Evaluating an Open Source eXtensible Resource Identifier Naming System for Cloud Computing Environments

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*—Clouds are continuously changing environments where services can be composed with other ones in order to provide many types of other services to their users. In order to enable cloud platforms to manage and control their assets, they need to name, identify, and retrieve data about their virtual resources in different operating contexts. These tasks can not be easily accomplished using only the DNS and this leads cloud service providers to design proprietary solutions for the management of their name spaces. In this paper, we discuss a possible cloud naming system based on the eXtensible Resource Identifier (XRI) technology. More specifically, we evaluate the performance of OpenXRI, one of its open source implementations, simulating typical cloud name space management tasks.**

***Keywords*-Cloud Computing, Naming System, XRI, OpenXRI.**

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 VEs 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 or is damaged, the cloud can decide to move or "migrate" one or more VEs 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, etcetera. In addition, if we consider the perspective of cloud federation where clouds cooperate sharing computational and storage resources, a VE might 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 a VEs from a cloud to another.

Such a dynamic, variegated, and continuously changing scenario involves no just cloud services and VEs, 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 VE needs to be identified by a name, it may happen that different entities (e.g., cloud software middlewares, cloud administrators, cloud user, etcetera) may be interested to resolve that name retrieving either data concerning general information on the VE (e.g., CPU, memory, kernel, operating system, virtualization format version), data regarding the performance of the VE (e.g., used CPU and memory), or by means of a Single-Sign-On (SSO) authentication service (e.g., using an Identity Provider (IdP) asserting the trustiness of the VE when it migrate from a place to another in order to avoid identity theft), and many others. 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 our opinion, for the aforementioned concerns the management and integration of cloud name spaces can be difficult because such a scenario raises several issues concerning naming and service location for all the involved entities and the traditional DNS-based systems along with URL, URI, and IRI standards are inadequate for cloud computing scenarios.

In order to discourage a possible evolving scenario where each cloud could develop its own proprietary cloud naming systems with compatibility problems in the interaction among different cloud name spaces, this paper aims to propose a standard approach for the designing of a seamless cloud naming system able to manage and integrate independent cloud name spaces also in a federated scenario.

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 XRI technology motivating how it suits the management of cloud name spaces. In Section V we discuss OpenXRI [1], one of the XRI open source implementations, showing how to use it as cloud naming system. An analysis of the performances of OpenXRI managing a simulated cloud name space is discussed in Section VI. Conclusions and lights to the future 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 our previous work [2]. 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.

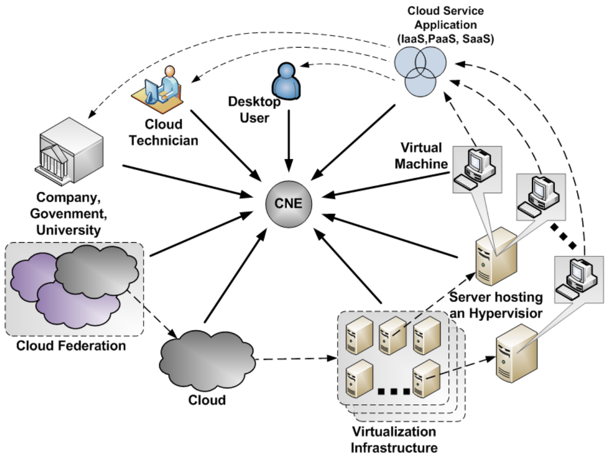
Cloud Service Application (IaaS,PaaS, SaaS)
Cloud Technician
Desktop User
Virtual Machine
Company, Govenment, University
CNE
Server hosting an Hypervisior
Cloud Federation
Cloud
Virtualization Infrastructure

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 [3] 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

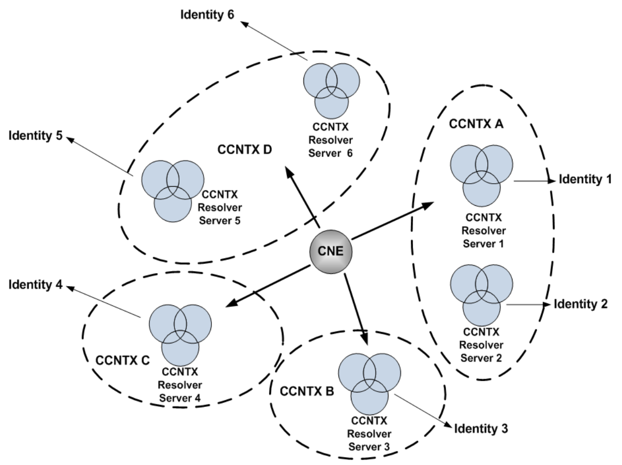

Figure 2. Examples of a generic CNE associated to several CCNTXs.

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 [4], [5]. As previously discussed, 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, as 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 the DNS is presented in [6]. 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 [7] 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 [8]. The authors adopt a set of gateways executing a dynamic DNS name delegation between DNS resolver and DHT node.

Regarding naming, name resolution, and service location in federated cloud environments, in our previous work [2], besides highlighting the cloud name space issues previously discussed, we proposed a generic theoretical cloud naming framework for the management of cloud name spaces. As possible representation of the cloud naming framework we chose XRI [9] and the eXtensible Resource Descriptor Sequence (XRDS) [10] technologies which are also the focus of this work. How will be described in the following Sections the aim of this paper is to evaluate the performances of several operational tasks using the OpenXRI implementation by means of the simulation of typical cloud name space management tasks.

IV. An XRI Naming System for Cloud Computing

In this Section, after a brief description the XRI technology, we motivate how it can help the cloud name space management.

The XRI protocol provides 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 tree. According to the XRI terminology, each entity in the tree 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.

An authority is resolved by means of an XRDS document representing a simple, extensible XML resource description format standard describing the features of any URI, IRI, or XRI-identified entity in a manner that can be consumed by any XML-aware system. Each XRDS describes which types of information are associated to an authority an the way in which they can be obtained. Using HTTP, XRI resolution involves two phases: authority resolution which is the phase required to resolve a XRI into a XRDS document from an XRI Authority Resolution Server (ARS), and Service End-Point Selection which is the phase of selection of the SEP server (e.g., web service, service provider, web application) returning the data describing the entity in a given context. The same SEP server can also return different data of the same authority.

In our opinion, as XRI meets the requirements of cloud name space management, it can be adopted to develop a seamless mechanism for retrieve data regarding CNEs. As XRI is compatible with IRI naming systems, there is not the need to use a unique global naming system, even though this would be possible. This feature allows clouds to manage their own XRI naming systems, mapping them on the global DNS maintaining the compatibility with the existing naming systems. Moreover, with XRI a cloud can keep different trees 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 the resolution of XRI authorities. In addition, the XRDS document can be used to describe a XRI authority associated to a target service of VM, indicating how to resolve it by means of the corresponding SEP.

For example the cloud service provider may need to retrieve three types of information about an authority representing a VM, resolving it in three different ways. In the fist way the VM has to be resolved by means of general data (e.g., CPU, memory, kernel, operating system), in the second way the VM has to be resolved by means of real time performance data (e.g., amount of used CPU and memory used), in the third way instead the VM has to be resolved by means real time data regarding an internal running application (e.g., the percentage of processed data). Such a situation can be addressed by mean of three different XRDs inside the XRDS document corresponding to the VM, authority, each one pointing to a target SEP server.

V. How to Manage Cloud Name Spaces Using an OpenXRI Architecture

Regarding the implementation of the XRI technology, currently there are not many available solutions on the market. Nowadays, in our opinion, the OpenXri Project is one of the best open source initiatives which aims to promote the development of XRI-based applications. Therefore, in this Section we describe how to implement a CNS using the java libraries developed by the Openxri Project. Our practice of CNS includes the following components: the XRI Authority Resolution Server (ARS) 1.2.1, the XRI Client Resolver (CR), the XRI Cloud Name Space Management (CNSM) front-end and the SEP Server. The OpenXRI has provided the java libraries to arrange the following components:

- **The ARS 1.2.1**, a server for the resolution XRI authorities (i.e., in our scenario CNEs). It is provided along with a web application to allow administrators to create, move, and delete authorities and thus managing XRI trees.
- **The XRI CR**, a software client which resolves an authority queering a ARS, retrieving the corresponding XRDS document, and performing a "SEP Selection Task" choosing the right SEP Server acting in a given CCNTX for the resolution of an XRI authority. The XRI CR is used by each entity interested to resolve a CNE name, e.g., another cloud, a desktop client, a service provider, an IT society, and so on.

The XRI ARS developed by the Openxri Project provides an administration web interface where an user can interactively manage his name space. Such a condition is very penalizing for our scenario, as we assumed that the cloud name space should be also managed automatically by the cloud middleware according to the business model in force on the cloud. In fact, in our opinion there are circumstances where the interaction with an administrator is required and other cases where the cloud has to arrange its assets automatically by itself. For such reasons we have designed the **XRI CNSM front-end** offering both a standard SOAP web service interface in order to make the naming system controllable by any cloud platform, and additional specific utilities for the name space management of cloud computing environments. The choice of using SOAP is motivated by the fact that it provides a consolidated framework offering security an Quality of Service (QoS) support. At any rate, nothing prevents the possibility to use another web service technology. In addition, **SEP Servers** have been developed by means of Restfull web services. The aim of the SEP Server is to resolve CNE name in a given CCNTX sending data in XML format to the XRI CR. In this case, the choice of the Restfull technology has been motivated by the fact that it offers better performances than SOAP in term of response time (this is very useful especially when it is needed to retrieve real-time data, e.g., the performances of a VM). At any rate, also in this case, nothing prevents the possibility to use another web service technology.

Security, is a hot topic in cloud computing. For this reason, even though security is not the focus of this paper is worthwhile to spend a few words about the security of the proposed CNS. Regarding the CNE name resolution, the XRI technology natively supports secure resolution using the Security Assertion Markup Language (SAML) [11]. As far as it is concerned the interaction between the cloud middleware and the XRI CNSM front-end, it can be easily secured using the WS-Secutity features [12] of the SOAP protocol. In the end, considering the information retrieval of an XRI software client from the Restful SEP Server, security can be accomplished using https.

In Figure 3 is depicted an example of CNE name resolution. In step 1, the XRI CR software wants resolver a CNE name and contacts the XRI ARS making a resolution request. In step 2, the XRI ARS resolves the name and responds to the XRI CR sending the XRDS document corresponding to the the CNE name. In step 3 the XRI CR performs a SEP selection task choosing the server for the resolution of the CNE name in a given CCNTX to which performing a Restfull web service request. In step 4 SEP server responds with the data resolving the CNE name in the corresponding CCNTX, so that the XRI CR can process the obtained XML data. In the rest of the paper, as there are not limits to the type of CNE which our architecture is able to manage, we focus on the name space management and information retrieval of VMs.

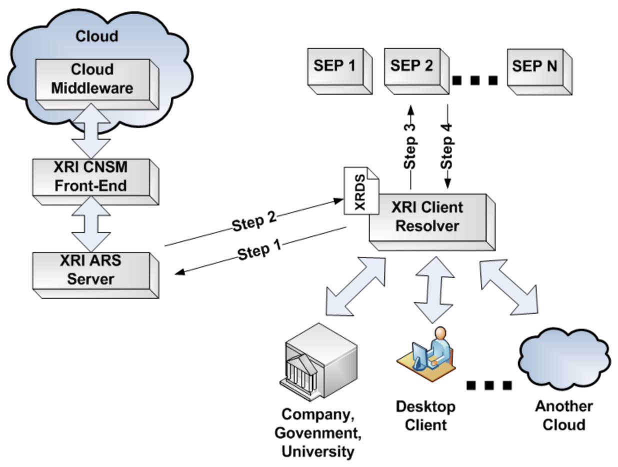

Figure 3. Example of CNE name resolution in a CNS.

As clouds are highly dynamic environments the logical and physical arrangement of its resources can continuously change. For example a VM can be either physically moved from a server running a hypervisor to another. Figure 4 depicts such a situation presenting an example of XRI name space changing due to a CNE name movement. In the XRI name space tree on the left of the Figure at t time the "VE2" CNE name is logically mounted under the "service2" CNE name, instead, on the right of the Figure, at $t + 1$ time is logically mounted under "service1".

VI. Authority Moving Performances

In this Section, we present several experiments on a real testbed in order to evaluate our XRI CNS implementation.

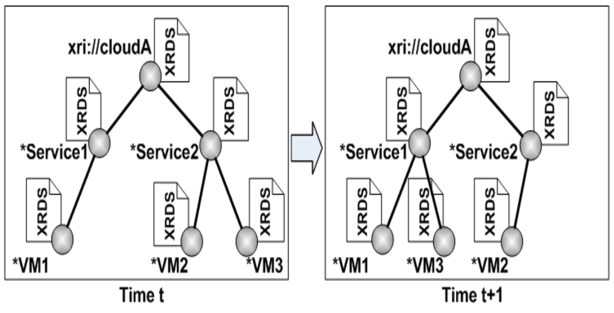

Figure 4. Example of CNE name movement.

More specifically, we focused on the evaluation of the costs due to the movement of XRI authorities (i.e. CNE names) within the XRI tree representing the cloud name space. In order to evaluate the goodness of the OpenXRI architecture for the management of cloud name spaces, in our opinion it is necessary to understand the overall behavior of the architecture under particular conditions of workload. As XRI follows a hierarchical approach for the management of name spaces using one or more tree structures, we decided to stress the operations of XRI authority movement in such structures considering two possible cases: "Wide Tree" and "Deep Tree". Considering the "Wide Tree" case, we performed movement tasks with 10, 100, and 1000 XRI authority. Instead, as far as it is concerned the "Deep Tree" case, we performed tests with 10, 20, and 30 levels in the XRI tree structure. Experiments have consisted in the movement of the last authority on the right under a the first authority on the left of the XRI tree.

Our experiments have been performed considering a testbed deployed inside a computer running a Redhat Enterprise Linux AS Release 3.0.8 operating system having the following hardware features: Blade LS21 AMD Opteron Biprocessor Dual Core 2218 2.6GHz 8GB RAM. Instead, in order to emulate a cloud middleware interacting with the SOAP web service interfaces of the CNSM front-end, we used JMeter, an open source automatic client tool, which has been deployed within another computer. More specifically, we store a typical cloud behavior pattern and then we applied it repetitively.

To estimate the workload of OpenXRI ARS 1.2.1 we evaluated the Response Time of the system expressed in *msecs*. We have measured the time interval between the request phase to the XRI CNSM front-end at Ts and the response time at Tr, taking place in the receiving phase. In our graphs we reported the total time spent to accomplish each task: $Tt = Ts + Tr$. The exchange of requests and responses is measured in a local network (LAN, without any Internet connection), since the measurements are not affected from the network communication parameters (e.g., throughput, delays, jitter, etcetera). The series of tests executed (50 runs for each simulation) guarantee a wide coverage of possible results. The confidence interval (at 95%) indicates the goodness of our analysis. Figure 5 summarizes the response time trend regarding the authority movement tasks using the "Wide Tree" and considering 10, 100, and 1000 authorities. On the x-axis we have represented the number of considered nodes, whereas on the y-axis we have represented the response time expressed in milliseconds. Observing the

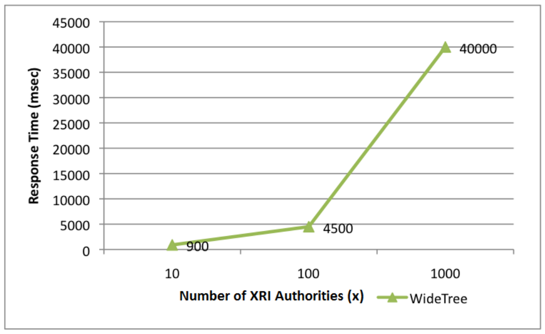

Figure 5. Authority movement tasks in the "Wide Tree" case with 10, 100, and 1000 authorities.

graph, we notice that with 1000 nodes we have a response time of 40 seconds, a rather high value, but reasonable considering the presence of 1000 operating VEs. Instead, in the case of 10 and 100 nodes, the "Wide Tree" needs a time that ranges from about 1 to 5 seconds in order to perform authority movement tasks. This latter results are reasonable in cloud environments, in particular, if we assume a scenario where a single VE needs to be boot up, from an unrunning state. Usually, the time needed for the VE boot-up is higher than the time spent by OpenXRI ARS for any type of tree reconfiguration.

Instead, the graph depicted in Figure 6 shows the response time trend concerning authority movement tasks considering a "Deep Tree" with 10, 20, and 30 tree levels. On the x-axis we have represented the number of levels, whereas on the y-axis we have represented the response time expressed in milliseconds. Observing the graph the worst case (an authority movement within a tree with 30 levels) implies

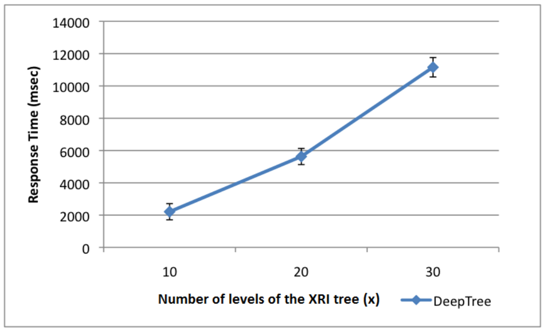

Figure 6. Authority movement tasks in the "Deep Tree" case with 10, 20, and 30 levels.

12 seconds. In reality, the case under analysis may be considered as an event with a low probability in cloud computing environments. In fact, it represents an hierarchical structure with 30 levels in which we should identify 30 CNEs with hierarchical relationships. However, with a few levels, and with our hardware configuration the response time we can achieved is rather low.

VII. Conclusions and Future Works

In this paper we presented a solution for identifying and resolving VMs and more in general various other CNEs in different operating cloud contexts. Particularly, we presented the XRI technology as a possible solution to these problems, evaluating one of its open implementation, that is OpenXRI. This works has highlighted how several tasks can be accomplished using OpenXRI for the management of cloud name spaces. The conducted experiments show the goodness of OpenXRI and how it is particularly suitable to our goals. In future works we are planning to apply some improvements to the OpenXRI Authotity Resolution Server 1.2.1 for the accomplishment of the tasks needed for the management of cloud name spaces.

Acknowledgements

The research leading to the results presented in this paper has received funding from the European Union's Seventh Framework Programme (FP7 2007-2013) Project RESERVOIR under grant agreement number 215605 and from the Union's Seventh Framework Programme (FP7 2007-2013) Project VISION-Cloud under grant agreement number 217019.

References

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

[2] 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.

[3] 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.

[4] 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.

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

[6] 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.

[7] 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.

[8] 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.

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

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

[11] "Security assertion markup language, oasis, http://www.oasis-open.org/committees/security."

[12] "Web services security: Soap message security 1.0, oasis, http://www.oasis-open.org/committees/wss."