Supporting Pervasive and Social Communications with FraSCAti

Rémi Mélisson, Daniel Romero, Romain Rouvoy, Lionel Seinturier

Electronic Communications of the EASST
Volume 28 (2010)

Proceedings of the Third International DisCoTec Workshop on Context-aware Adaptation Mechanisms for Pervasive and Ubiquitous Services (CAMPUS 2010)

Supporting Pervasive and Social Communications with FRASCATI

Rémi Mélisson, Daniel Romero, Romain Rouvoy, and Lionel Seinturier

13 pages

Guest Editors: Sonia Ben Mokhtar, Romain Rouvoy, Michael Wagner
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Supporting Pervasive and Social Communications with FRASCATI

Rémi Mélisson, Daniel Romero, Romain Rouvoy, and Lionel Seinturier

INRIA Lille – Nord Europe, ADAM Project-team,
University of Lille 1, LIFL CNRS UMR 8022,
59650 Villeneuve d'Ascq, France
firstname.lastname@inria.fr

Abstract: In pervasive environments, the runtime adaptation of applications is done considering available event sources and services. To do that, the events have to be collected and processed, and the volatile services identified and accessed. However, although the event flow and service mobility are key issues in the adaptation process, existing solutions fail to deal with them in a simple and flexible way. Therefore, in this paper we propose to face these issues by combining the SCA (Service Component Architecture) standard, micro-blogging services and discovery technologies. In particular, we benefit from the SCA extensibility, supported by the binding concept, for introducing two new communication mechanisms: *i) social bindings*, allowing asynchronous event exchange via Twitter and, *ii) pervasive bindings* that provide support for discovery using standard protocols such as UPnP. We add support for these bindings in FRASCATI (a platform for SCA) and illustrate their usage with a smart home scenario that requires the integration of heterogeneous technologies.

Keywords: social networks, pervasive computing, service-oriented architectures

1 Introduction

In pervasive environments, the adaptation of applications requires the collection, filtering and processing of events produced by different kinds of sources (*e.g.*, mobile devices and sensors), to decide the required reconfigurations. The different responsibilities of the adaptation—*i.e.*, monitoring, analysis, decision making and execution—can be distributed in several entities, which dynamically join and leave the environment, in order to benefit from the most powerful devices. Therefore, we can consider the adaptation as a process that requires the exchange of events between the different participants—*i.e.*, event sources, decision makers and adaptive applications. However, although the information flow and mobility are key issues in the adaptation process, the existing approaches fail to deal with them in a simple and flexible way.

In this paper we propose to extend the SCA (*Service Component Architecture*) [Ope07] model in order to increase communication capabilities in the changing pervasive environments. In particular, SCA provides the flexibility for adding new kinds of interaction by means of bindings. SCA bindings encapsulate the communication protocols, isolating distribution concerns from the business logic. We take advantage of this concept in order to bring social and pervasive communications into SCA. In this way, we introduce two new bindings: *social bindings* and *pervasive bindings*. The formers benefit from micro-blogging services, such as *Twitter* [Mak09], to build

social networks for asynchronous event exchanges in pervasive environments. The latter supports mobility of entities in the environment by providing discovery support via the UPnP (*Universal Plug and Play*) protocol [UPn08]. We choose UPnP because it is a well-accepted standard in the home entertainment industry to create, for example, TVs and Network-Attached Storage (NAS) devices. Furthermore, the Digital Living Network Alliance [1] (DLNA), which ensures interoperability between devices from different manufacturers, supports the usage of the UPnP technology. We integrate the social and pervasive bindings into FRASCATI [SMF+09] (a SOA platform for the SCA standard) and illustrate their use with a smart home scenario exhibiting challenges in terms of protocol heterogeneity and integration. We claim that the encapsulation of the distribution concerns in SCA bindings enables the transparent event exchange and dynamic service discovery in pervasive environments. We illustrate the usage of our social and pervasive bindings by defining a smart home scenario

The rest of this paper is organized as follows. We start by presenting a smart home scenario (cf. Section 2) in order to motive the use of our approach and the foundation of our proposal (cf. Section 3). We continue by describing the integration of our pervasive (cf. Section 4) and social (cf. Section 5) bindings for allowing event distribution. After discussing the state-of-the-art (cf. Section 6), we conclude and give some promising perspectives (cf. Section 7).

2 Motivating Scenario: The DIGIHOME Platform

The work presented in this paper is motivated by the DIGIHOME platform, which introduces the challenges for better connecting smart appliances in home environments. As explained in [RHT+10], a DIGIHOME environment is characterized by the diversity of communication protocols supported by the physical equipments deployed in a house. Although, standards like UPnP and more recently *ZigBee* [Zig10] have emerged and are supported by a growing number of commercial solutions, there is no general agreement on a single protocol for supporting the integration of home appliances. Additionally, the emergence of social networks like Facebook and Twitter have encouraged new means of communication where users can easily share information with a community of trusted contacts via micro-blogging services. In this context, we describe two scenarios where these technologies are smoothly integrated in a modular platform for implementing advanced context-aware home monitoring situations.

Scenario 1. Alice owns two digital frames at home. One of them is able to display pictures from a UPnP media server, via a WiFi connection. The other frame has the ability to subscribe to a *Really Simple Syndication* (RSS) feed remotely published. However, Alice is not interested in displaying pictures that are randomly or statically selected in a library. She would rather prefer to display pictures according to her personal preferences stored on her smartphone. This feature therefore requires the DIGIHOME system to automatically detect Alice's smartphone and retrieve her preferences for customizing display for the two frames. Additionally, when Bob joins the room, the DIGIHOME platform should import his preferences, and try to satisfy Bob's and Alice's desires to display pictures.

[1] Digital Living Network Alliance: http://www.dlna.org/home

Scenario 2. Alice plans to leave for holidays with Bob in a few days. Before leaving, Alice contacts her neighbors asking them if they could take care of the house while she is abroad. Neighbors can accept her proposal by following her on Twitter in order to be notified of particular events related to the house. Whenever a movement is detected in the house while Alice is away, the DIGIHOME platform will post a new message on Alice's Twitter account to notify her friends that something is happening in the house. Furthermore, to ensure the messages reception, the system filters the posts on the Alice's account and displays them on neighbors' UPnP TV. Depending on the content of the message, Alice's neighbors can walk to the house and physically check the status of the house. The DIGIHOME platform detects the neighbor's smartphone, disables the house monitoring, and logs the presence of the neighbor by posting another message on Twitter.

Challenges. The scenarios we describe here exemplify situations where the DIGIHOME platform has to integrate different technologies (UPnP, RSS, Twitter) to implement advanced situations combining the house equipments (media server, Frames, Set-Top Box), smartphones, and the Internet. In the remaining of this paper, we introduce a solution that aims to deal with this technology integration. In particular, we build our solution with SCA by defining two features: *pervasive bindings* and *social bindings*, which we use to implement scenarios 1 and 2, respectively.

3 Background on Service-Oriented Architectures

In this section, we present the foundation of our proposal—*i.e.*, the SCA (cf. Subsection 3.1) component model. We also introduce the FRASCATI platform (cf. Subsection 3.2), which we use to implement our approach.

3.1 The Service Component Architecture (SCA)

SCA is a set of specifications for building distributed application based on SOA and *Component-Based Software Engineering* (CBSE) principles [Ope07]. In SCA, the basic construction blocks are the *software components*, which have *services* (or provided interfaces), *references* (or required interfaces) and expose properties. The references and services are connected by means of *wires*. SCA specifies a hierarchical component model. This means that components can be implemented either by primitive language entities or by subcomponents. In the latter case the components are called *composites*. Figure 1, from the SCA specifications [Ope07], provides a graphical notation for these concepts as well as a XML-based assembly language to configure and assemble components.

SCA is designed to be independent from programming languages, *Interface Definition Languages* (IDL), communication protocols and non-functional properties. In this way, an SCA-based application can be built, for example, using components in Java, PHP, and COBOL. Furthermore, several IDLs are supported, such as WSDL and Java Interfaces. In order to support interaction via different communication protocols, SCA provides the concept of *binding*. For SCA references, *bindings* describe the access mechanism used to call a service. In the case of services, the bindings describe the access mechanism that clients have to use to call the service.

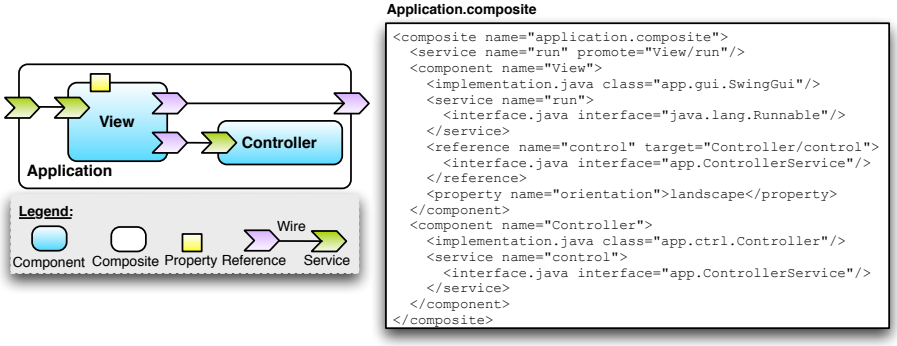

Figure 1: SCA graphical notation and assembly language.

3.2 The FRASCATI Platform

The FRASCATI platform [SMF+09] allows the development and execution of SCA-based applications. The platform itself is built as an SCA application—*i.e.*, its different subsystems are implemented as SCA components. FRASCATI extends SCA with reflective capabilities at the application level as well as at the platform level. By default, the FRASCATI platform is bundled with various plugins for the components Interface (Java, WSDL), Property (Java, XSD), Component (Java 5, Java Beans, Scala, Spring, OSGi, FRACTAL, BPEL, script), and Binding (Java RMI, SOAP, HTTP, JSON-RPC, SCA, OSGi, JNA). Furthermore, the platform has a run-time level that instantiates SCA assemblies and components. This run-time is not hard-coded into the architecture description. Rather, the run-time level defines a flexible configuration process, which is inspired by the *extender* and *whiteboard* [OSG04] design patterns of OSGi. In particular, both the core platform and various plugins are bundles, which are dynamically composed.

4 Scenario 1: Bindings for Pervasive Communications

In pervasive environments, the discovery capabilities enable the use of the available services in a transparent way and provide support for mobility. Therefore, we bring this functionality into SCA by defining *pervasive bindings*. In this section, we report on the integration of these bindings in the FRASCATI platform, using the UPnP technology (cf. Subsection 4.1 and Subsection 4.2). We also introduce our solution for the automation of service discovery in SCA-based applications (cf. Subsection 4.3). This integration is validated by the implementation of the first scenario (cf. Subsection 4.4).

4.1 UPnP Foundations for Pervasive Integration

As an Internet-based technology, *Universal Plug and Play* (UPnP) is built upon TCP/IP, UDP, HTTP, XML, and SOAP. The recent development of connected devices like *Set-top boxes* (STB) or smartphones has encouraged the integration of this technology as an industry standard for interacting with these smart appliances. Therefore, we believe that UPnP represents a key technology in pervasive environments for enabling the integration of physical devices with IT sys-

tems. Typically, this integration requires considering four aspects of UPnP: *service discovery*, *description*, *interaction*, and *profiles*.

Service Discovery. The most important feature of UPnP technology for pervasive usage is the discovery capacity. Using the *Simple Service Discovery Protocol* (SSDP) [UPn08], client and server (called, in UPnP terminology, *Control Point* [CP] and *Device*, respectively) can find themselves without any configuration. In order to establish a connection, a CP sends a lookup request (using HTTP over UDP), which includes a device type, a particular service or a specific device. Devices can send a response that contains the URL of their description. In a similar way, when a device joins the network, it sends an advertisement announcing its presence. Finally, the devices multicast *bye-bye* notifications indicating that they leave the network and therefore the provided services are no more available.

Service Description. Like any service-oriented application, UPnP devices have to describe their capabilities. To do that, UPnP devices and CPs are described by means of XML documents. These documents are structured following a collection of schema documents, which are also provided by the standard. For example, the DLNA imposes a particular schema for the description of entertainment devices that are able to share their content with each other across a home network.

Service Interaction. To support the remote invocation of services, UPnP exploits *Simple Object Access Protocol* (SOAP) as a privileged communication protocol. Thus, the method invocations are described with XML and sent via SOAP over HTTP.

Service Profiles. Additionally, UPnP provides a set of profiles representing manufacturer contracts and ensuring interoperability between heterogeneous hardware. Actually, the most widespread profiles are the *Audio Video set* (AV) profiles, which enable media-oriented home networks. However, the use of Web technologies, such as SOAP and XML, makes UPnP inappropriate for real-time operations including multimedia streaming or any other kind of operation that requires large amount of data transfer. Therefore, the AV profiles face this issue by describing UPnP devices in terms of their remote control and streaming capabilities. The remote control functionality can be considered as a simple service, while the streaming description allows devices to specify the stream properties, including transfer protocol (*e.g.*, RTSP/RTP, HTTP) and data format. Thus, the AV specifications split the connection negotiation in two steps: *communication description* and *communication establishment*.

4.2 Implementing Pervasive Services using UPnP and SCA

As mentioned previously, SCA introduces *bindings* as an open mechanism supporting communications between components. Bindings can be implemented in order to fulfill the requirements of business applications. In pervasive environments, the UPnP discovery capabilities—enabling spontaneous communications at runtime—can be considered for dealing with services mobility. The comprehensive UPnP communication stack allows the advertisement, discovery, and access of services provided by Java applications. Therefore, we introduce in SCA the `<binding.upnp device-type="...">` element specifying that a service (resp. reference) is advertised (resp. dis-

covered) via the UPnP protocol. Thus, UPnP can be used as an SCA binding for autonomous communications applying SOAP as an underlying mechanism for service invocations.

On the other hand, benefiting from the UPnP modularity in terms of the spontaneous communication process, we can discover services with UPnP and then access them with different communication protocols (not only SOAP), which are described by the others bindings (HTPP, WSDL, etc.) associated with the SCA services. This means that the service descriptions and access mechanisms are only advertised and discovered with UPnP. Then, using this information, the communications are achieved across a suitable protocol, which may be selected depending on QoS attributes. Thus, we restrict the usage of UPnP to the discovery part of the spontaneous communication process.

4.3 BINJIU: Facilitating UPnP integration into FRASCATI

In this section we describe BINJIU, our solution to provide the missing elements for implementing UPnP bindings in FRASCATI. In particular, this solution provides functionality for UPnP interface compilation and descriptor generation. BINJIU also enables the SCA services advertisement as UPnP devices and the casting of SCA references to UPnP control points. Below we describe these functionalities as well as the integration within FRASCATI as a dedicated binding.

UPnP Interface Compilation. In order to integrate a standardized UPnP profile into the FRASCATI platform, we have to implement an SCA component, which is compliant with the profile description. To do that, BINJIU provides a `upnp2java` command that parses UPnP descriptions, extract UPnP services, and generates the associated Java interfaces. Thus, this `upnp2java` command makes easier the integration of existing profiles, such as the *Content Directory UPnP profile*[2], described below:

```
public interface ContentDirectory {
    public int GetSystemUpdateID();
    public SearchResponse Search(String ContainerID, String SearchCriteria,
       String Filter, int StartingIndex, int RequestedCount, String SortCriteria);
    public String GetSearchCapabilities();
    ...}
```

UPnP Descriptor Generation. BINJIU supports also the generation of UPnP descriptions from Java interfaces, according to the UPnP schemas. In this way, we can create devices automatically exposing the UPnP descriptions of the underlying Java interfaces. Even if these XML descriptions are not standardized, it is a first step for autonomic bindings of Java native interfaces in distributed software.

UPnP Device Integration. The advertisement of an SCA service via UPnP technology requires to consider the different aspects discussed in Subsection 4.1. In this way, BINJIU dynamically encapsulates a Java interface and exposes it as a UPnP service, included into a device. This skeleton is able to follow the SSDP advertisement process and answer to UPnP connection

[2] UPnP Content Directory profile: http://www.upnp.org/standardizeddcps/documents/ContentDirectory1.0.pdf

requests from control points. The current BINJIU implementation is based on the CyberLink[3] UPnP stack for Java.

UPnP Control Point Integration. In the client side, BINJIU creates a proxy from a Java interface, which dynamically converts method calls to actions towards a UPnP device, depending on a device type or name. This stub is also implemented using CyberLink for Java.

FRASCATI Integration. BINJIU has been built as an autonomous application to make easier the integration of the UPnP technologies in Java-based applications. However, we need to integrate the BINJIU functionalities into the FRASCATI platform in order to enable the use of pervasive bindings (cf. Subsection 4.2) in SCA applications. To do that, we take advantage of the FRASCATI extensibility properties. In particular, FRASCATI already provides various SCA bindings (*e.g.*, RMI, REST, and WebServices), which means that there is an extension mechanism for integration of new kinds of bindings. Thus, we can add our support for pervasive bindings without impacting the platform functionality.

4.4 Validation

The first scenario, introduced in Section 2, illustrates how Alice's preferences influence the frames displaying pictures at home. In this section, we describe how FRASCATI and pervasive bindings can be used to realize this scenario.

Alice's home is equipped with different entertainment devices. In order to keep the data separated from the different computing resources, a NAS is available as a standalone device. This NAS broadcasts Alice's multimedia contents over the network, as a UPnP AudioVideo Profile (DLNA certified) device. Furthermore, we find a *Set-top box* (STB) as a central element of the environment, which is always available and is able to run FRASCATI applications. The two digital frames (UPnP AV and RSS based) are available in the environment and an Android smartphone is connected to the home network using a wireless connection. Figure 2 depicts the different entities in Alice's home and how they are associated with the DIGIHOME system.

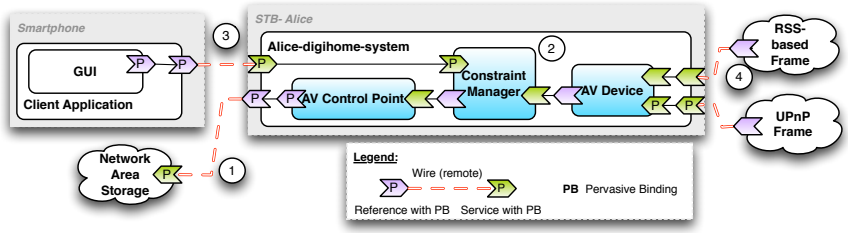

Figure 2: Using FRASCATI with UPnP bindings.

To access the pictures stored in the NAS, we need to generate a Java interface compliant with UPnP AV profile (cf. Figure 2:1) using the upnp2java command provided by BINJIU. Then, we create an SCA implementation of the AV profile with CP capabilities (the `AV Control Point` component) for communicating with the NAS over UPnP.

[3] CYBERLINK FOR JAVA: http://cgupnpjava.sourceforge.net

In order to filter NAS content according to Alice's preferences, we define a simple interface, which provides methods for defining constraints on the picture library (cf. Figure 2:2):

```
public interface IConstraintManager {
    public void addKeywordsConstraint (String[] keywords);
    public void addDateRangeConstraint (Date date1, Date date2);
    public void addFolderConstraint (String folderName);}
```

The `Constraint Manager` component provides this interface. This component is able to filter NAS responses and is advertised over the network as a UPnP device thanks to our UPnP binding (cf. Figure 2:3). By means of a simple control point generated by BINJIU, the Alice's smartphone can interact with this service. Finally, the STB exposes the filtered results as a UPnP AV device and a RSS publisher (cf. Figure 2:4), that are used by the digital frames for displaying the content.

5 Scenario 2: Bindings for Social Collaborations

The micro-blogging services enable the sharing of information in real-time by reaching a lot of friends in just a few seconds. One of the main advantages of these services is the short and simple nature of the posted messages. Furthermore, there are no restrictions on the subject associated with these messages. In general, this kind of informal communications was conceived for broadcasting simple events in the daily life, such as what people are doing, thinking, and experiencing [ZR09, McF07]. However, nowadays the micro-blogging services are also used for collaborative work in organizations [ZR09], publicity purposes, and even for broadcasting real-time news updates for recent crisis situations. Therefore, the simplicity and flexibility as well as the real-time notification property of micro-blogs make them a suitable option to enable event-based communication between the participants of the DIGIHOME platform.

In the rest of this section, we present how we bring micro-blogging-based communications into the SCA component model by defining social bindings (cf. Subsection 5.1) and the architecture of our solution (cf. Subsection 5.2). We finish with the validation of our approach by analyzing the performance overhead of our solution in our smart home scenario (cf. Subsection 5.3).

5.1 Enabling Social Collaborations using Twitter and SCA

Following the same idea of pervasive bindings (cf. Section 4), we enable social communications in pervasive environments by introducing a new kind of binding in SCA: the *social bindings*. These bindings allow the notification of situations identified by the system that may require human intervention, like the detection of an intruder in our smart home scenario (cf. Section 2). To do that, social bindings benefit from a simple but widely used micro-blog service: Twitter. Furthermore of the advantages of any micro-blogging service, the Twitter messages (so called *tweets*) can be posted using different formats (*e.g.*, JSON, XML, RSS and ATOM) and a simple RESTful API [Mak09].

Figure 3 depicts an example of social binding for the Twitter micro-blogging service. In the service side (right side of Figure 3), we define a service with a Twitter binding that indicates that the information provided by the service can be retrieved from a particular Twitter account.

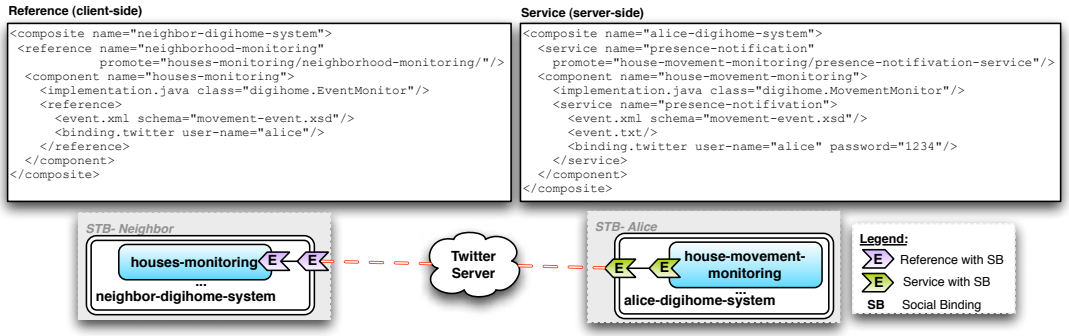

Figure 3: Example of a Social Binding for the Smart Home Scenario

We use the `user` and `password` attributes to indicate the Twitter user providing the information. As it can be seen, unlike the typical SCA bindings, the SCA services defining Twitter bindings do not require to define specific interfaces. Instead, the services specify the representations and types of the events that can be retrieved from the Twitter accounts. This information will be used for marking the tweets with the event type that contains using Twitter tags. This metadata is useful for filtering the events exchanged on the Twitter account. In the reference side (left side in Figure 3), we use the `<binding.twitter>` element to indicate that we want to retrieve events from Twitter. In the references, the `<event.representation>` element specifies the event representations. Thus, by encapsulating the event notifications as SCA social bindings, we support asynchronous event notification as well as interaction with people involved in the smart home.

5.2 Architecture of the Containers Supporting Social Bindings

We have integrated these social bindings into the FRASCATI platform. To do this, we applied the `ContainerComposite` architectural pattern promoted by the HULOTTE platform [LSDS10]. This means that we define a container composite that hosts components and implements event notification as tweets. In this way, the social communications are integrated in a transparent way and the event processing logic is not impacted by the technical specificities of Twitter.

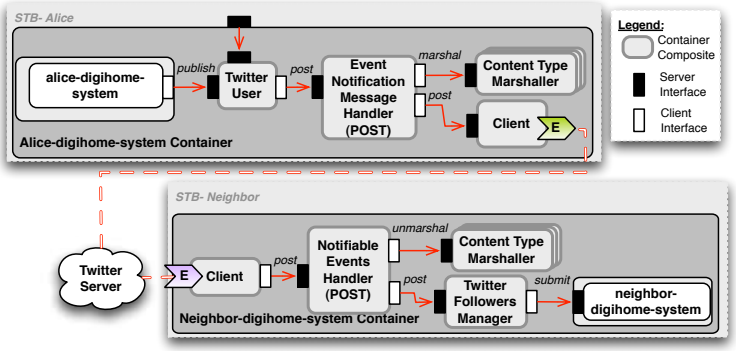

Figure 4: Social Bindings Service and Reference Architectures

Figure 4 depicts the containers implementing the Twitter bindings for the `neighbor-digihome-system` and `alice-digihome-system` composites (cf. Figure 3). In this architecture, we benefit from the Twitter RESTful API by building the social bindings upon REST bindings [RRSC10]. Therefore, in the server side (upper part of Figure 4), we observe that the `Event Notification Message Handler` component serializes events via the `Content Type Marshaler`. The `Event Notification Message Handler` posts the events using the `Client` component, which encapsulates the communication mechanism (*i.e.*, HTTP in the case of Twitter). The `Twitter User` component orchestrates the tweets publication process by providing the required information for the Twitter requests, such as user name, password, and event types.

On the other hand, the client side (low part of Figure 4) uses a `Twitter Followers Manager` component to periodically retrieve the recent tweets on the followed Twitter accounts. This component filters the tweets by using the metadata that includes event type. In this side, we also reuse the REST bindings functionality for communication (via the `Client component`) and message processing (via the `Notifiable Events Handler` and `Content Type Marshaler` components). Thus, the proposed architecture for these components modularizes the Twitter functionalities derived from REST principles [Fie00], which includes the addressing, access, and message representations. This modularity fosters the component reuse and the flexibility to choose different component implementations.

5.3 Validation

Implementation Details. We have integrated our social bindings into the FRASCATI platform (version 1.2). These bindings are based on the COMANCHE web server[4]. Both, FRASCATI and COMANCHE, are based on the FRACTAL component model and use the JULIA[5] implementation of the FRACTAL runtime environment [BCL+06].

Performance Results. To evaluate the social bindings, we implemented the event exchanges described in the scenario 2 (cf. Section 2) using the Twitter bindings.

Table 1 summarizes the results we obtained when executing several physical configurations for the scenario 2 using HTTP and Twitter as communication mechanisms for events exchange.

Event Providers Configuration	Event Provider	Event Representation				
		Object (ms)	JSON (ms)	XML (ms)	JSON (ms)	XML (ms)
a) 1 Local Provider	N/A	209.8	221	228	948.2	951.55
b) 1 External Provider	MacBook Pro	281.7	293.8	300.8	965.2	971.44
	Interaction	HTTP			Twitter	

Table 1: Performances of Twitter Bindings in terms of Latency

As it can be seen, the indirection introduced by the micro-blogging service increments the event exchange by approximately 350% compared to the configuration *a* for the two mecha-

[4] COMANCHE web server: http://fractal.ow2.org/tutorials/comanche.html
[5] JULIA: http://fractal.ow2.org/julia

nisms with the JSON representation. However, despite of this overhead, the simplicity of this interaction mechanism makes it still a suitable alternative to share events in an asynchronous way. Furthermore, the event notification rests less than a seconds, which is a reasonable overhead to communicate relevant situations in the context of smart homes.

6 Related Work

In this section we present two kinds of related work: proposals that benefit from micro-blogging services and approaches dealing with discovery in pervasive environments.

Pervasive Communications. In the literature, we find several middleware solutions, such as INDISS [BI05] (*INteroperable DIscovery System for network Services*) and ReMMoC [GBS05] (*Reflective Middleware for Mobile Computing*), that provide interoperability for discovery protocols. The problem with some of these solutions is that they are not flexible enough for using different interaction mechanisms when the services are discovered. Furthermore, they have been designed for working on devices with restricted capabilities, thus limiting the range of applications that can use them. Although in our solution we do not support discovery protocol interoperability, by benefiting from FRASCATI platform extensibility, we can easily add new pervasive bindings when required. Furthermore, by integrating BINJIU as SCA pervasive bindings, we promote its usage in different kinds of applications, not only the mobile ones.

Regarding approaches that exploit specifically UPnP, we find some solutions [FT09, MSS+09] in domestic environments (*i.e.*, automated homes) that aim to provide control on home appliances. In particular, this kind of approaches uses UPnP-based architectures to enable the simple and transparent access of different resources present in the environment. Some improvements are included in UPnP, such as service configuration persistence and enhanced management options. However, these solutions are difficult to apply in the development of new applications requiring discovery capabilities, while the combination of BINJIU and SCA make easier the incorporation of the UPnP technology when required.

Social Collaborations. To the best of our knowledge, there is no proposal dealing with event propagation via micro-blogging services in pervasive environments. However, we benefit from the ideas proposed in some of these approaches in order to improve our solution. For example, TwitterStand [SST+09] and TweetSieve [GGB+09] are systems able to extract relevant news from a noisy medium, such as Twitter. Thanks to the modular architecture of social bindings, we easily incorporate these techniques to support a smarter event selection. Furthermore, these techniques can also be used as a stabilization mechanism to avoid the notification of not relevant events for the system and participants.

In [MMC09], the authors propose a social networking middleware that combines social and physical proximity relationships between mobile users in order to suggest them potential people with whom they could perform activities of common interest. The mechanisms proposed in this work for determining the social and physical proximity can be also integrated in the social bindings in order to infer events that could be relevant for the system and home inhabitants.

7 Conclusion

In this paper, we reported on the implementation of new bindings for an SCA-based distributed system in a home pervasive environment. In particular, we demonstrated their integration into the FRASCATI platform, an open source implementation of the Service Component Architecture (SCA) standard [SMF+09]. Relying on well-accepted technologies, these new bindings increase capabilities of SCA in two complementary ways.

Using UPnP discovery and communication capabilities, pervasive bindings supply spontaneous communications between SCA components or legacy UPnP devices. On the other hand, social bindings use the Twitter micro-blogging service in order to disseminate applications events to end users. Two scenarios illustrating the integration of these capabilities into the FRASCATI platform have been presented to motivate and validate our proposals.

Future work includes to add support for new pervasive (*e.g.*, SLP [GPVD99] and Avahi[6]) and social bindings (*e.g.*, Facebook and OStatus[7]). We will take advantage of FRASCATI runtime reconfiguration capabilities to dynamically integrate these new bindings in SCA applications. Regarding UPnP, we plan to adapt the UPnP bindings for making them compliant with the Android platform. Moreover, we will integrate new UPnP certified profile implementations, in order to easily interact with home appliances. On the other hand, the identity management in a home environment represents an interesting issue. We already have to deal with preferences, private information (like Twitter login), and therefore, we need to provide a mechanism to ensure identity in pervasive environments.

Bibliography

[BCL+06] E. Bruneton, T. Coupaye, M. Leclercq, V. Quéma, J.-B. Stefani. The FRACTAL component model and its support in Java. *Software: Practice and Experience – Special issue on Experiences with Auto-adaptive and Reconfigurable Systems* 36(11-12):1257–1284, Aug. 2006. John Wiley & Sons.

[BI05] Y.-D. Bromberg, V. Issarny. INDISS: interoperable discovery system for networked services. In *Middleware '05: Proceedings of the ACM/IFIP/USENIX 2005 International Conference on Middleware*. Pp. 164–183. Springer-Verlag New York, Inc., New York, NY, USA, 2005.

[Fie00] R. T. Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California, Irvine, 2000.

[FT09] G. B. de Freitas, C. A. C. Teixeira. Ubiquitous services in home networks offered through digital TV. In *SAC '09: Proceedings of the 2009 ACM symposium on Applied Computing*. Pp. 1834–1838. ACM, New York, NY, USA, 2009.

[GBS05] P. Grace, G. S. Blair, S. Samuel. A reflective framework for discovery and interaction in heterogeneous mobile environments. *SIGMOBILE Mob. Comput. Commun. Rev.* 9(1):2–14, 2005.

[GGB+09] M. Grinev, M. Grineva, A. Boldakov, L. Novak, A. Syssoev, D. Lizorkin. Sifting micro-blogging stream for events of user interest. In *SIGIR '09: Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*. Pp. 837–837. ACM, New York, NY, USA, 2009.

6 AVAHI, a Zeroconf implementation: http://avahi.org

7 OSTATUS standard: http://ostatus.org

[GPVD99] E. Guttman, C. Perkins, J. Veizades, M. Day. Service Location Protocol, Version 2. RFC 2608 (Proposed Standard). http://tools.ietf.org/html/rfc2608, june 1999.

[LSDS10] F. Loiret, L. Seinturier, L. Duchien, D. Servat. A Three-Tier Approach for Composition of Real-Time Embedded Software Stacks. In *13th International SIGSOFT Symposium on Component-Based Software Engineering (CBSE)*. LNCS. Springer, June 2010.

[Mak09] K. Makice. *Twitter API: Up and Running Learn How to Build Applications with the Twitter API*. O'Reilly Media, Inc., 2009.

[McF07] P. McFedries. Technically Speaking: All A-Twitter. 44:84–84, 2007.

[MMC09] S. B. Mokhtar, L. McNamara, L. Capra. A middleware service for pervasive social networking. In *M-PAC '09: Proceedings of the International Workshop on Middleware for Pervasive Mobile and Embedded Computing*. Pp. 1–6. ACM, New York, NY, USA, 2009.

[MSS+09] L. F. Maia, D. F. S. Santos, R. S. Souza, A. Perkusich, H. Almeida. Seamless access of home theater personal computers for mobile devices. In *SAC '09: Proceedings of the 2009 ACM symposium on Applied Computing*. Pp. 167–171. ACM, New York, NY, USA, 2009.

[Ope07] Open SOA. Service Component Architecture Specifications. Nov. 2007.

[OSG04] OSGi Alliance. Listeners Considered Harmful: The Whiteboard Pattern. Aug. 2004.

[RHT+10] D. Romero, G. Hermosillo, A. Taherkordi, R. Nzekwa, R. Rouvoy, F. Eliassen. RESTful Integration of Heterogeneous Devices in Pervasive Environments. In *Proceedings of the 10th IFIP International Conference on Distributed Applications and Interoperable Systems (DAIS'10)*. LNCS 6115, pp. 1–14. Springer, June 2010.

[RRSC10] D. Romero, R. Rouvoy, L. Seinturier, P. Carton. Service Discovery in Ubiquitous Feedback Control Loops. In *Proceedings of the 10th IFIP International Conference on Distributed Applications and Interoperable Systems (DAIS'10)*. LNCS 6115, pp. 113–126. Springer, june 2010.

[SMF+09] L. Seinturier, P. Merle, D. Fournier, N. Dolet, V. Schiavoni, J.-B. Stefani. Reconfigurable SCA Applications with the FraSCAti Platform. In *Proceedings of the IEEE International Conference on Services Computing (SCC'09)*. Pp. 268–275. IEEE Computer Society, Sept. 2009.

[SST+09] J. Sankaranarayanan, H. Samet, B. E. Teitler, M. D. Lieberman, J. Sperling. TwitterStand: news in tweets. In *GIS '09: Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. Pp. 42–51. ACM, New York, NY, USA, 2009.

[UPn08] UPnP Forum. UPnP Device Architecture 1.1. http://www.upnp.org/resources/documents.asp, Oct. 2008.

[Zig10] ZigBee Alliance. ZigBee Technical Documents. http://www.zigbee.org, 2010.

[ZR09] D. Zhao, M. B. Rosson. How and why people Twitter: the role that micro-blogging plays in informal communication at work. In *GROUP '09: Proceedings of the ACM 2009 international conference on Supporting group work*. Pp. 243–252. ACM, New York, NY, USA, 2009.