An XDI-Based Approach to Represent and Exchange Data Between Federated Clouds

Antonio Celesti, Francesco Tusa, Massimo Villari and Antonio Puliafito
Dept. of Mathematics, Faculty of Engineering, University of Messina
Contrada di Dio, S. Agata, 98166 Messina, Italy.
e-mail: {acelesti, ftusa, mvillari, apuliafito}@unime.it

***Abstract*—Cloud providers need to manage and control their assets identifying, retrieving, and exchanging data about their virtual resources in different operating contexts. These tasks are not trivial at all and this leads cloud providers to design proprietary solutions for the management of their virtual resources and services. In this paper, considering IaaS clouds, we discuss an approach based on XDI for the representation of data associated to Virtual Machines (VMs). More specifically, we focus on a scenario including federated clouds renting VMs to other ones, where an exchange of related data is required.**

***Keywords*-Cloud Computing, Federation, Naming System, XDI, Higgings.**

I. INTRODUCTION

Nowadays, cloud providers supply many kinds of Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) to their users, e.g., common desktop clients, companies, governments, organizations, and other clouds. Such services can be arranged composing and orchestrating several Virtual Environments (VEs) or Virtual Machines (VMs) through hypervisors.

The overwhelming innovation of cloud computing is that cloud platforms can react to events internally rearranging the VMs composing their services pushing down management costs, and the interesting thing is that cloud users are not aware of changes, continuing to use their services without interruptions according to a priori Service Level Agreements (SLAs). For example, when a physical server hosting an hypervisor runs out of resources or is damaged, the cloud can decide to move or "migrate" one or more VMs into another server of the same cloud's datacenter acting as virtualization infrastructure. Further migrations can be triggered for many other reasons including power saving, service optimization, business strategy, SLA violation, security, etc. In addition, if we consider the perspective of cloud federation where clouds cooperate sharing computational and storage resources, a VM can migrate also into a server of another cloud's virtualization infrastructure. Another business model which can take place in federated scenarios might be the rent of VEs from a cloud to another.

Such a dynamic and continuously changing scenario involves not only cloud services and VMs, but also other cloud entities such as physical appliances and cloud users. All these entities need to be named and represented both in human-readable and in machine-readable way. Moreover, they need also to be resolved with appropriate data according to a given execution context. For example, as a VM needs to be identified by a name, it may happen that different entities (e.g., the cloud middleware itself, other federated clouds, cloud administrators, cloud users, etc) may be interested to resolve that name retrieving either data concerning general information on the VM (e.g., CPU, memory, kernel, operating system, virtualization format version, IP address, etc), data regarding processes running inside the VM,or data regarding the performance of the VM (e.g., used CPU and memory usage). In addition, the scenario becomes more complex if we consider the fact that these entities might hold one or more names and identifiers also with different levels of abstraction.

In order to discourage a possible evolving scenario where each cloud based on open source architectures might develop its own proprietary information management system with compatibility problems in the interaction among different cloud name spaces, in our previous work [1], we proposed a standard XRI-based approach for the designing of a seamless cloud naming system able to manage and integrate independent cloud name spaces, extending the OpenXRI libraries [2].

XRI, considered alone, does not support any data interchange mechanism between entities which want to exchange data each other according to their policies. In order to overcome this issue, the OASIS XDI Technical Committee developed the XRI Data Interchange (XDI) [3] technology.In this paper, we discuss how to apply XDI technology, using the Higgings framework [4], for the development of a federated IaaS cloud scenario, where each cloud needs to exchange data with other ones about rented VMs. More specifically, considering several clouds, each one managing its own VMs by means of XRI graphs, we will focus on an use case where a cloud lends VMs to another cloud, thence exchanging the related data (e.g., IP address, how to access the VM, features, performance, etc) in a secure way.

The paper is organized as follows: Section II provides a brief description of cloud name spaces. Section III describes the state of the art of naming systems and the most widely adopted solutions in distributed systems and in ubiquitous computing environments. In Section IV, we provide an

overview of the XDI technology motivating how it suits the management of cloud name spaces and data interchange between federated clouds. In Section V, we describe how to design an XDI-based data management system for a cloud federation scenario using the Higgings framework. In the end, in Section VI, we focus on how to represent resources and users in a cloud using XRI RDF graphs. Conclusions and remarks are summarized in Section VII.

II. Cloud Name Space Issues

In this Section, we briefly summarize the main cloud name space issues which have already been analyzed in [5]. Despite the internal cloud structure, we think cloud entities have many logical representations in various contexts. In addition, there are many abstract, structured entities (e.g., a distributed cloud-service built using other services, each one deployed in a different VE). These entities are characterized by a high-level of dynamism: allocations, changes and deallocations of VEs may occur frequently. Moreover, these entities may have one or more logical representations in one or more contexts. But which are the entities involved in cloud computing? In order to describe such entities, we introduce the generalized concept of *Cloud Named Entity* (CNE). A CNE is a generic entity indicated by a name or an identifier which may refer both to real/abstract and simple/structured entity. As depicted in Figure 1, examples of CNE may be a cloud itself, a cloud federation, a virtualization infrastructure, a server running an hypervisor, a VE, a cloud service, or cloud users including companies, governments, universities, cloud technicians, and desktop clients.

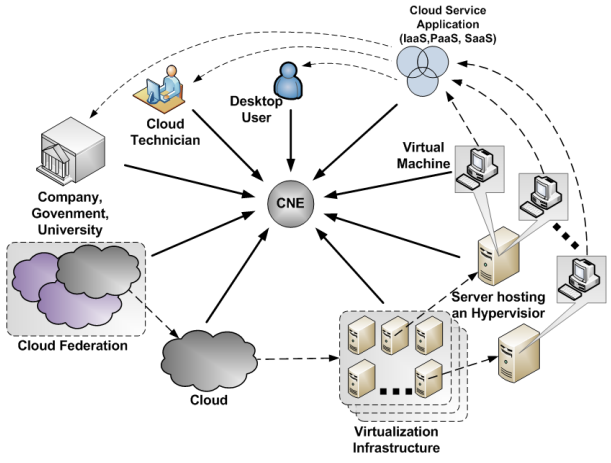

Figure 1. Examples of generic CNEs.

In our abstraction, we assume that a CNE is associated to one or more identifiers. As a CNE is subject to frequent changes holding different representations in various *Cloud Contexts* (CCNTXs), the user-centric identity model [6] seems to be the most convenient approach. We define a CCNTX as an execution environment where a CNE is represented by one or more identifiers and has to be processed. In this work, we assume a CNE is represented by one or more *CCNTX Resolver Server(s)*, which are servers returning data or services associated to a CNE in a given CCNTX. Figure 2 depicts an example of CNE associated with six identities within four CCNTXs. The target CNE holds identity 1, 2

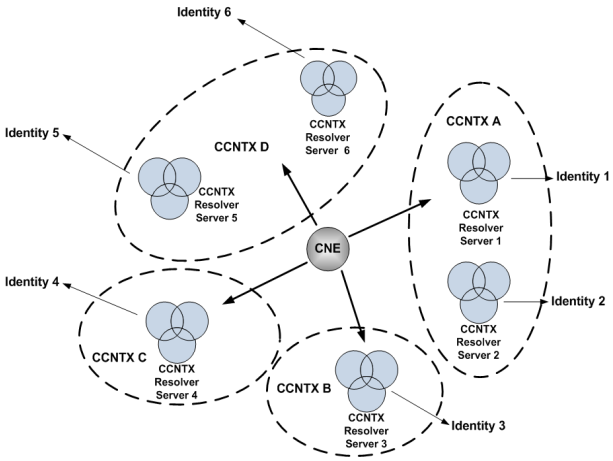

Figure 2. Examples of a generic CNE associated to several CCNTXs.

inside CCNTX A, identity 3 inside CCNTX B, identity 4 inside CCNTX C, and identity 5, 6 inside CCNTX D. We define a *Cloud Naming System (CNS)* as a system that maps one or more identifiers to a CNE. A CNS consists of a set of CNEs, an independent cloud name space, and a mapping between them. A cloud name space is a definition of cloud domain names. Instead, a name or identifier is a label used to identify a CNE. A client resolver which needs to identify a CNE in a given CCNTX performs a resolution task. Resolution is the function of referencing an identifier to a set of data or services describing the CNE in several CCNTXs.

III. Related Work and Background

Cloud computing is generally considered as one of the more challenging research field in the ICT world. It mixes aspects of Utility Computing, Grid Computing, Internet Computing, Autonomic computing and Green computing [7], [8]. Many authors are trying to describe what it exactly means, in terms of Utility Computing (as the Electricity Model, see [9]), its Economics and Benefits, and what are its Obstacles and Opportunities as the TOP 10 list reports in [10], [11]. Cloud, combined with statistical multiplexing, should increase resources utilization compared to traditional data centers, offering services below the costs of medium-sized datacenters and still making good profits (see [12]). In such new emerging environments, even though naming and resource location raise several issues, there have not been many related works in literature yet regarding naming

systems managing cloud name spaces, and DNS is still erroneously considered the "panacea for all ills". In fact, DNS presents some problems: it is host centric, unsuitable for complex data and services location, and it is not suited to heterogeneous environments. Possible improvements might come from the naming system works in high-dynamic, heterogeneous and ubiquitous environments. An alternative to DNS is presented in [13]. The authors propose a Uniform Resource Name System (URNS), a decentralized solution providing a dynamic and fast resource location system for the resolution of miscellaneous services. Nevertheless, the work lacks of an exhaustive resource description mechanism. With regard to naming system in ubiquitous computing, in [14] the authors propose a naming system framework for smart space environments. The framework aims to integrate P2P independent cloud naming systems with the DNS, but appears unfitted to be exported in other environments. In addition it aims to localize and identify an entity that moves from a smart space to another using as description mechanism the little exhaustive DNS resource records. A hybrid naming system that combines DNS and Distributed Hash Table (DHT) is presented in [15]. The authors adopt a set of gateways executing a dynamic DNS name delegation between DNS resolver and DHT node.

An interesting survey among different technologies for the Resource Discovery in Grid Environments has been done in [16]. The authors presented a valuable comparison among the P2P protocols ranging from Napster, Gnutella, CAN to Chord. It is interesting to notice the punctual evaluation (even taking into account the complexity of each one) of these protocol and their applicability in Grids. They mentioned that one of the main constrains in Grid is the scalability. Some of the protocols reported above are not really fully decentralized. (i.e., Napster) whereas others do not guarantee the operating in heterogeneous Grid environments. Other evaluations were conducted in [17] and [18]. Their assessments are about the possibility to use in Grid consolidated protocols for the Resource Discovery (RD) tasks. However many solutions adopted in Grid ([19]) along with the *advanced DHT* usage (see [20]), are not suitable in clouds at all. We can affirm that the level of heterogeneity in Clouds is higher of any Grid infrastructure. For that reason we cannot consider solutions embraced in Grid, but we have to look solutions widely used in distributed system as Internet (i.e., DNS approach). In our point of view concepts of systems heterogeneity and federation mechanisms need to be taken into account. Whether we consider the recent convergence of Web SSO systems in the Internet, in the last years we assisted to a wide use of OpenID [21]. It is considered as one of the widely digital identity protocol used for making *Federation* among web services. Providers that adopted such a technique range from AOL, BBC,Google, IBM, MySpace, Orange, PayPal, VeriSign, LiveJournal, to Yahoo [22]. The new version of OpenID, 2.0 was released to overcome some big issues [23]. The way for improving it, is to implement several clausals existing in the XRI Standard Specification [24], [25].

We can assume the XRI standard as a step over of the DNS protocol. All enterprises may continue in using their internal systems for cataloging resources and services, as LDAP, Active Directory (AD), owned database, etc; all these protocols are based on DNS. Our idea is to have an alternative to DNS, a kind of *advanced* DNS protocol, that is XRI, compliant with URI/URL approach able to overcome DNS limitations, also in terms of its representativeness. We can state that XRI might represent an useful abstraction of what already exists in the Internet. In particular we remind the XRI syntax and resolution infrastructure was designed explicitly for Internet-scale digital identity, and we are adopting it for enriching exchanged information in much more complex cloud scenarios maintaing its basic philosophy indispensable for the *Federated Digital Identity* management.

Regarding naming, name resolution, and service location in federated cloud environments, in our previous work [26], we highlighted the involved issues both debating a cloud name space analysis and proposing a generic theoretical cloud naming framework for the management of cloud name spaces. The cloud federation is a scenario where clouds establish a relationship in order to benefit new business advantages [27], for example renting single VM or whole cloud services to other clouds, for example when a cloud run out its computational and storage capabilities or when a cloud needs a service which is not able to allocate. In [28], considering such a cloud naming framework and several use-cases of the European Reservoir Project [29], we performed an analysis of the problems that such use-cases raise regarding the management of cloud name spaces, also debating how the aforementioned cloud naming framework could be adopted to manage naming and service resolution. As possible representation of the cloud naming framework we chose XRI [24] and the eXtensible Resource Descriptor Sequence (XRDS) [25] technologies. The major open source implementation of XRI is OpenXRI [2], which provides a basic Authority Resolution server along with java libraries for the development of XRI-based applications. Another interesting initiative is the Higgings Personal Data Service [4] framework developed by the Eclipse community. Higgings implements the XRI Data Interchange (XDI) [30], i.e., a generalized, extensible service for sharing, linking, and synchronizing structured data over the Internet using XRI-addressable RDF graphs. As well as XRI, XDI is under development by the OASIS [31] Technical Committee.

IV. XDI and Cloud Computing

In this Section, after a brief description the XDI technology, we motivate how it can help the cloud name space management and data interchange between federated clouds.

A. XDI Overview

XDI (XRI Data Interchange) is a generalized, extensible service for sharing, linking, and synchronizing structured data over the Internet and other data networks using XRI-addressable RDF graphs. XDI is under development by the OASIS XDI Technical Committee. The main features of XDI are: the ability to link and nest RDF graphs to provide context; full addressability of all nodes in the graph at any level of context; representation of XDI operations as graph statements so authorization can be built into the graph (i.e., a feature called XDI link contracts); standard serialization formats including JSON and XML; and a simple ontology language for defining shared semantics using XDI dictionary services. The XDI protocol can be bound to multiple transport protocols. The XDI TC is defining bindings to HTTP and HTTPS, however it is also exploring bindings to XMPP and potentially directly to TCP/IP. XDI provides a standardized portable authorization format called XDI link contracts. Link contracts are themselves XDI documents (which may be contained in other XDI documents) that enable control over the authority, security, privacy, and rights of shared data to be expressed in a standard machine-readable format and understood by any XDI endpoint. XDI enable to achieve a secure interchange of data between different software entities by means of secure communication channels. These channels can be secured through different techniques, including the Security Assertion Markup Language (SAML), based on the Identity Provider/Service Provider (IdP/SP) model.

RDF graphs are created using XRI, i.e., a standard syntax for identifying entities, regardless any particular concrete representation. The XRI system is similar to DNS, including a set of hierarchical XRI authorities but more powerful. The protocol is built on URI (Uniform Resource Identifiers) and IRI (Internationalized Resource Identifiers) extending their syntactic elements and providing parsing mechanisms. Particular types of URI are URN and URL. Since an URL is also an URI, the protocol provides a parsing mechanism from XRI to URL. Therefore XRI is also compatible with any URN domain. XRI supports persistent and reassignable identifiers by means of i-numbers (Canonical ID) and i-names (Local ID). It also provides four types of synonyms (LocalID, EquivID, CanonicalID, and CanonicalEquivID) to provide robust support for mapping XRIs, IRIs, or URIs to other XRIs, IRIs, or URIs that identify the same target entity. This is particularly useful for discovering and mapping to persistent identifiers as often required by trust infrastructures. XRI enable organization to logically organize entities building XRI RDF graphs. According to the XRI terminology, each entity in the graph is named authority. The protocol provides two additional options for identifying an authority: Global Context Symbols (GCS) and cross-references. Common GCS are "=" for people, "@" for organization, and "+" for generic concepts. For example the xri://@XYZ*marketing indicates the marketing branch of an organization named XYZ, where the "*" marks a delegation.

B. Why Does XDI suit Cloud Computing?

XDI meets the requirements of cloud name space management, data retrieval and data interchange especially in federated cloud environments. With XDI a cloud can keep different RDF graphs representing IaaS, PaaS, and SaaS. In addition, such a technology can be used for both identify and resolve VMs and whole *aaS by means of data retrieval mechanisms. For example, the cloud service provider may need to retrieve three types of information about a VM: general data (e.g., CPU, memory, kernel, operating system); real time performance data (e.g., amount of used CPU and memory used); real time data regarding an internal running process (e.g., the percentage of processed data). Moreover, considering IaaS federated clouds, each provider needs to exchange part of its data with other clouds. For example, let us consider two clouds: A and B. Cloud A, logically organize its own VMs by means of an XRI graph. As Cloud B has run out of resources, it require three VMs to cloud A. So that, cloud A instantiate the three VMs and update its XRI graph. Then, in order to allow cloud B to access the VMs, cloud A, after an authentication of cloud B, discloses how to access the VMs and related data. Authentication can be easily achieved using SAML Single Sign-On (SSO) mechanisms.

In addition, XDI might be used to logically represent instances of composed services. For example, let us consider a service instance composed of several elementary services each one running in a different VM. Thanks to XDI it is possible to create in the graph of a cloud an entry representing the service instance, linking the entries representing the VMs on which the service instance is made up. Other possible applications of XDI can regard for example the management of physical assets, clients, and so on.

V. How to Achieve XDI in Federated Cloud Using Higgings

Starting from the considerations of the previous Section, in the following we will point out a concrete scenario of cloud federation, specifically aimed to the IaaS context. In order to address either the problem of sharing information among clouds and achieving their authentication process, we rely on the XDI features. More specifically, our scenario takes advantage of the employment of the Higgings framework, that represents a Personal Data Service (PDS) including the implementation of XDI features.

In the first part of the Section we will introduce and describe the Higgings Framework, whereas in the second part we will present the reference scenario where our solution aims to address the IaaS cloud federation problem.

A. The Higgings Framework

Higgins is an open source project that aims to provide to individuals more control over their personal identity, profile and social network data.

The project is organized into three main areas:

1) **Active Clients**. An active client integrates with a browser and runs on a computer or mobile device.
   - *Higgins 1.X:* the active client supports the OASIS IMI protocol and performs the functions of an Information Card selector.
   - *Higgins 2.0:* the plan is to move beyond selector functionality to add support for managing passwords, Higgins relationship cards, as well other protocols such as OpenID. It also becomes a client for the Personal Data Store (see below) and thereby provides a kind of dashboard for personal information and a place to manage "permissioning" deciding who gets access to what slice of the user's data.
2) **Personal Data Store (PDS)** is a new work area under development for Higgins 2.0. A PDS stores local personal data, controls access to remotely hosted personal data, synchronizes personal data to other devices and computers, accessed directly or via a PDS client. It allows the user to share selected aspects of their information with people and organizations that they trust.
3) **Identity Services** - Code for (i) an IMI and SAML compatible Identity Provider and (ii) enabling websites to be IMI and OpenID compatible.

B. Reference Scenario

As we have already introduced, in this Section we consider the IaaS cloud federation scenario. In particular, we suppose to have a wide distributed infrastructure, composed of different clouds, belonging to different administrative domains. Each cloud is able to satisfy service requests (in the case of IaaS we consider VMs as resources) coming from its users. When, for some reason (e.g. a temporary load peak) a given cloud is not able to satisfy users' requests anymore, instead of rejecting them, it could ask the additional needed resources to external providers. These latter might be other clouds able to join the federation.

Obviously, the achievement of this process may be difficult due to several issues that have to be addressed: first of all, when a cloud has expired its resources and ask them to external providers, these have to correctly identify the entity that have generated the request, and accept it only if it has been originated from a trusted source. This leads to the need of managing the authentication process. The simplest solution consists in the possibility of creating a set of credentials for each cloud, on every other cloud aiming to attend the federation. Even though this solution is the straightforward one, its applicability is limited to a scenario just formed by a small number of entities. If we consider the hypotheses of a growing number of clouds, the creation of credentials for each one may be a different task to manage.

In order to simplify the authentication in such scenarios, the most common adoptable solution might be based on the SSO. Instead of creating lots of credentials for authenticating each cloud on the others, it could be employed a more flexible solution relying on trusted third-parties. This approach minimizes the number of expected credentials, since a given cloud just need to have an account on one (or more) of these third-party to be authenticated on all the entities (clouds) that are trusted with them.

Once the authentication task has been solved, in order to allow resources sharing among the cloud federation entities, a way to organize clouds information is also needed. A cloud service provider, in fact, may need different kinds of information regarding VMs: associated resources, instantaneous workload and internal application state. In a federated environment, part of this information might be shared among the entities taking part to the infrastructure. As we have already pointed out previously, if each cloud stores information by means of an RDF XRI graph, the process of communicating requests for new resources, their allocation on external providers and finally their exploiting will be more flexible and simpler. Section VI will provide more details and examples on RDF XRI graphs generated in our testbed.

For addressing either authentication and information sharing among clouds aiming to federate themselves, we propose the employment of an XDI based framework whose implementation, in our case, relies on Higgings. In the following we provide an overview of the operations involved in the creation of a binding among two different clouds for sharing resources: we will assume that the involved entities are based on Higgings for implementing their features. Once the authentication process is performed using the SSO, the involved clouds will the able to share the needed information using the XRI representation transmitting them over a secure channel created among them.

Assuming the internal cloud organization as depicted in Figure 3, we can consider a three layered stack where in the top part lies the cloud manager layer (that manages all the high-level operations such as authentication, resource discovery etc.).

As the Figure shows, Cloud Manager includes the Resource Manager, which is able to manage resources allocation on external providers if needed through the Cross-cloud Federation Manager component. (for further details see [27]). When a cloud (we call it Cloud A) has expired its own resources and needs to gain them from the outside, a Discovery process is started from the Resource Manager. The result of this task will be a list of external clouds able to satisfy the request. From the retrieved set, it will be

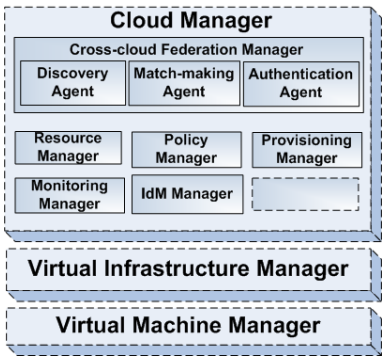

Figure 3. Three layered organization of a Cloud

necessary to select the cloud resource provider best fitting the request: such a task will be accomplished during the Match-Making process. The result of this last operation will consist in the URL of the cloud able to satisfy the resources need of Cloud A (in our case we assume Cloud B as the cloud selected from the Match-Making).

Figure 4 shows the steps involved in the authentication and information sharing between two different clouds (Cloud A and Cloud B): in step 1, the URL coming out from the Match-Making process is used by the Authentication Agent of Cloud A for contacting the destination cloud from which gaining external resources (Cloud B). An HTTP post is forwarded (step 2) with the username and session key (if exists) to the Authentication Agent of Cloud B. The session key is used by the AA of Cloud A as a token proving that a login has been correctly performed on a given IdP and identifies a communication session in a unique way. This means that the key will exist only if the AA already has performed the authentication with an IdP and a session has been created.

During the step 3, the AA of Cloud A is redirected to the IdP trusted with Cloud B for verifying its identity. In step 4, the IdP receives from the AA of Cloud A information about the username and the session key and verifies the existence of a binding between that username and that session key querying a local Database. If an entry exists within the Database, a session for Cloud A has been already created and IdP sends an answer with StatusCode 200, otherwise the login process has to be started for proving the identity of the cloud.

In the last case, in step 5, the AA of Cloud A send its username and password to the IdP that verifies them checking within the LDAP server: if a user exists with that credentials, a new session key is saved within the local Database associated to the username and is sent back to the AA of the Cloud A.

Now that the login phase has been accomplished and the AA of Cloud A holds a valid session key (also registered within the Database), in step 6, it is redirected to the AA of Cloud B where it is now authenticated and able to perform operations: using its username and its session key, Cloud A can now perform the operations needed for creating VMs instances within Cloud B. In the example reported in Figure 4, the *Add* operation is performed for allocating 3 new VMs.

The AA of Cloud B receives the request and control the associated username and session key: such information are then forwarded to the component that is responsible of managing the XRI graph, which performs the addition of the needed nodes and associate them with the session key associated to the username from which the VMs request is coming. From now on, only the entity that holds that key will be able to access those graph node for retrieving information on the new instantiated VMs. An example of possible XRI graphs of cloud A and B is analyzed in the next Section.

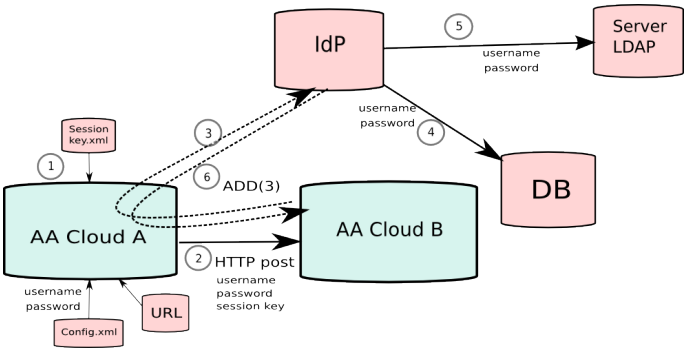

Figure 4. Authentication process and resource information sharing among Cloud A and Cloud B

VI. XRI Graph Representation of VMs

In this Section, we show how can be possible to logically organize and manage data associated to a cloud provider by means of XDI and XRI-addressable RDF graphs. More specifically, considering the scenario, we have already pointed out, we discuss how to represent both VMs hosted within the cloud data-center and VMs lent/borrowed to/from other cloud providers. Moreover, for each XRI graphs, we will show the corresponding XDI documents generated in our testbed. XDI allows to represent RDF XRI graph by means of three main elements:

- **Subject**, e.g., <xdi:s xri="entry"> ... </xdi:s>. It can be a real/abstract entity represented by means of an XRI entry. Examples can be, the cloud iteself, an administrative domain, a cluster, a server, a VM, a cloud-based service instance, an user, etc.
- **Reference**, e.g., <xdi:ref xri="entry"> ... </xdi:ref>. It is a reference to a subject.
- **Predicate**, e.g., <xdi:p xri="relation"> ... </xdi:p>. It can be relation between two or more subjects.

At the beginning, how depicted in Figure 5, cloud A has an administrative domain including cluster1 and cluster2. Cluster 1 includes server1 which hosts VM1 and VM2,

instead cluster 2 includes server2 which hosts VM3. Cloud A has also two users: user1 and user 2. User1 holds VM1 hosted in server1 of cluster1 and VM3 hosted in server 2 of cluster 2. User 2 holds VM2 hosted in server1 of cluster1.

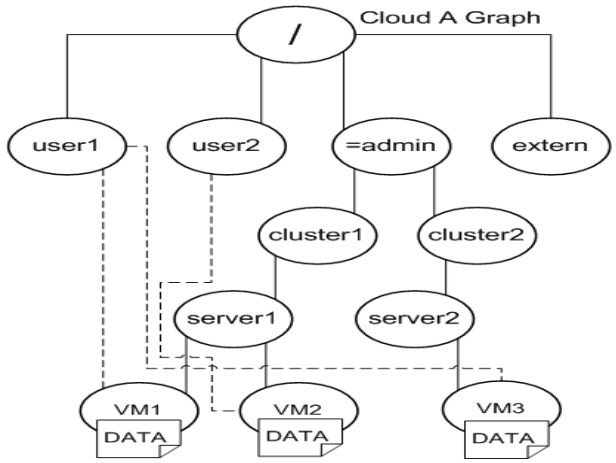

Figure 5. Cloud A graph before federation.

The corresponding XDI code is shown in the following.

```
<?xml version="1.0" encoding="UTF-8"?>
<xdi:xdi xmlns:xdi="http://xdi.oasis-open.org">
  <xdi:s xri="=user1">
    <xdi:p xri="$has$a">
      <xdi:ref xri="+VM1"/>
      <xdi:ref xri="+VM3"/>
    </xdi:p>
  </xdi:s>
  <xdi:s xri="=user2">
    <xdi:p xri="$has$a">
      <xdi:ref xri="+VM2"/>
    </xdi:p>
  </xdi:s>
  <xdi:s xri="+extern"/>
  <xdi:s xri="+admin">
    <xdi:p xri="$has">
      <xdi:ref xri="+cluster1"/>
      <xdi:ref xri="+cluster2"/>
    </xdi:p>
  </xdi:s>
  <xdi:s xri="+admin+cluster1">
    <xdi:p xri="$has">
      <xdi:ref xri="+server1"/>
    </xdi:p>
  </xdi:s>
  <xdi:s xri="+admin+cluster2">
    <xdi:p xri="$has">
      <xdi:ref xri="+server2"/>
    </xdi:p>
  </xdi:s>
  <xdi:s xri="+admin+cluster1+server1">
    <xdi:p xri="$has">
      <xdi:ref xri="+VM1"/>
      <xdi:ref xri="+VM2"/>
    </xdi:p>
  </xdi:s>
  <xdi:s xri="+admin+cluster2+server2">
    <xdi:p xri="$has">
      <xdi:ref xri="+VM3"/>
    </xdi:p>
  </xdi:s>
  <xdi:s xri="+VM2">
    <xdi:p xri="DATA">
      <xdi:data><![CDATA[DATA]]></xdi:data>
    </xdi:p>
  </xdi:s>
  <xdi:s xri="+VM3">
    <xdi:p xri="DATA">
      <xdi:data><![CDATA[DATA]]></xdi:data>
    </xdi:p>
  </xdi:s>
  <xdi:s xri="+VM1">
    <xdi:p xri="DATA">
      <xdi:data><![CDATA[DATA]]></xdi:data>
    </xdi:p>
  </xdi:s>
  <xdi:s xri="+admin+cluster1+server1+VM1"/>
  <xdi:s xri="+admin+cluster1+server1+VM2"/>
  <xdi:s xri="+admin+cluster2+server2+VM3"/>
</xdi:xdi>
```

At the same time (see Figure 6), cloud B includes cluster1 with server1 and server2. Server1 hosts VM1 and VM2, instead server2 hosts VM3. For simplicity, let us suppose that all VMs are reserved.

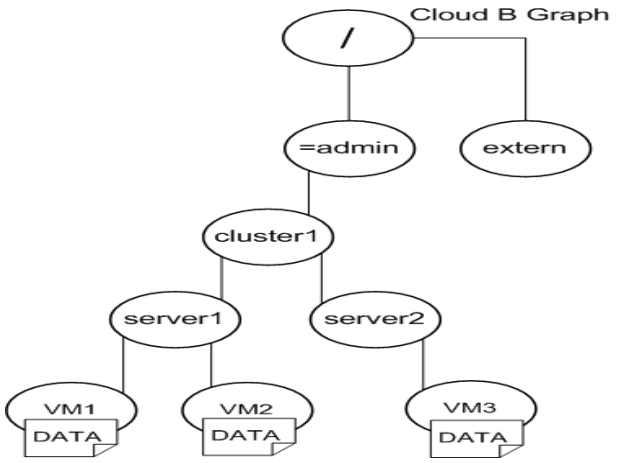

Figure 6. Cloud B graph before federation.

The corresponding XDI code is shown in the following.

```
<?xml version="1.0" encoding="UTF-8"?>
<xdi:xdi xmlns:xdi="http://xdi.oasis-open.org">
  <xdi:s xri="+extern"/>
  <xdi:s xri="=admin">
    <xdi:p xri="$has">
      <xdi:ref xri="+cluster1"/>
    </xdi:p>
  </xdi:s>
  <xdi:s xri="=admin+cluster1">
    <xdi:p xri="$has">
      <xdi:ref xri="+server1"/>
      <xdi:ref xri="+server2"/>
    </xdi:p>
  </xdi:s>
  <xdi:s xri="=admin+cluster1+server1">
    <xdi:p xri="$has">
      <xdi:ref xri="+VM1"/>
      <xdi:ref xri="+VM2"/>
    </xdi:p>
  </xdi:s>
  <xdi:s xri="=admin+cluster1+server2">
    <xdi:p xri="$has">
      <xdi:ref xri="+VM3"/>
    </xdi:p>
  </xdi:s>
  <xdi:s xri="+VM2">
    <xdi:p xri="DATA">
      <xdi:data><![CDATA[DATA]]></xdi:data>
    </xdi:p>
  </xdi:s>
  <xdi:s xri="+VM3">
```

```
    <xdi:p xri="DATA">
      <xdi:data><![CDATA[DATA]]></xdi:data>
    </xdi:p>
  </xdi:s>
  <xdi:s xri="+VM1">
    <xdi:p xri="DATA">
      <xdi:data><![CDATA[DATA]]></xdi:data>
    </xdi:p>
  </xdi:s>
  <xdi:s xri="=admin+cluster1+server1+VM1"/>
  <xdi:s xri="=admin+cluster1+server1+VM2"/>
  <xdi:s xri="=admin+cluster1+server2+VM3"/>
</xdi:xdi>
```

Then, let us suppose that user2 of cloudA requires three additional VMs. As cloud A realizes that it does not have enough resources for instantiate further VMs, it establishes a federation with cloud B, as already described in the previous Section. After authentication, cloudA sends a request for the instantiation of three VMs. So that, cloud B instantiates three VMs in its own datacenter, i.e., VM4 in server1 and VM5 and VM6 in server2. The corresponding updated graph is depicted in Figure 7. Moreover, an user entry for cloud A is created linking the three new instantiated VMs with a reference to them.

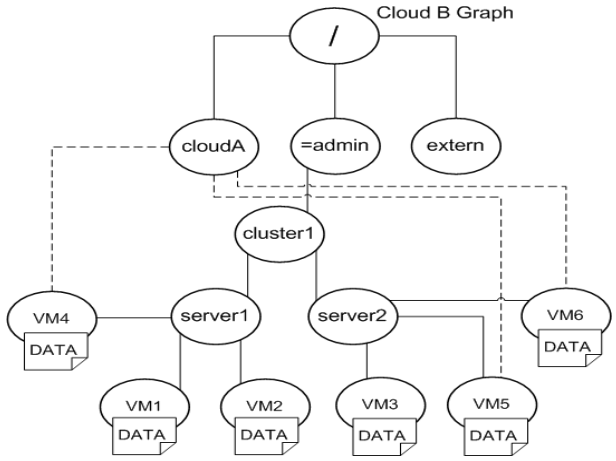

Figure 7. Cloud B graph after federation.

The corresponding XDI code is shown in the following.

```
<?xml version="1.0" encoding="UTF-8"?>
<xdi:xdi xmlns:xdi="http://xdi.oasis-open.org">
  <xdi:s xri="+extern"/>
  <xdi:s xri="=admin">
    <xdi:p xri="$has">
      <xdi:ref xri="+cluster1"/>
    </xdi:p>
  </xdi:s>
  <xdi:s xri="=admin+cluster1">
    <xdi:p xri="$has">
      <xdi:ref xri="+server1"/>
      <xdi:ref xri="+server2"/>
    </xdi:p>
  </xdi:s>
  <xdi:s xri="=admin+cluster1+server1">
    <xdi:p xri="$has">
      <xdi:ref xri="+VM1"/>
      <xdi:ref xri="+VM2"/>
      <xdi:ref xri="+VM4"/>
    </xdi:p>
  </xdi:s>
  <xdi:s xri="=admin+cluster1+server2">
    <xdi:p xri="$has">
      <xdi:ref xri="+VM3"/>
      <xdi:ref xri="+VM6"/>
      <xdi:ref xri="+VM5"/>
    </xdi:p>
  </xdi:s>
  <xdi:s xri="+VM2">
    <xdi:p xri="DATA">
      <xdi:data><![CDATA[DATA]]></xdi:data>
    </xdi:p>
  </xdi:s>
  <xdi:s xri="+VM3">
    <xdi:p xri="DATA">
      <xdi:data><![CDATA[DATA]]></xdi:data>
    </xdi:p>
  </xdi:s>
  <xdi:s xri="+VM1">
    <xdi:p xri="DATA">
      <xdi:data><![CDATA[DATA]]></xdi:data>
    </xdi:p>
  </xdi:s>
  <xdi:s xri="=admin+cluster1+server1+VM1"/>
  <xdi:s xri="=admin+cluster1+server1+VM2"/>
  <xdi:s xri="=admin+cluster1+server2+VM3"/>
  <xdi:s xri="+VM6">
    <xdi:p xri="DATA">
      <xdi:data><![CDATA[DATA]]></xdi:data>
    </xdi:p>
  </xdi:s>
<xdi:s xri="+cloudA">
    <xdi:p xri="$has$a">
      <xdi:ref xri="+VM6"/>
      <xdi:ref xri="+VM5"/>
      <xdi:ref xri="+VM4"/>
    </xdi:p>
  </xdi:s>
  <xdi:s xri="+VM5">
    <xdi:p xri="DATA">
      <xdi:data><![CDATA[DATA]]></xdi:data>
    </xdi:p>
  </xdi:s>
  <xdi:s xri="+VM4">
    <xdi:p xri="DATA">
      <xdi:data><![CDATA[DATA]]></xdi:data>
    </xdi:p>
  </xdi:s>
</xdi:xdi>
```

Finally, cloud B sends the data related to the three instantiated VMs (e.g., how to access the VM, IP address, CPU, RAM, storage, operating system, etc) to cloud A which update its XRI graph. For simplicity in the XDI document, we indicated such information with the string "DATA". As depicted in Figure 8, cloud A, under the "extern" entry, creates an entry for cloud B, linking nodes representing the three extern VMs. More specifically, +Ext-VM1, +Ext-VM2, +Ext-VM3 are aliases of the three VMs instantiated in cloud B.

The corresponding XDI code is shown in the following.

```
<?xml version="1.0" encoding="UTF-8"?>
<xdi:xdi xmlns:xdi="http://xdi.oasis-open.org">
  <xdi:s xri="=user1">
    <xdi:p xri="$has$a">
      <xdi:ref xri="+VM1"/>
      <xdi:ref xri="+VM3"/>
    </xdi:p>
  </xdi:s>
  <xdi:s xri="=user2">
    <xdi:p xri="$has$a">
      <xdi:ref xri="+VM2"/>
      <xdi:ref xri="+EXT+VM1"/>
      <xdi:ref xri="+EXT+VM3"/>
      <xdi:ref xri="+EXT+VM2"/>
```

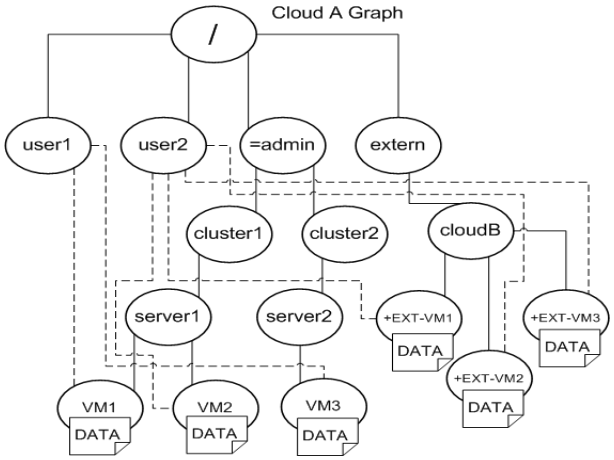

Figure 8. Cloud A graph after federation.

```
    </xdi:p>
  </xdi:s>
  <xdi:s xri="+extern">
    <xdi:p xri="$has">
      <xdi:ref xri="+cloudB"/>
    </xdi:p>
  </xdi:s>
  <xdi:s xri="+admin">
    <xdi:p xri="$has">
      <xdi:ref xri="+cluster1"/>
      <xdi:ref xri="+cluster2"/>
    </xdi:p>
  </xdi:s>
  <xdi:s xri="+admin+cluster1">
    <xdi:p xri="$has">
      <xdi:ref xri="+server1"/>
    </xdi:p>
  </xdi:s>
  <xdi:s xri="+admin+cluster2">
    <xdi:p xri="$has">
      <xdi:ref xri="+server2"/>
    </xdi:p>
  </xdi:s>
  <xdi:s xri="+admin+cluster1+server1">
    <xdi:p xri="$has">
      <xdi:ref xri="+VM1"/>
      <xdi:ref xri="+VM2"/>
    </xdi:p>
  </xdi:s>
  <xdi:s xri="+admin+cluster2+server2">
    <xdi:p xri="$has">
      <xdi:ref xri="+VM3"/>
    </xdi:p>
  </xdi:s>
  <xdi:s xri="+VM2">
    <xdi:p xri="DATA">
      <xdi:data><![CDATA[DATA]]></xdi:data>
    </xdi:p>
  </xdi:s>
  <xdi:s xri="+VM3">
    <xdi:p xri="DATA">
      <xdi:data><![CDATA[DATA]]></xdi:data>
    </xdi:p>
  </xdi:s>
  <xdi:s xri="+VM1">
    <xdi:p xri="DATA">
      <xdi:data><![CDATA[DATA]]></xdi:data>
    </xdi:p>
  </xdi:s>
  <xdi:s xri="+admin+cluster1+server1+VM1"/>
  <xdi:s xri="+admin+cluster1+server1+VM2"/>
  <xdi:s xri="+admin+cluster2+server2+VM3"/>
  <xdi:s xri="+extern+cloudB">
    <xdi:p xri="$has">
      <xdi:ref xri="+EXT+VM1"/>
      <xdi:ref xri="+EXT+VM2"/>
      <xdi:ref xri="+EXT+VM3"/>
    </xdi:p>
  </xdi:s>
  <xdi:s xri="+EXT+VM2">
    <xdi:p xri="DATA">
      <xdi:data><![CDATA[DATA]]></xdi:data>
    </xdi:p>
  </xdi:s>
  <xdi:s xri="+EXT+VM3">
    <xdi:p xri="DATA">
      <xdi:data><![CDATA[DATA]]></xdi:data>
    </xdi:p>
  </xdi:s>
  <xdi:s xri="+EXT+VM1">
    <xdi:p xri="DATA">
      <xdi:data><![CDATA[DATA]]></xdi:data>
    </xdi:p>
  </xdi:s>
  <xdi:s xri="+extern+cloudB+EXT+VM1"/>
  <xdi:s xri="+extern+cloudB+EXT+VM2"/>
  <xdi:s xri="+extern+cloudB+EXT+VM3"/>
</xdi:xdi>
```

In the end, references to entries +Ext-VM1, +Ext-VM2, +Ext-VM3 are linked to user2 who is able to access the three VMs. The interesting thing is that both cloud A and user2 are not aware of the details of where the three VMs are hosted: they can only know the information related to the three VMs. Further information related to cloud B are hidden.

VII. Conclusions and Remarks

In this paper, we focused on how to apply XDI to a federated cloud scenario to represent and exchange data related to cloud-based services. As pointed out, XDI can be used to design several high-level management mechanisms in a cloud system, supporting both data models and security. More specifically, we discuss how can be possible to design XDI-based mechanisms in a cloud system using the Higgings framework focusing on a scenario of federated IaaS clouds lending/borrowing VMs each other. In the end, an use case has been described and implemented, showing the XRI graphs representing VMs and the corresponding XDI documents before and after a federation between two clouds. XDI is a generalized, extensible service for sharing, linking, and synchronizing structured data over the Internet originally thought for web-based systems. In this paper, we hope to success stimulating your interest toward the designing and development of XDI-based mechanisms for cloud computing system. In future works we plan to evaluate the performance of the system also considering the overhead due to the security.

References

[1] M. V. A. P. Antonio Celesti, Francesco Tusa, "Evaluating an open source extensible resource identifier naming system for cloud computing environments," in *The Third International IARIA Conference on Evolving Internet (INTERNET 2011)*, pp. 26–31, June 2011.

[2] OpenXRI Project, XRI applications and libraries, http://www.openxri.org/.

[3] D. Reed, G. Strongin, XDI (XRI Data Interchange), A White Paper for the OASIS XDI Technical Committee v2, OASIS, 2004.

[4] "Higgings personal data service, http://www.eclipse.org/higgins/."

[5] A. Celesti, M. Villari, and A. Puliafito, "Ecosystem of cloud naming systems: An approach for the management and integration of independent cloud name spaces," (Los Alamitos, CA, USA), pp. 68–75, IEEE Computer Society, 2010.

[6] G.-J. Ahn, M. Ko, and M. Shehab, "Privacy-enhanced user-centric identity management," in *IEEE International Conference on Communications, ICC '09*, pp. 14–18, June 2009.

[7] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud computing and grid computing 360-degree compared," in *Grid Computing Environments Workshop, 2008. GCE '08*, pp. 1–10, 2008.

[8] R. L. Grossman, "The case for cloud computing," in *IT Professional*, vol. 11, pp. 23–27, March 2009.

[9] E. Brynjolfsson, P. Hofmann, and J. Jordan, "Cloud computing and electricity: beyond the utility model," *Commun. ACM*, vol. 53, pp. 32–34, May 2010.

[10] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *Commun. ACM*, vol. 53, pp. 50–58, April 2010.

[11] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the clouds: A berkeley view of cloud computing," Tech. Rep. UCB/EECS-2009-28, EECS Department, University of California, Berkeley, Feb 2009.

[12] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Generation Computer Systems*, vol. 25, no. 6, pp. 599 – 616, 2009.

[13] D. Yang, Y. Qin, H. Zhang, H. Zhou, and B. Wang, "Urns: A new name service for uniform network resource location," in *Wireless, Mobile and Multimedia Networks, 2006 IET International Conference*, pp. 1–4, 2006.

[14] Y. Doi, S. Wakayama, M. Ishiyama, S. Ozaki, T. Ishihara, and Y. Uo, "Ecosystem of naming systems: discussions on a framework to induce smart space naming systems development," in *ARES*, p. 7, April 2006.

[15] Y. Doi, "Dns meets dht: treating massive id resolution using dns over dht," in *Applications and the Internet International Symposium*, pp. 9–15, January 2005.

[16] S. Chaisiri and P. Uthayopas, "Survey of resource discovery in grid environments," tech. rep., High Performance Computing and Networking Center, Department of Computer Engineering, Faculty of Engineering, Kasetsart University, 50 Phaholyothin Rd., Chatuchak, Bangkok 10900, Thailand, April 2008.

[17] A. Sharma and S. Bawa, "Comparative analysis of resource discovery approaches in grid computing.," *JCP*, vol. 3, no. 5, pp. 60–64, 2008.

[18] A. Hameurlain, D. Cokuslu, and K. Erciyes, "Resource discovery in grid systems: a survey," *Int. J. Metadata Semant. Ontologies*, vol. 5, pp. 251–263, July 2010.

[19] H. Sun, J. Huai, Y. Liu, and R. Buyya, "RCT: A distributed tree for supporting efficient range and multi-attribute queries in grid computing," *Future Gener. Comput. Syst.*, vol. 24, no. 7, pp. 631–643, 2008.

[20] Y. Mei, X. Dong, W. Wu, S. Guan, and J. Li, "Sdrd: A novel approach to resource discovery in grid environments," in *Advanced Parallel Processing Technologies* (M. Xu, Y. Zhan, J. Cao, and Y. Liu, eds.), vol. 4847 of *Lecture Notes in Computer Science*, pp. 301–312, Springer Berlin / Heidelberg, 2007. 10.1007/978-3-540-76837-1_34.

[21] Wikipedia OpenID, http://en.wikipedia.org/wiki/OpenID, July 2011.

[22] OpenID World Wide Usage, http://www.ariadne.ac.uk/issue51/powell-recordon/, June 2007.

[23] The Security Vulnerability of Reassignable Identifiers, http://dev.inames.net/wiki/XRI_and_OpenID, July 2011.

[24] Extensible Resource Identifier (XRI) Syntax V2.0, Committee Specification, OASIS, 2005.

[25] Extensible Resource Identifier (XRI) Resolution V2.0, Committee Draft 03, OASIS, 2008.

[26] A. Celesti, M. Villari, and A. Puliafito, "Ecosystem of cloud naming systems: An approach for the management and integration of independent cloud name spaces," *IEEE International Symposium on Network Computing and Applications (IEEE NCA10)*, vol. 0, pp. 68–75, 2010.

[27] A. Celesti, F. Tusa, M. Villari, and A. Puliafito, "How to enhance cloud architectures to enable cross-federation," in *2010 IEEE 3rd International Conference on Cloud Computing*, pp. 337–345, IEEE, July 2010.

[28] A. Celesti, M. Villari, and A. Puliafito, "A Naming System Applied to a Reservoir Cloud," in *2010 Sixth International Conference on Information Assurance and Security (IAS)*, pp. 247–252, IEEE, August 2010.

[29] Resources and Services Virtualization without Barriers (Reservoir) European Project, http://www.reservoir-fp7.eu/.

[30] "Xri data interchange, oasis, http://wiki.oasis-open.org/xdi/xdigraphmodel."

[31] Organization for the Advancement of Structured Information Standards (OASIS), http://www.oasis-open.org.