A Framework for QoS-Aware Service-based Mobile Systems

Joel Gonçalves, Luis Lino Ferreira

CISTER Research Center
Polytechnic Institute of Porto (ISEP/IPP)
{vjmg; llf@isep.ipp.pt}

Abstract. In this paper we propose a framework for the support of mobile application with Quality of Service (QoS) requirements, such as voice or video, capable of supporting distributed, migration-capable, QoS-enabled applications on top of the Android Operating system.

Keywords: Quality of Service, mobile systems, Android OS.

1 Introduction

Mobile applications are increasingly more ubiquitous and more dynamic. Furthermore, mobile systems are increasingly open to third-party developed applications, being expectable that users put more and more pressure on locally available resources. But, even considering the substantial increase in devices' capabilities, it is not expectable that they will be able to simultaneously support all applications the users may want to execute. The solution to this is to allow applications to scavenge resources available in neighbor nodes by allowing applications to migrate some or all of its services into other nodes (or from other nodes). Therefore, there is a growing need to develop frameworks and applications that are able to reconfigure considering system-wide distribution of application and resources. Applications must therefore support: i) the capability to connect seamlessly to remote services and; ii) the capability to move some of its services to remote nodes.

Our goal is to provide augmented functionalities (quasi-)transparently in existent middleware and operating system. The Android operating system is used both due to its open source nature and potential market, but also due to its innovative architecture. Although its use to support real-time applications is still debatable [4], it nevertheless provides a suitable architecture for QoS-aware applications in ubiquitous, embedded systems.

Android applications (constituted by Intents, Activities and Services) and its resources are contained in an Android Package (*APK*) which can be installed in a local device. During runtime, the APKs components can be assembled to create dynamic applications, but these functionalities are only available in the local node. In this work, we extend this concept to a fully distributed and dynamic environment, where applications use the Activities and Services from the *APKs* whether they are installed locally or remote. Further, we allow *APKs* to migrate to other nodes, by user demand or due to system's reconfiguration. Finally, the framework is also able to support

INForum 2010 - II Simpósio de Informática, Luís S. Barbosa, Miguel P. Correia (eds), 9-10 Setembro, 2010, pp. 711–714

applications with QoS requirements, both during its run-time operation and particularly during the migration of services.

The *APK*s are executed as independent processes, with distinct permissions, and, importantly, also with different QoS parameters. Figure 1 provides an example: the initial scenario is presented in the left part, where Device 1 is executing an application that uses services from three distinct *APKs*, in Device 1 (*APK* A) and Device 2 (*APK* B and *APK* C).

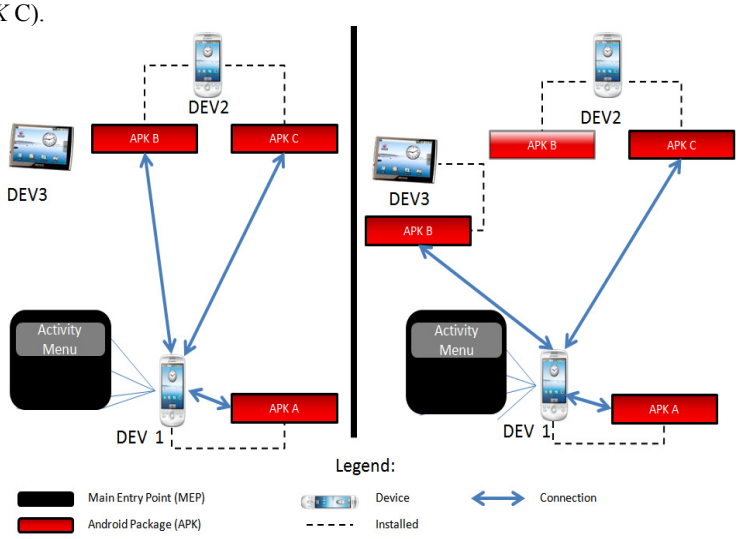

Fig. 1. Example scenario

The right side of Figure 1 provides a new system configuration, which includes a new device: Device 3. Assuming that this device has more resources available than Device 1, it enables a new application configuration with a higher QoS level if *APK* B is migrated to Device 3.

Algorithms and frameworks as the ones proposed in CooperatES [3] can be used to find the new configuration for the system services, maximizing the rewards for the overall system. As a result of the evaluation, Device 3 now offers the services of *APK* B. Mobile code mechanisms of the proposed framework support such approach, making possible to transfer (guaranteeing the QoS requirements), the code and state of service B from Device 1 to Device 3, install the corresponding *APK* file, rebind the connections between Device 1 and the service, and continue its operation.

2 Framework Architecture Overview

In order to implement our approach we assume the existence of a QoS Manager on the Android Operating System, the addition of application layer features to support the core functionalities of a mobile code framework and the use of supporting libraries which allow programmers to use the full set of capabilities offered by the framework.

The *Framework Core* functionalities (Figure 2) is constituted by separate modules implemented as an Android services, which takes care of service migration to and from another node, interacting with the QoS Manager in order to determine if the QoS requirements of the service can be supported.

Any Android application can use the services of the framework in order to support installing and running an Android *APK* in another node. More advanced services, which require the rebinding of connections between components, are only supported if applications use the *Mobile Library*.

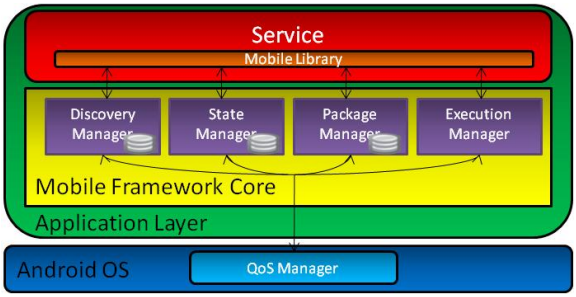

Fig. 2. Framework core modules

The *Mobile Library* offers a set of helper classes that extend the core functionalities of Android with QoS and distributed capabilities. The library allows to transparently extend the *Activity* and *Service* abstractions for distributed systems. Consequently, application code is only required to send an *Intent*, after which the framework is responsible for determining if it refers to a local or remote component. The library also implements the services required for communication establishment and automatic rebinding of service's connections when a service migrates. This library [5] already implements some generic mechanisms, which can be immediately used by developers, but also allows extensions to the base classes for new implementations.

The core services provided by the framework are: *Discovery Manager*, *Package Manager*, *State Manager* and *Execution Manager*.

The *Discovery Manager* module is designed to discover neighbour devices on a local network and advertise the host device available resources. To that purpose, every node in the network periodically broadcasts information regarding its status and installed services, such as the *APKs* installed, their associated Intents interfaces and resource availability.

The *Package Manager* is used to install, uninstall and transfer the code of *APKs* between Android devices. Applications can start executing when the node receives an Intent request from a remote *Execution Manager* (see below). It is also responsible for the interaction with the *QoS Manager* in order to request specific QoS levels for the service being handled. It is the responsibility of the *QoS Manager* to accept or reject service installations if the QoS required level cannot be guaranteed.

The *State Manager* handles transfer of state for statefull services. The *State Manager* is responsible for transferring either the full state or specific state items similarly to what has been proposed in [1, 2]. The flexibility on the implementation of the state migration policies can be of paramount importance for the use of code

mobility techniques in real-time systems since it can allow the reduction on the unavailability time of a service.

The *Execution Manager* allows launching services on a host device or on a remote node. Android *Intents* are exchanged between devices and used locally to start up a service, which can be a standard Android *Activity* or *Service. Intents* that address *Activities* or *Services* in remote nodes are parsed to extract their parameters and sent to the remote *Execution Manager*, which then reconstructs the intent and launches it locally (on the remote node).

Services which are based on the framework use the new versions of *Service* and *Activity* classes; therefore, when calling for a remote service they can connect directly using the functionalities provided by the Mobile Library.3

3 Conclusions

In this paper we proposed a framework for the development of distributed QoS aware applications with self-reconfiguration capabilities. The framework is particularly targeted for the Android Operating System and its implementation extends the main abstractions used in Android – *Activities*, *Services* and *Intents*, allowing for transparent interactions of application code in both local and distributed settings. Quality-of-Service requirements of both applications and reconfiguration services are handled by supporting an underlying operating system QoS Manager module.

This framework will be used to develop the real-time capabilities of the Android operating system and particularly to develop adequate strategies for multiple parameter quality-of-service of applications (considering both tasks and communication streams). We also plan to investigate further on how to better manage dynamic adaptations during reconfiguration/mobility phases. An implementation of the framework proposed in this paper is available at [5].

Acknowledgment. This work is partially funded by the Portuguese Science and Technology Foundation (Fundação para a Ciência e a Tecnologia - FCT) and project RESCUE (PTDC/EIA/65862/2006).

References

1. Preuveneers, D. and Berbers, Y. (2010) “Context-driven migration and diffusion of pervasive services on the OSGi framework”, International Journal of Autonomous and Adaptive Communications Systems, Vol. 3, No. 1, 2010, pp. 33-22.
2. Malek, S., Edwards, G., Brun, Y., Tajalli, H., Garcia, J., Krka, I., Medvidovic, N., Mikic-Rakic, M., Sukhatme, G., ”An Architecture-Driven Software Mobility Framework”, Journal of Systems and Software, special issue on Software Architecture and Mobility, Vol. 83, Issue 6, Jun. 2010, pp. 972-989.
3. Nogueira, L., Pinho, L., "Time-bounded Distributed QoS-Aware Service Configuration in Heterogeneous Cooperative Environments", Journal of Parallel and Distributed Computing, Vol. 69, Issue 6, 2009, pp. 491-507.
4. Maia, C., Nogueira, L., Pinho, L., “Evaluating Android OS for Embedded Real-Time Systems”, to be published in the 6th International Workshop on Operating Systems Platforms for Embedded Real-Time Applications (OSPERT 2010), 2010.
5. Distributed and Mobile Framework for Real-Time Systems (DiseRTS), http://www.hurray.isep.ipp.pt/activities/RTSoft/distFrameworkOverview.ashx/, July 2010.