middleware fails 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 adaptive mobile systems. In this paper we present a set of requirements for future mobile middleware which have been derived by considering the shortcomings of existing approaches and the needs of application developers. Key among these requirements is 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. The paper concludes with the presentation of an architectural framework within which middleware researchers can deploy solutions to the problems identified.

1. INTRODUCTION

Mobile applications are required to operate in environments that change. Specifically, the availability of resources and services may change significantly and frequently during typical system operation [10, 12]. As a consequence, mobile applications need to be capable of adapting to these changes to ensure they offer the best possible level of service to the user [12]. For example, many mobile web browser applications use a range of techniques to adjust their network requirements to meet available levels of service. While early research focused on applications which adapted to changes in network characteristics, there is now increasing interest in applications that adapt to general environmental and contextual triggers such as changes in a system's physical location, e.g. the GUIDE system [5] which supplies users with information tailored to their current location.

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 [10] 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. In the general case, it has been demonstrated that maintaining transparency in the face of mobility is not practical and that it is difficult for a system to adapt without support from the application.

Careful examination 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 [6]. Secondly, in all cases, 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 document, we explore the requirements for a unified architecture which supports multiple contextual attributes coupled with a control flow mechanism. The benefits of such an approach are clearly illustrated using numerous real-world examples.

Section 2 illustrates the shortcomings of existing mobile middleware by using a number of example scenarios. The shortcomings identified in these scenarios are then generalised in section 3 to produce a set of requirements for future mobile middleware. These generalised requirements are further refined into a set of architectural requirements in section 4. A framework for developing components which address these requirements is presented in section 5 and section 6 contains our concluding remarks.

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 middleware platforms provide support for adaptive applications by notifying applications when certain 'interesting' changes in attributes occur, e.g. bandwidth falls below some specified minimum level. It is then the responsibility of the application to adapt to these notifications in an appropriate way, e.g. by reducing its bandwidth requirements. However, this approach can be shown to lead to inefficient solutions because of the lack of support for enabling coordination between the adaptation policies of multiple applications. In addition, there is insufficient control over the implications of having multiple, and possibly conflicting, attributes triggering adaptation.

In the following scenarios we illustrate a range of problems which can occur when the above approach is adopted.

2.1 Scenarios

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 [1] model, is to enable the operating system to switch hardware resources into low power mode when not in use. For example, the hard-disk can be requested to spin down. 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 inefficient 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 which implement an auto-save feature. In the absence of any coordination between applications each application will 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.

Conflicting Adaptation

In this scenario we illustrate the potential problems that can occur in a system that utilises separate adaptation mechanisms for different attributes. In more detail, we consider a hypothetical mobile system which hosts two independent adaptation mechanisms, i.e. one for managing power and the other for managing network bandwidth. However, 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.

This example highlights the problem of relying on independent and uncoordinated adaptation mechanisms. Coordination or harmonisation is clearly required between adaptation mechanisms and in the example presented this could have prevented the conflict from occurring. More specifically, the instruction to applications to utilise more bandwidth could be withheld if, for example, conserving power was the system's primary goal.

Using Remote Scenarios

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 (see figure 1). 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.

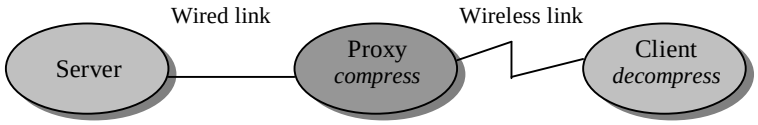

Figure 1: Use of a proxy to reduce bandwidth requirements over a wireless link.

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 constraints 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 reducing the requirement for bandwidth over the wireless link is 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. In addition, the level of compression performed on the proxy should correspond to the decompression capability and priorities of the client. For example, in figure 2 the client has an upper bound on both the bandwidth available for transmission and the processing power which may be allocated to the decompression process. The graph illustrates that this provides a range of possible compression rates which can be negotiated with the proxy. In particular, there is little point in the proxy compressing beyond the capabilities of the client or in compressing insufficiently to enable timely transmission over the network.

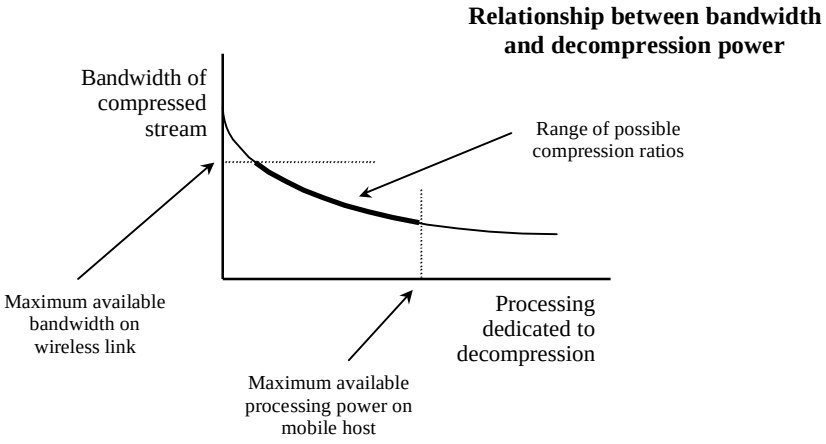

Figure 2: Finding the best compression rate according to the processing load of the client.

An additional complication to this scenario is 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 these cases, the introduction of a proxy object would be entirely the wrong solution and would in fact add further delay into the system. This example highlights the desirability of adopting an end-to-end approach, which, in turn, requires the monitoring of all components involved in the interaction.

Sharing Demand for Network Bandwidth: Web Browsing and Viewing a Video Stream

This scenario considers the problem of supporting two applications on a single mobile host which both make conflicting demands on network bandwidth, e.g. an adaptive web browser application and an application for viewing a video stream. In this case, we consider how the two adaptive applications might adapt to changes in the quality of the underlying network. In particular, following a drop in available bandwidth the two applications could react in a variety of ways, such as:

i. The web browser could stop downloading in order to dedicate its portion of bandwidth to the other application. This reaction would be appropriate if maintaining the presentation of the video stream had a high priority and if a minimum level of throughput was required in order to view the stream at the required frame rate.

ii. Both applications could adapt and share the available bandwidth equally. This strategy could be used where no prioritisation has been specified between the two applications.

iii. The video stream viewing application could adapt by reducing its bandwidth requirement, e.g. by reducing its frame rate, in order to enable the web browser to utilise a greater share of the available bandwidth. This approach could be taken if, for example, the web browser needed to perform an important download.

The reaction that would be most appropriate depends on both the user's requirements and the context of other attributes, such as total network bandwidth. In order for the two browsers to adapt in a coordinated manner there is a basic requirement for *system-wide adaptation policies*. Without such policies, coordinated adaptation between applications is difficult because each application will only be capable of independent adaptation, i.e. adaptation that does not take into account the implications of its adaptation on other system resources and, consequently, other applications.

3. ANALYSIS

The previous scenarios illustrated a number of potential problems with current approaches to developing adaptive mobile systems. In this section, we generalise on these findings to present a critique of existing mobile middleware and its role in supporting adaptive applications.

3.1 Architectural Model

In order to provide a framework for analysing the architectural model of existing mobile systems we consider the flows of information and control between the application and the underlying middleware (figure 3). The framework comprises two layers, the upper application layer and the lower layer representing system/middleware support. Between these two layers we can identify four distinct flows of control and information.

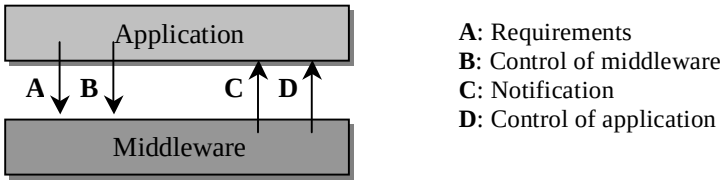

Figure 3: Directed flows between application and middleware.

Flow A: Represents the requirements set by the application concerning the resources or attributes supported by the underlying infrastructure. For example, in the case of network adaptation this flow could represent the application's network QoS requirements.

Flow B: Represents the ability of the application to control the functionality of the underlying infrastructure. In the case of reflective middleware this could represent, for example, the control of the middleware using an appropriate meta-protocol.

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. Such notification could then trigger the application to adapt.

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 middleware layer 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.

It is important to emphasise the difference between flows C and D. In more detail, flow C is a flow of information and therefore could become a trigger for adaptive behaviour. However, unlike flow D, it does not explicitly *require* the application to adapt. In contrast, flow D expects the application to react and then return the results of this reaction, e.g. the affect on other attributes such as power, network bandwidth etc.

Consideration of this framework enables a classification of current systems according to the types of flows supported. For example, network based middleware systems such as BAYOU [22], Odyssey [18], MOST [10] and Rover [11] support flows B and C. In this case the application specifies QoS requirements for the network channel and the underlying platform tries to achieve these requirements. However, if this is not possible then applications are notified in order to enable adaptation to take place.

Reflective middleware platforms such as Adapt [8] tend to support flow A in addition to flows B and C. The application is thus able to alter the functionality of the platform according to its needs via the middleware's meta-level interface. Application adaptation can be performed with the use of notifications from the underlying infrastructure.

Context-aware applications like GUIDE [5], Stick-e Notes [19] and Cyberguide [15] are based only on flow C. The underlying platform or device provides the contextual information to the application and the application is responsible for adapting to the change of the context.

According to our knowledge the only middleware platform that provides a flow of control from the platform to the application, i.e. flow D, is ISIS-META [17] though it should be noted that only a small amount of research has been undertaken into making ISIS operate in a mobile environment [4]. In ISIS-META the platform is able to explicitly request a specific behaviour from the application through the use of call-back functions called *actuators*. Flows A and C still exist in the system where the application specifies requirements to the platform and receives information about the state of the underlying network.

3.2 Multiple Attributes

Future mobile systems will need to be capable of adapting to a large and heterogeneous set of attributes. To facilitate this process, future middleware platforms will need to provide support for these attributes within a common framework. More specifically, while current mobile middleware platforms tend to rely heavily on integrating QoS feedback and adaptation with network bindings, future systems will need to integrate these parameters with attributes such as location.

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 2.1). Moreover, current research [6, 7, 9, 13, 14] has identified the need to provide adaptation solutions based on the combination of different attributes.

Current systems simply do not provide support for programmers that enables them to construct applications which adapt to multiple attributes and to identify and cope with conflicts in adaptation strategies.

4. ARCHITECTURAL REQUIREMENTS

The previous sections have demonstrated the limitations of current approaches for supporting adaptive mobile applications. In particular, these approaches lack appropriate support for enabling applications to adapt to numerous different attributes in an efficient and coordinated way. A new approach is therefore required which provides a common space for the coordinated, system-wide interaction between adaptive applications and the complete set of attributes that could be used to trigger adaptation.

This section considers a set of requirements that could be used to develop an appropriate architecture for supporting adaptive mobile applications.

Supporting a Common Space for an Extensible Set of Attributes

The first key requirement of the architecture is to provide a common space for handling the adaptation attributes used by the system. It is important that new attributes can be introduced into the system as and when they become important, e.g. the cost of specific services for mobile users or information about human physiology for wearable computers. The fact that new contextual attributes for triggering adaptation can arise implies that:

1. The set of attributes that can trigger adaptation needs to be extensible.
2. The characteristics of all these attributes vary.

The first of these implications places a specific requirement on the architecture to allow the incorporation of future attributes that can become important for mobile systems. In order to support this, a common interface could be used for communication between devices (that monitor attributes) and the architecture. In this way, new attributes could be incorporated providing there is a device driver that exports the predefined interface.

Although this approach appears feasible, the second implication limits the extent to which a common interface for all devices can be used. In particular, the attributes that are monitored differ in a wide range of characteristics (different sets of values, different types of values, etc.). A feasible approach is to group the attributes into sets with common characteristics. Future attributes could match the characteristics of a predefined group and use the predefined interface for that group. Moreover, new groups of attributes can be introduced when the predefined ones are unsatisfactory. An even more flexible approach is to provide a meta-layer that will allow the device module to specify the functionality of the interface itself using, for example, a language such as XML.

Application Control and Coordination

A second requirement is the need to be able to control adaptation behaviour across all components involved in the interaction. As described earlier, one of the main limitations of current approaches is the fact that the applications themselves are responsible for triggering an adaptive mechanism when the underling infrastructure notifies them about any changes. In order to support flexible and coordinated adaptation there is a requirement for triggering adaptation on a system-wide level. Given this approach, the decision about when and how an application should adapt is pushed into an external entity, with cross-application knowledge, while the adaptive behaviour is still a part of the application's characteristics.

Specifying a flexible interface between the platform and the application is necessary because the trivial approach of defining a generic interface for adaptation triggering does not provide the level of control required. More specifically, the fact that every application can provide different types of adaptive modes (e.g. different states of operation) with different characteristics makes the specification of a generic interface impractical. A more better approach is to define a mechanism whereby the applications themselves can register the set of adaptive modes that they support. The registration process would need to include the intended effects of each mode of operation in order to allow the external entity to choose which one to invoke.

Invoking an adaptive mode on an application requires a decision-making mechanism. The system may specify an aim according to the current situation (e.g. reduce the network bandwidth requirements). However, making a decision about which application and which adaptive mechanism should be triggered requires a method of predicting the results of each action. One possible approach is to request applications to provide information on the potential affects that a certain adaptation might have on the resources of the system. This approach cannot be based on discrete values because the actual affects are influenced by several factors that may be difficult to predict. An alternative approach could be based on monitoring the resources of the system. In this case, the platform can attempt various alternative combinations of adaptation and find the one closest to the desired result.

The most efficient solution could be an integration of the two aforementioned approaches. More specifically, the first approach could be used for the initial, course-grained, decision on the most efficient combination of adaptation triggering. Following this, the monitoring approach could be used for making minor adjustments in order to achieve the most efficient result. Another factor that can affect the decision-making mechanism is the fact that an adaptive mechanism could have side effects on other attributes of the system. One solution would be to have a system where all modules have profiles that specify their effects on all the

attributes of the system. Although this is possible with the use of profiling tools, it is not feasible to require application developers to provide such detailed information.

Support for System Wide Adaptation Policies

A further requirement is to support the notion of system-wide adaptation policies. More specifically, such policies should enable a mobile system to operate differently given the current context and the requirements of the user.

The specification of adaptation policies should be goal-oriented. Two kinds of goals can be identified:

1. Effects on resources. The policy specifies a specific aim for the use a specific resource. Example policies include reducing the required network bandwidth, keeping the cost at a specific level and maximising the duration of operation of the system.
2. Effects on applications. The policy specifies the mode of operation for specific applications. Example policies include defining priorities on applications which determine the order in which they are allocated resources and maximising the duration of operation of the system while having a specific application operating with full functionality.

Suitable methods for specifying adaptation policies need to be investigated.

Distributed Operation

A final requirement arises from the fact that most mobile applications operate in a distributed environment including, e.g. a host application on the mobile system, possibly one or more proxies and a server. The adaptation mechanism in such systems usually requires adaptive behaviour and co-ordination of the distributed parts of the system. For this reason, the adaptation mechanisms need to have coordination across all the components involved in the interaction.

5. ARCHITECTURAL FRAMEWORK

Existing mobile systems which use middleware platforms typically have an architecture similar to that shown in figure 4. The key points to note from this figure are as follows. Firstly, mechansisms and policies for adapatation are tightly coupled and encapsulated in both applications and supporting middleware. This is a natural consequence of the trend towards applications being responsible for adapting to changes in context but leads to the problems discussed in sections 2 and 3. Secondly, as discussed earlier, there is no flow of control from the middleware to the applications, making coordinated responses to change impossible.

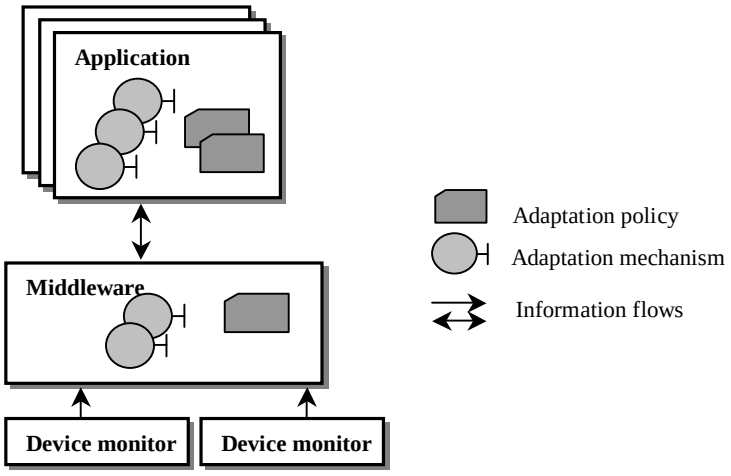

Figure 4: Typical middleware architecture.

We propose that future systems should adopt an architecture in which mechanisms and polices are decoupled and, furthermore, mechanisms are exposed in order to enable control by independent entities. This separation of mechanisms and policies is, of course, a well established principle of distributed systems but one which has not yet been adopted by the developers of adaptive mobile systems. One possible interpretation of such an architecture is shown in figure 5.

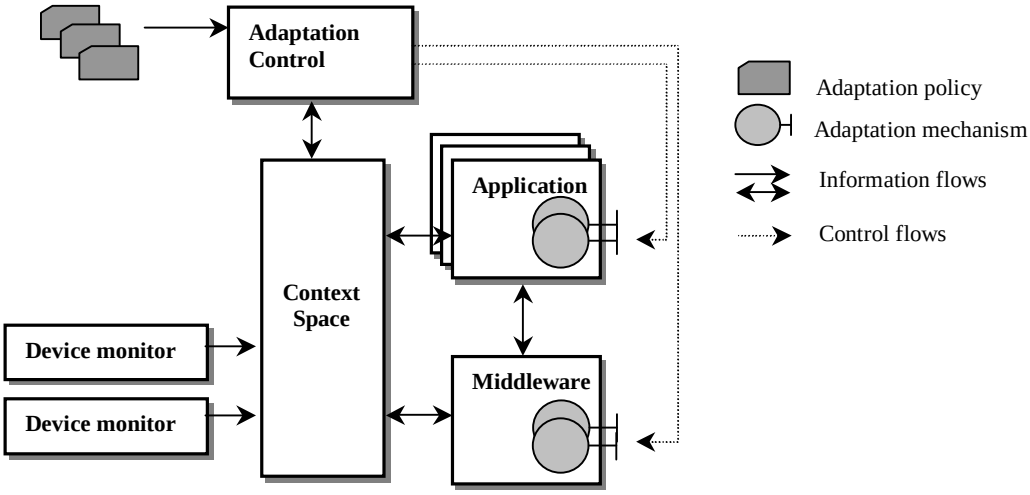

Figure 5: The proposed architectural framework.

The main components of our architecture are as follows:

Context Space: Central to our architecture is the context space. This acts as a repository and distribution bus for information relating to QoS and context within the system. In particular, it is responsible for storing information from the device monitors, applications and middleware for use in determining the correct adaptation strategy in a given situation. The space must enable information from remote sources to be made available.

Device Monitors: Device monitors are typically simple daemon processes which monitor the state of devices and software components and report this information to the context space. Examples might include network device drivers and power management systems.

Applications and Mechanisms: Applications that include mechanisms for adaptation can register with the context space for information and control. It is the responsibility of the application developers to make the interfaces for adaptation mechanisms available.

Middleware and Mechanisms: In common with applications, middleware platforms can register with the context space for information and control. This enables the system to control and coordinate the actions of the middleware and the applications to avoid duplication of effort or conflicting actions.

Adaptation Control and Policies: The key aspect of our architecture is the adaptation control module. This is responsible for coordinating system responses to changes in the environment and resolving potential conflicts when multiple attributes change. The module is driven by a series of policies, which we envisage as being self-contained units that specify how a system should respond in a given situation.

In this paper we do not propose specific technologies for implementing the above components. However, there are clearly lots of existing research results which have been used in similar frameworks and which could be used in this architecture. For example, the context space could be based on an eventing platform and associated event repository such as [20], a tuple space as proposed by Davies et. al. in [6] or a persistent object store provided by a middleware platform such as CORBA or DCOM. Similarly, mechanisms and policies for middleware adaptation have already been under development for a number of years and these can be integrated to provide a foundation for adaptive applications.

The most novel aspect of our architecture is that we are hypothesising that policies can be constructed to support system wide adaptation to multiple triggers in an independent manner. Moreover, it will be necessary for such policies to be applicable in a wide range of system and application configurations and for the system to be able to understand and monitor the results of the policies' actions. A number of policy specification techniques have been developed including rule-based scripts [2], finite state machines [3] and encapsulated policy objects [16]. How this diverse range of approaches can be integrated into the architecture, or how a single approach can be applied is an open research issue.

6. CONCLUSIONS

In this paper we have explored the requirements for future mobile middleware and suggested an abstract framework within which new and existing research activities can be applied. These requirements have been validated using a number of real-world scenarios which, in addition, have illustrated the shortcomings of existing approaches. Furthermore, we have mapped the abstract requirements derived from these scenarios into a set of concrete architectural requirements. These architectural requirements have then been used to develop a high-level architectural framework for supporting adaptive mobile systems. We hope that these requirements and the associated architectural framework will provide input into existing and future research efforts in the field of adaptive mobile systems. In particular, we hope that future middleware will provide better support for developers of applications which need to adapt to multiple contextual triggers in a cooperative environment.

Our future work will be to evaluate the feasibility of our approach by developing an implementation of the core components of our architecture. We will then compare the performance of a number of adaptive applications operating in conventional environments with those operating in our environment as their context changes.

References

1. *Advanced Configuration and Power Interface Specification*, Revision 1.0, Intel/Microsoft/Toshiba, February 1999.
2. Li, B., K. Nahrstedt, "A Control-based Middleware Framework for Quality of Service Adaptations". To appear in IEEE *Journal of Selected Areas in Communications, Special Issue on Service Enabling Platforms*, 1999.
3. Blair, L. "The Role of Temporal Logic and Timed Automata in Distributed Multimedia Systems". In *Proceedings of Modal and Temporal Logic Based Planning for Open Networked Multimedia Systems (PONMS '99)*, 1999.
4. Cho, K. and K. Birman. "A Group Communication Approach for Mobile Computing", *In Proceedings of IEEE Workshop on Mobile Computing Systems and Applications*, 1994.

5. Davies N., K. Cheverst, K. Mitchell and A. Friday. "Caches in the Air: Disseminating Information in the Guide System". *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications (WMCSA '99)*, 1999.

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

7. Elis C. "The Case for Higher-Level Power Management". *Proceedings of HotOS*, 1999.

8. Fitzpatrick T., G. Blair, G. Coulson, N. Davies and P. Robin. "Software Architecture for Adaptive Distributed Multimedia Applications". *In Proceedings of IEE Software*, 145(5), pp 163-171, 1998.

9. 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.

10. 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.

11. Joseph A., J. Tauber and F. Kaashoek. "Mobile Computing with the Rover Toolkit". *IEEE Transactions on Computers: Special issue on Mobile Computing*, 43(3), 1997.

12. Katz R. "Adaptation and Mobility in Wirless Information Systems", *IEEE Personal Communications*, 1(1), pp. 6-17, 1994.

13. Kravets R. and P. Krishnan. "Application-Driven Power Management for Mobile Communication", *In Fourth ACM International Conference on Mobile Computing and Networking (MOBICOM '98)*, 1998.

14. Kunz T. and J. Black. "An Architecture for Adaptive Mobile Applications". *In Proceedings of the 11th International Conference on Wireless Communications (Wireless '99)*, 1999.

15. Long, S., R. Kooper, G.D. Abowd and C.G. Atkeson. "Rapid Prototyping of Mobile Context-Aware Applications: The Cyberguide Case Study". *In Proceedings of the 2nd ACM International Conference on Mobile Computing (MOBICOM)*, 1996.

16. Lupu E. and M. Sloman. "A Policy Based Role Object Model". In *Proceedings of the First International Distributed Object Computing Workshop (EDOC'97)*, 1997.

17. Marzullo K., R. Cooper, M. Wood and K. Birman. "Tools for Distributed Application Management". *IEEE Computer*, 24(8), pp 42-51, 1991.

18. 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.

19. Pascoe J. "The Stick-e Note Architecture: Extending the Interface Beyond the User", In *Proceedings of the International Conference on Intelligent User Interfaces*, 1997.

20. Spiteri M. and J. Bates. "An Architecture to Support Storage and Retrieval of Events". In *Proceedings of Middleware '98*, 1998.

21. Stem M. and R. Katz. "Reducing power consumption of network interfaces in hand-held devices". In *Third International Workshop on Mobile Multimedia Communications (MoMuc-3)*, 1996.

22. Terry D. B., M. Theimer, K. Petersen and A. J. Demers. "Managing Update Conflicts in Bayou, a Weakly Connected Replicated Storage System". *In Proceedings of the 15th ACM Symposium on Operating System Principles*, 1995.