Design and Implementation of the Open Connectivity Services Framework

Luis Diez[1], Olivier Mehani[2], Lucian Suciu[3], and Ramón Agüero[1]

[1] Universidad de Cantabria, Santander, Spain
{ldiez,ramon}@tlmat.unican.es
[2] NICTA, Sydney, Australia
olivier.mehani@nicta.com.au
[3] Orange Labs, Rennes, France
lucian.suciu@orange-ftgroup.com

Abstract. The Open Connectivity Services (OConS) framework is currently being defined within the framework of the SAIL project. Its main objective is to offer adaptive connectivity services to seamlessly address user and service requirements while complying with operator policies and dealing with the heterogeneous and changing network conditions of the future Internet. This paper describes a realization of this framework. It supports flexible integration of both legacy and novel mechanisms and protocols by mapping them to three abstract functional entities: information, decision and execution elements. Organization of these entities is done through an orchestration process to combine and integrate the various mechanisms into a full service for the user. We introduce the OConS protocol used for communication between the presented entities as well as the role played by the orchestration process. We then present the concrete example of a testbed based on this implementation, which shows the feasibility and effectiveness of the proposed approach.

Key words: Future Internet, Connectivity Services, Access Selection, Implementation, Demonstration

1 Introduction

Within the most promising research forums on Future Internet is an ongoing debate on the appropriateness of clean-slate approaches. Such approaches allow to sever the ties with the limitations of the currently deployed technologies, and support novel solutions better tailored to current needs. Conversely, following a less radical evolutionary path would allow for easier incremental deployment while avoiding the need for an altogether impossible *flag day*. So far, no clear direction has been agreed upon. A key element in working towards such an answer, however, is rooted in experimental initiatives aimed at testing the feasibility of one or the other approach.

The Scalable and Adaptive Internet Solutions (SAIL) project is one research initiative in the Future Internet realm [1], with a strong emphasis on integration and prototyping. It focuses on three core concepts: the use of information-centric networking (NetInf), the possibilities which appear with the advent of cloud networking concepts extending virtualization techniques to the network itself (CloNe) and the provision of seamless integration of legacy and future technologies into open connectivity services (OConS). The present paper focuses on the latter. Indeed, due to their rigidity and lack of large-scale dynamic reconfiguration capabilities, legacy connectivity services have increasing problems coping with the rapid evolution of communication requirements and patterns. New solutions are needed, which are dynamic, flexible and open, so as to be able to adapt to both the requirements of the different users and services as well as the ever-changing conditions of nowadays' heterogeneous networks.

Resources to be controlled and the connectivity issues which this raises are numerous and varied, each of them with their particular characteristics, procedures, algorithms and protocols. A monolithic design would therefore not fare well in dealing with such issues. The approach proposed within OConS is that of a highly modular architecture flexible enough to cope with these conditions [2]. A preliminary version of this framework was presented in [3]. The services are structured into to three complementary phases (information gathering, decision making and execution enforcement), and the associated entities performing these tasks. Lately the focus has been put on the elements needed to organize and control these entities as well as the various procedures required to achieve this orchestration.

As already mentioned, one of the key aspects in this line of research is the assessment, over real platforms, of the proposed architectures, procedures and protocols. This is the main focus of this paper; it presents the activities towards an implementation of the OConS framework as well as the use of this implementation into a real testbed, and uses it as an evaluation of the feasibility of the proposed open connectivity concepts. The implementation of the OConS architectural entities is detailed, as well as the protocol supporting signalling and communication between them. A scenario where a wireless station needs to select its access network is then introduced to illustrate the use of the framework. It is then extended to show how the orchestration process is used to support on-the-fly integration of other mechanisms such as a dynamic mobility management scheme which uses tunnels to handle mobility at the network level.

This paper is structured as follows. Section 2 introduces the OConS architecture and presents implementation details of the entities and protocol. Section 3 presents the testbed setup, identifying the OConS components and their interactions. Proof-of-concept experimental results with this setup are reported in Section 4, while the paper is concluded in Section 5, which also covers directions for future work.

2 OConS Architecture

In this section, an overview of the OConS design and architecture is first presented, reminding the reader to its basic functional components and those orchestration elements which allow their coordination. We then detail their im-

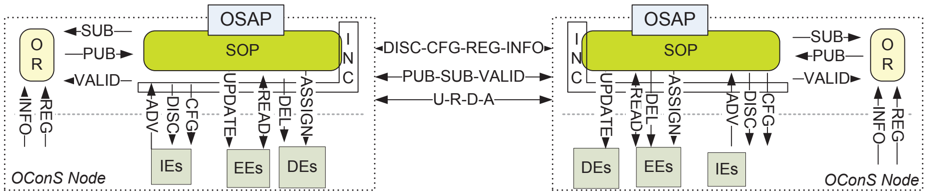


Fig. 1. Interactions between OConS elements. All communication goes through the INC, except for service requests through the OSAP, and the tighter SOP–OR link.

plementation and the needed communication protocol, which together form the basis of a common library to ease the integration of existing mechanisms within the OConS framework.

2.1 Architecture and Design

All conceptual elements in the OConS framework, as well as their interactions, is shown in Fig. 1 and described thereafter.

Basic entities. The goal of OConS is to provide an open framework to support any type of connectivity service. This requires a rather high level of abstraction in defining its constituent components. In order to manage this intrinsic openness, three elementary and generic entities have been defined [2], [3].

Information Entities (IE) take care of gathering the relevant information, and provide it to interested parties following the generic data model defined in [4].
Decision Entities (DE) take data from the IEs and run a decision algorithm in order to determine which actions need be taken in light of this information.
Execution Entities (EE) enforce the decisions made by the DE and optionally report the status details of the execution of the task.

Levels of orchestration. To support dynamic configuration, combination and instantiation of mechanisms and services, the instances of these three basic components need to be coordinated. This is the role of the *orchestration function.* For the sake of flexibility, it must cover the entire communication scope, thus bringing about three main orchestration levels [4].

Link Connectivity Services do not span further than one hop and are closely related to the physical and data-link layers and their operational parameters.
Network Connectivity Services affect the network and transport layers and are independent of end user applications. They usually involve two or more nodes.
Flow Connectivity Services are also related to the network and transport layers, but show a tighter link with the applications and services they support.

Orchestration functionalities. The orchestration function is a key enabler for the applications, or OConS users, to request specific services from the framework. They respond to these requests by appropriately configuring the relevant OConS components.

The Service Orchestration Process (SOP) is in charge of coordinating and overseeing all the orchestrations tasks. It keeps track of the OConS mechanisms which are available and what they can offer. These mechanisms are defined as a combination of entities (*i.e.*, IEs and EEs) which a particular DE requires in order to inform and enforce a decision.

The Orchestration Service Access Point (OSAP) is the external interface of OConS to its users. It is the point of entry for user requests (*i.e.*, connectivity requirements), and that through which OConS can report back about the available capabilities or status.

The Orchestration Registry (OR) acts as a repository of all the entities within a node, as well as those discovered on other nodes. These entities can be combined to instantiate various mechanisms, which are also kept track of within the OR.

The Orchestration Monitor (OM) retains the statuses of the OConS components and mechanisms launched within different OConS-enabled nodes.

OConS messaging. The previously introduced OConS entities and functionalities interact with each other by exchanging *OConS messages*. Dealing with a distributed set of components for which connectivity might not be fully established and vary widely, it is important to offer a communication facility which properly decouples messaging from the underlying communication technology. This is done through an inter-node communication hub (INC) which is in charge of relaying the messages to its destination, either locally or remotely, over whichever *transport method* (*e.g.*, local IPC, TCP/IP or Ethernet) is available to reach it. This is depicted in Fig. 1.

While messages to node-local entities (identified by their locally-attributed *entity ID*) can be passed directly, those to remote destinations are encapsulated in *OConS packets* which are wrapped into the relevant headers for the selected medium. The INC maintains a unique *node ID* which is mapped to the appropriate locator for the selected transport method. To ensure global uniqueness, these node IDs must be assigned following a random algorithm with good uniqueness properties. An example of such is that proposed for Unique Local IPv6 Addresses [5].

2.2 Framework Implementation

We now introduce our implementation of these functional blocks and the corresponding orchestration logic. The communication model is based on *Request/Response–Notification* messages. For this purpose, an *OConS protocol library* has been implemented in C/C++ to provide low-level functionalities allowing the use of the OConS protocol and interfaces. The detailed description of interfaces, messaging and protocols is given in [4], but we summarize it here.

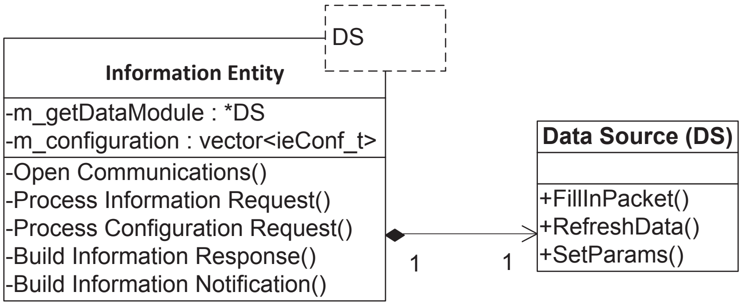

DS
Information Entity
-m_getDataModule : *DS
-m_configuration : vector<ieConf_t>
-Open Communications()
-Process Information Request()
-Process Configuration Request()
-Build Information Response()
-Build Information Notification()
1
1
Data Source (DS)
+FillInPacket()
+RefreshData()
+SetParams()

Fig. 2. Structure of the generic IE object provided by the OConS protocol library.

OConS protocol library. The library implements the common OConS communication facilities to enable both inter-entity and inter-node communications. It manages direct communication with the local INC and provides helper functions for message manipulation. These helpers are in charge of the encapsulation within the OConS headers based on which forwarding to the relevant entity or node through the INC. Message content manipulation is similarly abstracted from the on-wire TLV (Type, Length, Value)-based data representation from the entity implementation. Additionaly, the library provides high-level procedures to support the OConS orchestration functionalities, such as mechanism identification, entities registration or bootstrap management.

This library aims at offering a common background allowing the developer to implement the entities needed for a new connectivity mechanism, as well as to setup appropriate orchestration between those entities. From a system point of view, each entity, as well as the INC, appears as a separate process. The INC waits for connections of the local entities using IPC mechanisms, and listens on the various supported remote transport methods for OConS messages. At start-up, the entities connect to this local IPC and, after an initialisation phase where its entity ID is determined and its existence is recorded in the local OR, it can exchange OConS messages with both local and remote entities through the INC.

Generic Information Element template. As they follow a rather regular pattern, the behaviour of IEs is easily generalised. The library therefore provides a C++ template for such a generic IE, which we provide as an example of how it can be used. Fig. 2 shows the structure of this object. It assumes that each IE is usable and configurable by one or more DEs requiring the exposed information. The generic IE is able to act both reactively, upon request from a DE, or proactively, sending it update notifications periodically.

The generic IE includes a reference to an object responsible for providing the particular information for a given instance. This exemplifies the modularity of the design, where the OConS layer is shared between all entities, but the actual operation, such as data collection in the case of an IE, is specific to the mechanism being provided. In this case, the outer class implements the *Information Entity* interface, while the *Data Source* (DS) class collects the data from actual sources such as hardware drivers or software systems. It is worth noting that the DS should be aware of the relevant parts of the OConS data model and properly represent the collected information according to this model. It is the role of the `FillInPacket()` method to do so.

3 Integration Tesbed

We recall that the main goal of this work is the empirical assessment of the feasibility of the OConS architecture, so as to fill the gap between architectural descriptions and specifications, and its implementation in real platforms. As the proposed framework is intended to cope with a wide range of different connectivity technologies and protocols with varied features, it is crucial to confirm that is it open enough to be effective. We present our experimental testbed work towards this in this section.

While one driving goal of the OConS framework is its openness, this is not necessarily the case of the underlying technologies. This limits the choices when developing such an experimental testbed to those pieces of hardware which drivers offer the possibility to access and manipulate low-level parameters. Moreover, it is desirable to focus on hardware available off the shelf as this is representative of the vast majority of already deployed equipments which OConS is intended to work atop. Considering these intrinsic limitations, the platform we developed is based on the IEEE 802.11 technology. The presence of heterogeneous networks is emulated by deploying multiple access points, configured to operate on orthogonal channels.

This approach, which undoubtedly presents some limitations, however also has clear advantages. Wi-Fi hardware drivers such as MadWifi[1] provide ample access to information and control interfaces, while `hostapd`(8)[2] makes it possible to use regular computers to deploy access points, therefore easing the deployment of modifications on the network side.

The proposed testbed implements parts of the overall OConS reference scenario [2], presented in Fig. 3. It comprises two access points and an end-user terminal. All three implement the OConS framework through the protocol library introduced in Section 2. The connectivity services available in this testbed include procedures for access selection and support for dynamic mobility management. Though often crudely merged in the literature, both mechanisms are clearly different: access selection takes place at the mobile node and usually relies on signal quality information from the access elements to decide which to use, while mobility management leverages in network anchor points to provide support for session continuity throughout access changes through the use of IPv6 tunnels [6]. Under the orchestration of the OConS framework these two mechanisms are able to collaborate, resulting in an improved and more flexible service.

[1] http://madwifi.org

[2] http://hostap.epitest.fi/hostapd/

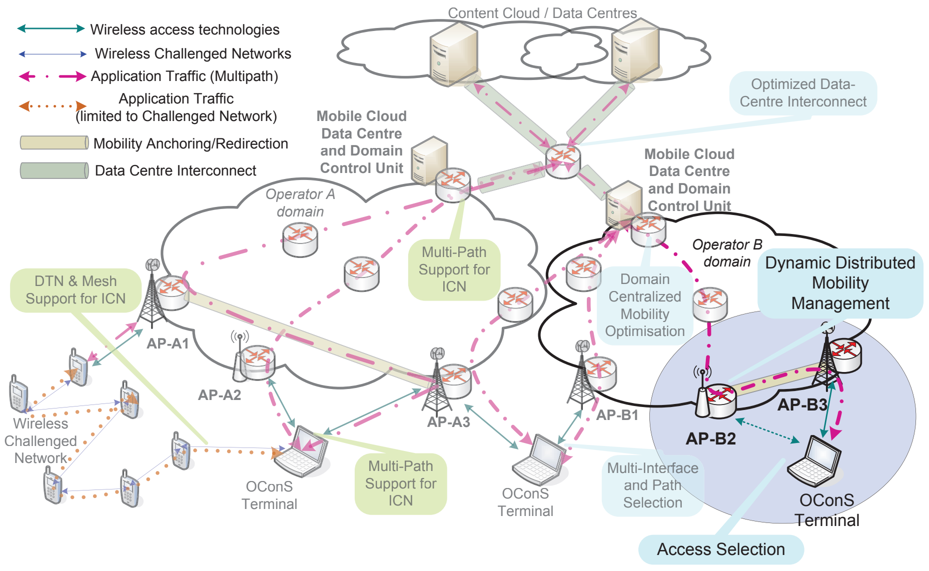

Wireless access technologies
Wireless Challenged Networks
Application Traffic (Multipath)
Application Traffic (limited to Challenged Network)
Mobility Anchoring/Redirection
Data Centre Interconnect
Content Cloud / Data Centres
Optimized Data-Centre Interconnect
Mobile Cloud Data Centre and Domain Control Unit
Mobile Cloud Data Centre and Domain Control Unit
Operator A domain
Operator B domain
Multi-Path Support for ICN
DTN & Mesh Support for ICN
Dynamic Distributed Mobility Management
Domain Centralized Mobility Optimisation
AP-A1
AP-A2
AP-A3
AP-B1
AP-B2
AP-B3
Wireless Challenged Network
OConS Terminal
Multi-Path Support for ICN
OConS Terminal
Multi-Interface and Path Selection
OConS Terminal
Access Selection

Fig. 3. The OConS overall scenario [2]. It includes mechanisms for end-user mobility such as access selection and distributed mobility management (bottom right).

The presented scenario supports a dynamic activation of mobility anchors at the access routers, so that a mobile node (MN) can use direct IP routing for sessions initiated while using the current access router, whilst forwarding traffic from the previous access routers, acting as temporary mobility anchors. The OConS approach, unifying these mechanisms into a common framework, enables them to interact—for example by allowing some triggering events (*e.g.*, layer-2 hand-offs) to enrich the decisions or by considering other pieces of information describing the connectivity context—to provide a finer-grained and more accurately controlled support of session continuity while providing network access through the current best access element.

Following the architectural description from Section 2, each of the nodes implements orchestration functionalities which coordinate the operation of their functional elements. One OConS mechanisms has been developed which links both connectivity procedures. It is embodied as one DE located at the MN, one EE similarly located and takes care of the layer-2 handover (it also triggers the subsequent layer-3 handover) and various IEs, providing contextual information so as to enrich the decision making process. The specific IEs and the information they provide is described below.

Radio channel quality (IE_{LQ}) is implemented as a scanning module which informs any subscriber entity about the current quality (SNR or RSSI) of each access point from a list of available and usable networks. This entity is located on the MN.

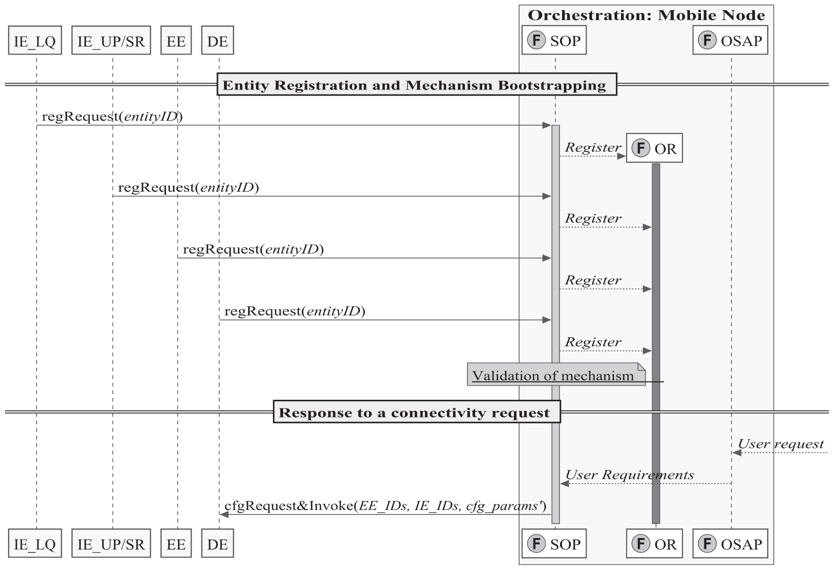


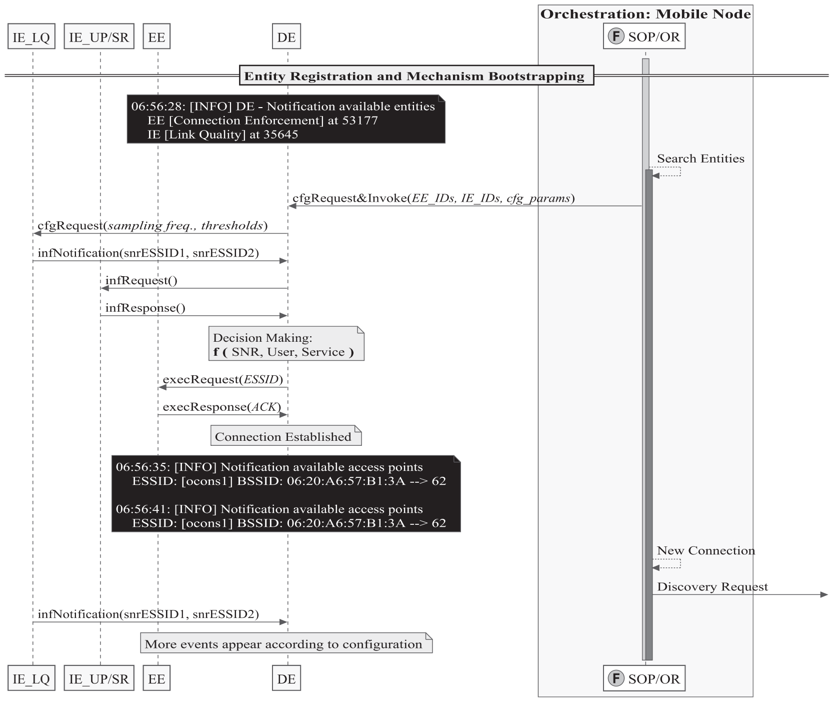
Fig. 4. Basic OConS Orchestration. Entities register to the SOP and OR via the INC. When all entities composing a mechanism are ready, the DE gets a notification.

User profile (IE_{UP}) stores information about the type of user requesting a connectivity service. This is used so the DE can distinguish the privileges of a subscriber and maintain accounting tasks. This entity is also placed at the MN.

Service requirement (IE_{SR}) , also located in the AN, tags each type of data service to allow their particular management according to their specific requirements (*e.g.*, video streaming or file transfer).

Traffic load (IE_{TL}) stores information about the current traffic load at the access router, where it is located, so as to support load balancing.

Once all the entities have started up, the first orchestration steps take place. They comprise registration (Fig. 4), search and configuration (both in Fig. 5 in section 4). During this first discovery procedure, all entities register to the SOP by means of a *regRequest* primitive together with their information. This comprises both the entity type and entity ID as indicated in Fig. 4 (in which the circled F's indicates functionalities which have been implemented jointly with the INC), which are afterwards stored within the OR. Anytime a new entity registers, the SOP checks whether some new mechanisms have become available (that is, all the entities composing the mechanism are registered). If a new mechanism can be executed, the SOP informs the appropriate DEs about how to reach the rest of entities. At the bootstrapping the only entity configured is the DE by setting the IE/EEs which it has to use to perform the mechanism. The latter configurations (DE–IE) are discussed on the next section.


Orchestration: Mobile Node
IE_LQ
IE_UP/SR
EE
DE
F SOP/OR
Entity Registration and Mechanism Bootstrapping
Search Entities
cfgRequest&Invoke(EE_IDs, IE_IDs, cfg_params)
cfgRequest(sampling freq., thresholds)
infNotification(snrESSID1, snrESSID2)
infRequest()
infResponse()
Decision Making:
f (SNR, User, Service)
execRequest(ESSID)
execResponse(ACK)
Connection Established
New Connection
Discovery Request
infNotification(snrESSID1, snrESSID2)
More events appear according to configuration
IE_LQ
IE_UP/SR
EE
DE
F SOP/OR

```
06:56:28: [INFO] DE - Notification available entities
    EE [Connection Enforcement] at 53177
    IE [Link Quality] at 35645
```

```
06:56:35: [INFO] Notification available access points
    ESSID: [ocons1] BSSID: 06:20:A6:57:B1:3A --> 62

06:56:41: [INFO] Notification available access points
    ESSID: [ocons1] BSSID: 06:20:A6:57:B1:3A --> 62
```

Fig. 5. Example sequence of OConS messages showing the search and registration of compliant IEs, their use by a DE, and the execution of the decision by the EE.

In a similar way, any OConS node can also carry out remote discovery provided there exists a common underlying transport method. In this case, the SOP starts this remote discovery if some mechanisms required depend on remote entities. It is interesting to highlight the fact that the current implementation incorporates this degree of flexibility and thus allows a mechanism to be defined by entities spanning different nodes.

4 Proof of Concept

This section presents some of the tests carried out on the testbed presented in section 3. The objective of these tests was the assessment of the correctness of the implementation. The points which have been verified to work at the time of this writing are presented below. Our focus was on the correctness of the OConS protocol implementation, and its usability to compose a distributed mobility management service from the available basic elements. Fig. 5 shows a sequence of OConS messages exchanged during one of the tests. It is based on the output of a monitoring and configuration GUI application which has been developed to ease the process of tracking the message exchanges.

The following verifications have been performed.

Entity registration with INC. This point has been previously discussed in Section 3. When started, each entity contacts the locally running INC to make it aware of their existence. This was previously illustrated in the top part of Fig. 4.

Mechanism dependency check. Upon receiving a registration request, the SOP checks the availability of a mechanism. If all entities to compose it are available, the INC informs the DE about the entity types and location, as the bottom part of Fig. 4.

Inter-entity communication for mechanism composition. The DE is able to correctly configure the appropriate IEs, requesting information or subscribing so as to be periodically notified. The EE properly responds to execution requests from the DE, providing information about the current execution (this was checked by removing networking privileges from the EE process and ensuring that it reports a connection failure).

Remote discovery. After the connection the INC starts the remote discovery service, which allows the DE to become aware of the remote entities which might extend and enhance the service, in particular with the possibility of using the IE_TL mentioned in Section 3.

Algorithms adaptability to context changes. Once the remote discovery has been carried out, the DE is able to adapt its operation. The selection algorithm can take into account new information which can be gathered from the remote entities, and react accordingly.

With this testbed setup, we have demonstrated that OConS allows for the integration of a very flexible platform. It enables easy extension and configuration, as well as the integration of new functionality, in the form of new entities which conforms to the framework. Moreover, a GUI allowing to monitor the information exchange and offering possibility of parsing their contents has been developed. This therefore complements the OConS protocol library with an invaluable debugging tool.

5 Conclusions

In spite of the relevance that the research on the Future Internet realm has recently gained, there is a clear need to foster the assessment of the corresponding architectures, protocols, techniques and algorithms over real testbeds. This is a crucial point to facilitate their forthcoming rollout. It is one of the leitmotifs of the SAIL project, which is supporting implementation-related activities. The Open Connectivity Services framework, which is one of the main components of the SAIL solution, is also pursuing this goal. This paper has presented an implementation of the OConS framework and its deployment over a real testbed. It described the OConS architectural entities and introduced the role played by the orchestration process, as a means to enable the integration of various mechanisms by the composition of various functional elements.

As an illustrative example we have shown the integration of an enhanced access selection process with a dynamic mobility management procedure, using the possibilities which are brought about by the OConS framework. Due to the intrinsic limitations of available technologies (especially considering the need to develop functionalities on the network side), we have used a Wi-Fi based testbed, although the implementation is generic enough so as to be used on top of any other technology.

Based on this preliminary demonstration, several lines of action have been opened. First of all we aim at introducing additional mechanisms (for instance, flow management and multi-path), orchestrating them to offer a better service to the OConS user. Besides, the current testbed will be extended so as to integrate different mobility management solutions. We will also foster the integration with the other SAIL components (NetInf and CloNe). Furthermore, we intend to release the OConS protocol library described here to let the community take advantage of the service composition flexibility afforded by the OConS framework.

Acknowledgments

This work is being carried within the SAIL Project under Grant Agreement Number 257448 of the Seventh Framework Programme of the European Union.

References

[1] SAIL project. *Description of Project Wide Scenarios and Use Cases.* Deliverable FP7-ICT-2009-5-257448-SAIL/D2.1(D-A.1). EC Information Society Technologies Programme, Apr. 2011.

[2] SAIL project. *Architectural Concepts of Connectivity Services.* Deliverable FP7-ICT-2009-5-257448-SAIL/D-4.1(D-C.1). EC Information Society Technologies Programme, July 2011.

[3] R. Agüero et al. "OConS: Towards Open Connectivity Services in the Future Internet". In: *MONAMI 2011, Third International ICST Conference on Mobile Networks & Management.* Ed. by S. Sargento and R. Agüero. Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering. Aveiro, Portugal: Springer-Verlag Berlin, Sept. 2011.

[4] SAIL project. *Architecture and Mechanisms for Connectivity Services.* Deliverable FP7-ICT-2009-5-257448-SAIL/D4.2(D-C.2). EC Information Society Technologies Programme, July 2012.

[5] R. M. Hinden and B. Haberman. *Unique Local IPv6 Unicast Addresses.* RFC 4193. Fremont, CA, USA: RFC Editor, Oct. 2005.

[6] D. B. Johnson, C. E. Perkins, and J. Arkko. *Mobility Support in IPv6.* RFC 6275. Fremont, CA, USA: RFC Editor, July 2011.