Multi-device Single Sign-On for Cloud Service Continuity

Patricia Arias Cabarcos, Florina Almenares, Rosa Sánchez Guerrero, Andrés Marin, Daniel Díaz-Sánchez,
Telematics Egineering Department, University Carlos III of Madrid, Leganés, Madrid (Spain)

Abstract-- **The great variety of consumer electronic devices with support of wireless communications combined with the emerging Cloud Computing paradigm is paving the way to real anytime/anywhere computing. In this context, many services, such as music or video streaming, are delivered to the clients using Cloud-based providers. However, service continuity when moving across different terminals is still a major challenge. This paper proposes a novel middleware architecture that allows security sessions initiated from one device to be seamlessly transferred to a second one, as might be desirable in the enjoyment of long running media.** [1]

I. INTRODUCTION

Many organizations are seeking to deliver services to their users by means of Cloud-based providers (e.g. video or audio streaming). This is typically motivated by a desire to avoid management of commodity services which, through economies of scale, can often be delivered more efficiently by such providers. This trend, together with the increasing usage of small portable devices and wireless networks is paving the way to real anytime/anywhere computing. Nowadays, due to the dramatic evolution of technological convergence, the different consumer electronic devices that a user owns have similar capacities and may allow him to access the same applications and services. The usage of one device or another will depend on the context, i.e. on which option suits better to the current situation. Furthermore, users want to enjoy services on the move, which entails dynamic changes of context. As a consequence, a user enjoying a long duration service may desire to switch the session across different devices during the lifetime of the service consumption, maintaining continuity.

In this context, the need of adequate Multi-device Single Sign-on (MD-SSO) technologies comes into scene. MD-SSO is defined as "*Single sign-on for users that crosses devices, i.e. the session is initiated from one device or user-agent, and subsequently transferred to a second, as might be desirable in the enjoyment of long running media*" [1].

A use-case that perfectly illustrates the added value of MD-SSO is the following: Bob is on his way home listening to his favorite song list; while he drives, the music is reproduced by the car audio system. When he goes out of the car and walks towards the house, the session is seamlessly transferred to his personal music player and it continues with the current song in the same exact point. Finally, when he enters home, the music session is switched again from Bob's player to his home sound equipment. This switching between Bob's personal music player, in-car player, and house music equipment provides service continuity. Another example could be the case where a user arrives home and wants to transfer his browsing session and current activity on social networks from his smartphone to his TV, which offers a better display and prevents his phone to run out of battery.

Both use cases are feasible with current state of the art technology, and the concept of session mobility across terminals is far from new. In fact, the idea is described as an essential pillar in the traditional literature on ubiquitous computing [2] [3]. Nevertheless, few implementations have been developed and there is no mainstream adoption of MD-SSO technologies, which is especially remarkable and led us to analyze the reasons and propose a solution.

Related academic work in the field usually focus on specific modifications to protocols, and industry work concentrates on proprietary implementations that only work within a narrow set of devices belonging to the same manufacturer or for a specific application or service. Also, some recent approaches[2] are based on synchronization of session data, which are stored on a proxy or external entity accessible by different devices. As it can be seen, the main problems are related to interoperability across heterogeneous systems. Furthermore, in the case of synchronization, outsourcing the storage of personal data to a third entity raises privacy issues, since sessions may be tracked. Regarding usability, repeated authentication is required when accessing synchronization data from a different device, which is a cumbersome task and imposes an undesirable delay. In order to overcome these barriers, we state that an open holistic architecture is required to guarantee flexibility and allow interoperation across all platforms. Thus, a new layer must be defined and embedded in consumer electronic devices to abstract the complexity of session transference. With the aforementioned premises in mind, we propose a middleware architecture aimed at enhancing the user experience when consuming services on the move and maintaining a smooth personal experience even when changing terminals. The original value of our contribution is that serves as a foundation to construct universal MD-SSO for any kind of applications and services, and allows interworking between heterogeneous devices. We focus on maintaining the security sessions (i.e. authentication/authorization state) when changing terminals to achieve service continuity based on single sign-on.

According to the aforementioned goals, the rest of the paper is structured as follows: section II summarizes the main requirements to build a generic MD-SSO system; section III gives a global overview of the proposed middleware architecture; and section IV outlines our work in the prototype, the main conclusions and future lines.

[1] This work was partially founded by the Spanish Ministry of Science and Innovation under the project TEC2010-20572-C02-01 (CONSEQUENCE).

[2] Mozilla Weave: https://wiki.mozilla.org/Labs/Weave/0.2

II. Requirements for a generic MD-SSO system

We identify a basic set of requirements that will be used as principle guidelines to model the proposed architecture for MD-SSO. On the one hand, the functional requirements are:

1) Context management: contextual information must be extracted to determine when a session transfer can be performed.

2) State management: state data of running applications must be obtained before any session transfer. The security context is especially relevant for future session restoring.

3) Session transfer: application data and security context must be transferred between different devices.

4) A*utomatic session restoring*: the destination device must be capable of automatically restoring the applications and its running security sessions on behalf of the user.

On the other hand, is important that the MD-SSO system fulfills two basic non-functional requirements:

1) User centricity: the system must be user friendly, user controlled and take user preferences into account.

2) Flexibility: the system must allow interoperation across heterogeneous platforms and protocols and be capable of accommodating different applications and services.

The aforementioned functional and non-functional requirements lay the foundations for the generic MD-SSO architecture explained in the next section.

III. Middleware architecture for MD-SSO

The architecture of the proposed middleware for MD-SSO consists of a series of interconnected software blocks that distribute all the functionality in a modular fashion, as depicted in Fig.1. To clarify the details of these modules, we provide an individual explanation of each one:

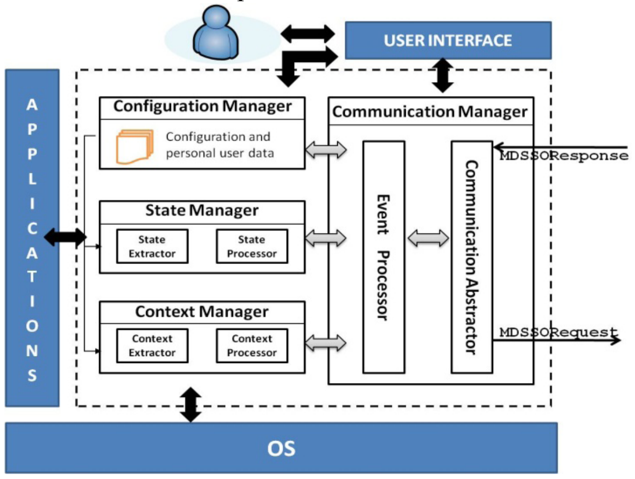

Fig. 1. Middleware architecture for multi-device Single Sign-On

A. Context Manager

The *Context Manager* is in charge of extracting context information, such as remaining battery, available devices in the proximity, current location or time. This context is then processed to determine if a session transfer can be performed. In order to make this decision, user configuration data provided by the Configuration Manager is also employed.

B. State Manager

The function of the *State Manager* is obtaining the state of the applications that are currently running in the origin device, so it can be subsequently transferred to a destination device. This module is the key for flexibility. Any application developer or service provider that wishes to offer MD-SSO to the final users, should implement a *plugin* component for this module. The *plugin* should conform to a defined format and its function is to retrieve the state of the associated application and encapsulate it in a XML file. Furthermore, when a session transfer is received, the correspondent *plugin* will be provided with the XML file with the state information in order to notify its associated application and restore the session.

C. Configuration Manager

The *Configuration Manager* module is responsible of handling user configurable information: preferences for session mobility, user credentials that may be required for authentication of terminals on behalf of the user, etc… This information is accessed and manipulated via a user interface.

D. Communication Manager

The *Communication Manager* handles communication with other external devices. It receives notifications from the *Context Manager* when a session transfer is possible. As a reaction to these events, the module asks the *State Manager* to get the information of the running applications that are suitable for a session transfer. Under user consent, the Communication Manager sends a package with all the state information required for session restoring in the destination device. Two communication primitives have been defined for this purpose: `MD-SSORequest` and `MD-SSOResponse`. The first message includes data about the sessions to be restored; and the second one notifies the origin device indicating if the session transfer has been successful or not. It is to mention that different communication technologies (e.g. Bluetooth, RFID) can be used depending on availability and/or user preferences.

IV. Implementation and conclusions

So far, we have formally defined the middleware (i.e: APIs, communication primitives, storage formats, etc.) and partially implemented a prototype. Based on it, we are testing a use-case where the state of the browsing activity is transferred between two devices, and we are defining a testbed for performance measurement. Since current solutions for MD-SSO are proprietary or limited, we maintain that a generic middleware, as the one proposed here, is indispensable to foster healthy progressive adoption by industries and users.

References

[1] P.Madsen (ed.).: "*Liberty ID-WSF Multi-Device SSO Deployment Guide*", 2008.

[2] M. Satyanarayanan, *Pervasive Computing Vision and Challenges*. IEEE Personal Communications, 2001.

[3] M. Weiser, *The Computer for the 21st Century*. Scientific American, September 1991.