

AAA in a Cloud-Based Virtual DIME Network Architecture (DNA)

Francesco Tusa, Antonio Celesti

Dept. of Mathematics, Faculty of Engineering, University of Messina
Contrada di Dio, S. Agata, 98166 Messina, Italy.
e-mail: {ftusa, acelesti}@unime.it

Dr. Rao Mikkilineni

Kawa Objects Inc.
Los Altos USA
e-mail: rao@kawaobjects.com

Abstract—The design of a cloud architecture can be difficult according to the business logic complexity which Cloud Service Providers (SPs) want to achieve. This paper describes a possible solution for simplifying the design of cloud-based services exploiting a DIME Network Architecture (DNA). More specifically, exploiting the signaling channel and FCAPS of DIMEs, we discuss an approach for the management of SSO Authentication, Authorization, and Accounting in cloud-based services.

Keywords—Cloud Computing, Distributed Computing, DIME Network Architecture, Signaling Channel, SSO Authentication, Authorization, Accounting

I. INTRODUCTION

Evolution has taught us that security, self-protection, and real-time monitoring and control of the environment are essential components of an organism's survival. Cellular organisms have evolved complex mechanisms to ensure their survival and have deployed them using a genetic computing model that supports replication, repair, recombination and reconfiguration [1].

The new computing model, known as the Distributed Intelligent Managed Element (DIME) Network Computing Model introduced in the last WETICE conference in Greece [2] borrows various abstractions from the genetic computing model and incorporates security as one of the components to manage service workflows deployed using distributed von Neumann computing elements. The DIME computing model consists of a signaling network overlay over the computing service network and allows parallelism between the control (setup, monitoring, analysis and reconfiguration based on workload variations, business priorities and latency constraints) and the computing functions of the distributed software components.

As depicted in Figure 1, a local DIME of a DIME network is a computing entity which allows fault, configuration, accounting, performance and security (FCAPS) tasks through five different threads: "F", "C", "A", "P", "S" all connected on a signaling channel. A local DIME manager manages all DIME components. The DIME's computing component is called Managed Intelligent Computing Element (MICE). The MICE receives the instructions for the service or application to be executed through the "C" thread using the signaling path, whereas its inputs and outputs are managed

through the data path. Specialization of the service component that is provided by the DIME is

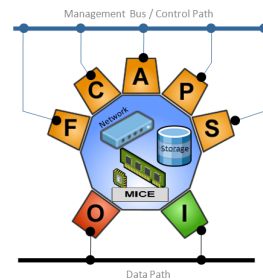


Figure 1. Local DIME entity.

accomplished by programming individual MICE to perform specific tasks using the input and deliver the output to appropriate DIME in the network. These tasks, depending on user requirements are programmed and executed as loadable modules in each DIME. The distributed software components along with associated profiles defining their use and management constraints are executed by DIMEs endowed with self-management and signaling-enabled-control architecture. A workflow is implemented by connecting the input and output of different DIME's MICEs on the data path and it is organized as a Directed Acyclic Graph (DAG). Considering cloud computing environments, a DIME network architecture (DNA) can be implemented using virtual servers using the Hypervisor.

Nowadays, Cloud Computing represents a tempting business opportunity for ICT operators thanks to the success of the virtualization technologies. Nevertheless, the designing of complex cloud environments implementing the business logic of Service Providers (SPs) is not so trivial. In fact, SPs have to control complex systems providing Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS), while monitoring their performance and security. Furthermore, SPs have to dynamically audit, configure, rearrange and scale their cloud-based services according to given Service Level Agreements (SLAs). In this paper, we discuss how to simplify the designing of cloud-based services using the DIME Network Computing Model, focusing our attention on security. More specifically, we will

discuss how to manage Authentication, Authorization, and Accounting in a DIME network deployed on a pool of virtual resources.

The paper is organized as follows. Section II describes how the DIME Network Computing Model facilitates the designing of cloud-based services. A discussion about DIME network security issues in a cloud computing environment is made in Section III. In Section IV, we will describe how to address authentication, authorization, and accounting for the arrangement of workflows implementing cloud-based services specifically focusing on Singel Sign On (SSO) authentication. Conclusions are summarized in Section V.

II. THE DIME NETWORK COMPUTING MODEL FOR CLOUD COMPUTING

Nowadays, cloud computing is an affordable business opportunity for ICT operators. Currently, many cloud solutions are increasingly being deployed on the ICT market but at the same time the lack of standards, are making the ICT operators uncertain about how to build their business. Often, the implementation of a cloud architecture is not so simple due to the complexity of the business logic which cloud SPs want to implement. A study held by Lorente et al. has identified a generic three-layer which classifies cloud architectures. Starting from the bottom these layers are respectively Virtual Machine Monitor (VMM), Virtual Infrastructure Manager (VIM), and Cloud Manager (CM).

At VMM layer it is possible to classify all those solutions (i.e., hypervisors) which allow to build VMs running a guest OS on a physical server (e.g., Xen, Kvm, VMware, Virtual Box, etcetera).

The software solution which orchestrate a physical pool of server running each one an hypervisor can be classified at VIM layer. All these solutions commonly allow to manage a pool of VMs as a whole for example preparing disk images, setting up virtual networking, and orchestrating allocation, management, and deallocation of VMs, regardless of the underlying VMM. The solutions acting at this layer are able to provide IaaS, which can range for example, from the simple VM with a customized guest OS, to a pool of VMs interconnected each other by a private virtual network. Example of software solutions acting at this layer are OpenNebula [3], and CLEVER [4].

Instead, the CM layer represents a higher degree of abstraction. At this layer, cloud SPs, using an infrastructure acting at the underlying VIM, develop their business logic building their own cloud platform allowing them to create their services. Such services are typically more evolved than the ones classified as IaaS, and they can be categorized as PaaS and SaaS. Users which run these services are not aware on where they are deployed. An example of software solution acting at this layer is Claudia [5], instead another solution simultaneously acting both at VIM and CM is Eucalyptus [6].

Considering the aforementioned layers, the one which is today the most difficult to develop is the CM layer. In fact, cloud architects when designing a software platform working at CM layer and the services it provides, have to plan fault tolerance, configuration, accounting, performance, and security management. Depending to the business logic complexity that cloud architects want to achieve, the design of services is not so trivial. In this context the DIME model can greatly facilitate the designing, implementation, and maintenance of a seamless environment of CM layer where complex cloud-based services can be executed. Moreover, the DIME paradigm relieves the developers to use a CM middleware for developing it business logic.

Figure 2 depicts the logical scheme of a cloud SP which builds its business logic by means of a virtual DIME network deployed on a IaaS provided by a Virtual Infrastructure Provider (that can be Amazon or a private infrastructure built using a middleware of VIM layer).

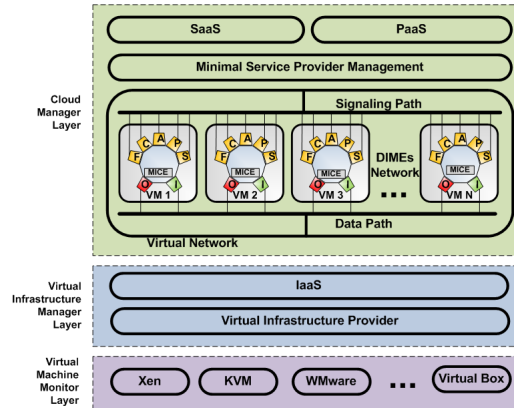


Figure 2. Virtual DIME Network in a three-layer cloud computing stack.

The SP delivers PaaS and SaaS to their users (e.g., organizations, universities, ICT societies, other cloud SPs, desktop and mobile users) instantiating them on a virtual DIME network. We assume that the DIME network is deployed in a IaaS and that each DIME is deployed in a VM with a “just enough” guest OS. Using the virtual network which interconnects the VMs, DIMEs interact with each other using the signaling and data paths. From the cloud provider point of view, a basic service can be deployed within a DIME through the “C” thread of FCAPS and executed by the DIME’s computing component, i.e. MICE. Providers can also compose a service using other services creating a workflow in which the I/O of the involved DIMEs are interconnected by the data path. In this scenario, the cloud SP needs only a minimal service administration for arranging and managing workflows, by exploiting the signaling path of the virtual DIME network. In this manner, by using the signaling path, the cloud SP can smoothly control cloud-based services either individually or as a whole

performing supervision, addressing, alerting, and mediation tasks.

In order to clarify the ideas about how a cloud-based service can be arranged using DIMEs, we describe an example of SaaS for video transcoding organized as a DAG workflow of DIMEs.

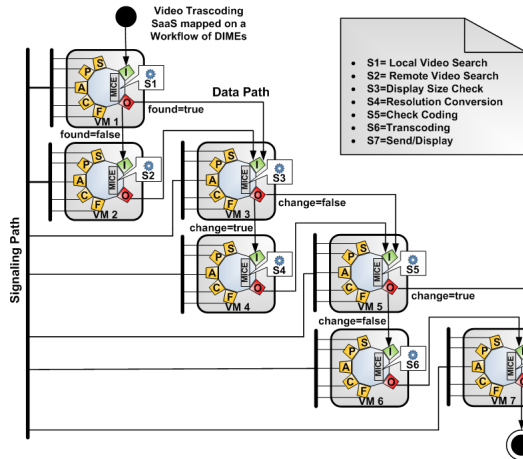


Figure 3. Example of workflow for a Video Transcoding SaaS.

As depicted in Figure 3, the SaaS is formed combining seven DIMEs each one executing a different service. For example, the DIME deployed on VM1 runs service S1 performing local video search, the DIME deployed on VM2 runs service S2 performing remote video search, the DIME deployed on VM3 runs service S3 performing display size check, and so on. When a user runs the SaaS he/she first performs a local video search by means of S1. If the video is not found the flow passes to S2 which carries out a remote video search, and then if the video is found the flow passes to S3 which check the video size. If the video size has to be changed the flow passes to S4 and then to S5 which performs a check coding. If the coding has to be changed the flow passes to S6 which transcodes the video passing the output to S7 which allows to display/download of the video. The interesting thing of such an approach is that the whole flow can be continually monitored by the SP through the signaling path.

III. SECURITY ISSUES IN A DIME NETWORK FOR THE ARRANGEMENT OF CLOUD SERVICES

In this Section after a general discussion on security in cloud services, we will analyze the security requirements of a Virtual DIME Network.

A. Security Issues in Traditional Cloud Services

Security and privacy are important aspects, especially when important data resides on the cloud's servers. Loss or leak of data can not just cause loss of revenues but also legal actions [7]. In particular, when handling personal

data, certain regulations may be applied. The EU's data protection law, states for example, that data may only be stored in certain countries with adequate protection [8]. Due to the absence of standards, cloud security, data privacy and ownership are approached differently by each provider [9].

Generally, encryption and authentication should be used on all cloud services. Encryption can guard, for example, against interception between VMs at network level [10]. Due to the low-level of IaaS, the customer has most control over the security compared with PaaS and SaaS. When using PaaS the customers may be able to craft their own authentication system or adapt other parts of the system. However, below the application level, security is dealt by the provider, who often gives little or no information about their practices [10]. When using SaaS the user has to rely even more on the provider to implement sufficient security mechanisms.

Most cloud services can be accessed with a web-browser and the standard HTTP is used to connect to the cloud. To provide encryption and secure identification of the server SSL/TLS is used. Further security approaches used for authentication and authorization include Public Key Infrastructure (PKI) and X.509 SSL certificates [9]. However, these mechanisms need to be implemented properly. The Amazon EC2 uses public-keys for authentication [10]. For hybrid clouds VPNs are used [11]. The Amazon Virtual Private Cloud (Amazon VPC) service does this.

B. Authentication and Authorization in Virtual DIMEs

In a Cloud environment, services are offered by the cloud manager as IaaS, PaaS or SaaS by means of specific middlewares that manage their deployment, contextualization and user authentication. The DIME networks we introduced, aim to offer a new computing model in which a SP deploys services as workflows.

In order to ensure the interaction among all the network components and the users who access the services, it is necessary to introduce a mechanism able to ensure security, enforcing authentication and authorization constraints. According to the DIME computing model, thanks to the FCAPS employed for exchanging control signals among the distributed software components, this task may be accomplished by the thread "S" connected on the signaling path.

A SP may arrange a network of DIME using the VMs made available by one (or more) Virtual Infrastructure Provider(s). The DIME network will be thus employed to create different workflows where the end-users will be able to instantiate their own service instances. To enable the deployment of the DIME network, the SP must authenticate itself on the Virtual Infrastructure Provider. Once this authentication is accomplished, the SP will be able to create the DIME network on which one (or more) workflow(s) will be arranged.

Since different end-users might exploit a given workflow for creating their own service instances, an authentication mechanism must be employed for identifying the end-users able to interact with the SP. Each end-user, in fact, for accessing the SP's resources need to be recognized within the SP domain. Furthermore, the SP must be able to provide access to the service instances only to the end-users that have deployed them. Thus, in order to allow the interaction among the end-users and the DIME network, two different levels of control are needed: the authentication level, to check the end-users identity and enable the interaction with the SP resources (i.e. the DIMEs network); the authorization level for checking which specific workflows and service instances each end-user can access, and which operations can perform on them.

Within such a dynamic DIMEs scenario, new DIMEs may join a given network arranged by an SP, while others may leave it: the workflows offered to the end-users by the SP, may thus scale according to the requested number of instances or other decisions made by the SP itself. If we consider each DIME as a "resource", the end-users need a mechanism for granting access to them for instantiating their services. The simplest solution would be to create end-users credentials for each DIME of the network that has to be accessed. Even though this solution is conceptually straightforward, its implementation could be difficult to achieve, due to large number of accounts that have to be created.

Our approach to address the end-user authentication problem within a SP domain, conveniently takes advantage of Single Sign On mechanisms with the employment of Identity Providers (IdPs). Solution offering IdP services are for example Shibboleth, OpenID and Security Assertion Markup Language(SAML).

As will be explained in the following, when a DIME network is created on a IaaS, an IdP for that specific domain will manage all the staff related to the users authentication. This means that all the DIMEs of a specific domain will be trusted with their IdP and the end-users will just need an account on the IdP for gaining access to the resources of the SP domain. With this approach, it is not needed for the SP users to get an account on each DIME and to perform the authentication each time a single DIME has to be accessed. Furthermore, thanks to the SSO, once the authentication has been performed with the IdP and a security context has been created, each DIME trusted with the IdP may be accessed.

The great advantage of this solution consists of the ability of including new DIME within the network in a dynamic fashion: if the new DIME relies on the IdP for the authentication task, the users that already performed the authentication with such an IdP will not need to have specific accounts on the new DIME and will be able to access it by mean of the security context already established.

A further advantage of this approach may be related to

the creation of cross-domain DIME networks. In the same way a DIME network is composed of computing elements belonging to the same IaaS, it could be possible to create a computing network using DIMEs instantiated on separated domains. The management of the new DIME component will be easy and immediate if it will be configured (using FCAPS) to perform authentication relying on the IdP of the existing DIME network. Therefore, also in this scenario, the SSO authentication may be exploited taking advantage of its security, flexibility and scalability.

The SSO authentication allows users to easily access the resources of a SP. Since this latter may deploy different workflows on the arranged DIME network, each offering a different service to different users, an authorization mechanism has to be integrated in the DIME network infrastructure. Using such a mechanism, it should be possible to control the users access to the instances of the services running within the DIME network. As stated, since each workflow may be used for allocating different service instances, a given user must access only the ones belonging to him. Furthermore, when a user interacts with its service sending commands to the DIMEs, it could be possible to select which operations may be performed, depending on a set of policies associated to the user. To implement the above mentioned capabilities, we propose the employment of a profile management system. As will be explained in the following, it might be implemented using XACML [12].

IV. THE DIME SECURITY SCENARIO

In this Section we will present a concrete scenario where the concepts of authentication, authorization and Accounting (AAA) will be specifically pointed out.

A. AAA in a Cloud Virtual DIME Network

When a user wants to access a service instance running on a DIME network, according to the security mechanisms we already highlighted, will first prove its identity interacting with the IdP deployed within the Virtual Infrastructure. Subsequently, the Authorization Manager will check if the set of resources the user would like to access is included within his profile. Finally a log of all the performed operation will be produced and registered within the Accounting Repository. Looking at the Figure 4, we can notice how the network of DIMEs implementing a workflow is interconnected with other three components, deployed on the Virtual Infrastructure from the SP:

- The domain Identity Provider (VM Z),
- The Authorization Manager (VM Y) and
- The Accounting Repository (VM X).

These three components are deployed on the Virtual Infrastructure on independent VMs and are able to interact with the signaling network by means of *Signaling Adaptors* that is able to interpret the messages coming from the DIMEs. As discussed above, the authentication is the first step that

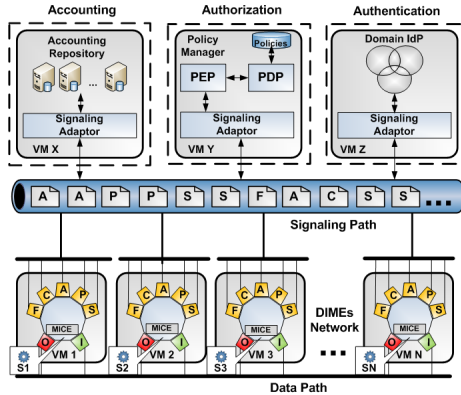


Figure 4. AAA scenario in a virtual DIME Network Architecture (DNA).

has to be performed when a user wants to access the DIME resources: the user, in order to prove its identity for accessing the DIMEs of the SP domain, employs the signaling overlay network (more specifically, the thread “S” of FCAPS). If the user is trying to access the domain resources for the first time, the computing node that receives the user request will redirect him to the IdP acting as trusted third party for the whole SP domain. The user will interact with the IdP for proving his identity and, if the authentication is performed correctly (i.e. the user has valid credentials for accessing the DIMEs of the domain), a security context will be established. Using this token, the user will be able to access the domain resources. Thanks to the SSO, when a new DIME has to be accessed, if the authentication has been correctly performed, and a security context already exists for that user, any further authentication will not be needed. Furthermore, if for some reasons a new computing resource is needed by the SP, it might be easily configured for relying to the same IdP for the authentication process thus avoiding tedious environment re-configurations.

As discussed, thanks to the security context establishment, a user may potentially access all the resources of the DIME network. However, this is not what we actually expect from the scenario: as remarked, each user must be allowed to access only his services instances deployed within a given workflow. The way this task is accomplished relies on the employment of the Authorization Manager. The Authorization Manager has to be designed to be compliant with the XACML policy language [12].

XACML is the OASIS standard language for the specification of authorization policies whose formalism for specifying Access Control (AC) policies is based on four components: Attributes and Functions, Rule, Policies and Policy Set.

- *Attributes* are characteristics of subjects, resources, actions or environments that can be used to define a restriction; *Functions* are possible operators which can be used for normative data-types.

- *Rules* are the basic elements of a Policy. A Rule identifies a complete and atomic authorization constraint
- *Policies* are combination of one or more rules
- *Policy sets* are combinations of multiple policy which represents the conditions to apply in case the authorization decision has to consider AC requirements of multiple parties.

XACML does not only provide language for policy specification: it also provides a method for evaluating a policy based on the values of the policy attributes associated with a request. The process for evaluating and enforcing AC policies is based on two main entities: Figure 4 shows the *PEP* and *PDP*. The first is in charge of evaluating applicable policies and returning an authorization decision. The latter is the entity performing AC by enforcing the stated authorizations.

For example, if we assume that the Policy Repository contains the following Policies:

- 1) *User X can deploy services on DIME 1*
- 2) *User X can access services on DIME 1*
- 3) *User Y can switch the output of the DIME 2*

When User X want to deploy a service on a workflow that includes DIME 1, it contacts the involved computing node. This latter, in turn, using the signaling network overlay, sends a message containing the access request to the Authorization Manager. Such a request is caught from the PEP, eventually reformatted in a XACML compatible way and forwarded to the PDP, which evaluates the policy rules applicable to the request according to the set stored within the repository. The result of the policy evaluation (Permit, Deny or Not Applicable) is returned back to the PEP. It analyzes the policy evaluation result and takes the final authorization decision concerning the request. The authorization decision is forwarded to the DIME 1 using the signaling network: in this specific example, since we assumed that User X requests a service deployment on DIME 1, he will be allowed to perform the requested operation. Thanks to the self-management capabilities of the DIMEs, all the performed operations will be also forwarded to the Accounting Repository using the “A” thread.

In the following we will pay attention specifically on the Authentication process. Further details about the other tasks may be case study of future works.

B. Workflow Interdomain Authentication

The authentication scheme using the concept of IDP, besides addressing scenarios where a user gains the access to the DIMEs belonging to the same administrative domain, also allows to address scenarios where a user is able to gain the access to DIMEs belonging to different administrative domains. This enables the arrangement of interdomain workflows addressing different real situations. One of these situations is when an user wants to develop a

distributed DIME workflow for arranging a PaaS which takes advantages from the geographical place where each service composing the workflow is running, for example gathering local information. Another possible situation is when a user for economic reasons, in order to build his/her workflow wants some DIMEs hosted in a cloud, e.g., Amazon, and some other DIMEs hosted in another cloud. For example, this latter situation could be motivated by the fact that Amazon offers affordable charges for running a particular service included in the workflow.

Figure 5 depicts a scenario of a workflow spread over the DIMEs belonging to different administrative domains. Domain A and domain B respectively represent two different

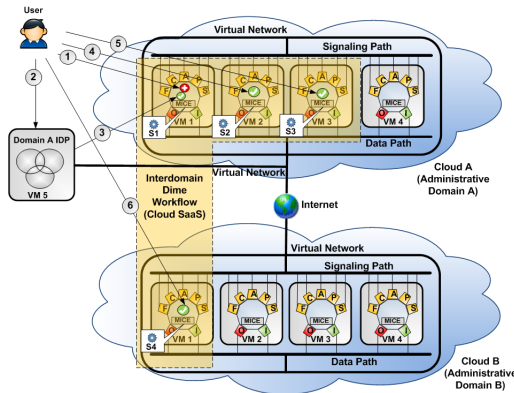


Figure 5. Interdomain workflow scenario.

clouds whose virtual networks are interconnected through the Internet. The interdomain workflow includes four services: S1, S2, S3, and S4. The first three are respectively executed by three different DIMEs hosted in VM1, VM2, and VM3 of domain A. These three DIMEs are trusted with the IDP of Domain A hosted in VM5. Furthermore, let us suppose that S1 is the entering point of the workflow (i.e., the service from which the user begins to use the workflow). In step 1, the user tries to access to the VM1 DIME whose MICE runs S1. In step 2, the thread “S” of VM1 DIME redirects the access request to the IDP hosted in VM5 which is connected on the signaling path. In this example, for simplicity we do not provide details about authorizations, supposing that the user, after authentication, has the full control of the DIMEs of the whole workflow. In step 3, through the signaling path, the thread “S” of VM1 DIME establishes a trust context, so that the user can gain the access to the VM1 DIME. In steps 4 and 5 when the user wants to access to VM2 and VM3 DIMEs, no further authentications are needed because the thread “S” of the two DIMEs realize that a trusted context already exists for the user. Even though, S4 is placed in domain B, as the VM1 DIME has been instantiated setting up the thread “S” to be trusted with the domain A IDP, the user can gain the access to S4 without any further authentication. In this manner, the

user performing the SSO authentication once can gain the access to all DIMEs in which the workflow is deployed.

V. CONCLUSIONS AND REMARKS

The DIME Network Architecture (DNA) can greatly facilitate the designing, implementation, and maintenance of a seamless cloud environment where complex cloud-based services can be arranged on DIME workflows. In this paper we exploit the signaling channel, FCAPS, and the self management features of the DNA for the achievement of SSO Authentication across multiple administrative domains, policy-based Authorization using profiles, and Accounting. More specifically, we focused on the SSO Authentication whereas Authorization and Accounting will be further analyzed in future works.

REFERENCES

- [1] P. B. Maxine Singer, *Genes & Genomes: a Changing Perspective*. University Science Books, Mill Valley, CA, 1991.
- [2] R. Mikkilineni, “The Network-Centric Computing Paradigm for Multicore, the Next Big Thing?”, *Convergence of Distributed Clouds, Grids and Their Management*, 2010, <http://computingclouds.wordpress.com>.
- [3] B. Sotomayor, R. Montero, I. Llorente, and I. Foster, “Resource Leasing and the Art of Suspending Virtual Machines,” in *High Performance Computing and Communications (HPCC)*, pp. 59–68, June 2009.
- [4] F. Tusa, M. Paone, M. Villari, and A. Puliafito., “CLEVER: A Cloud-Enabled Virtual Environment,” in *15th IEEE Symposium on Computers and Communications (ISCC)*, pp. 477–482, June 2010.
- [5] MORFEO Claudia Project, <http://claudia.morfeo-project.org>.
- [6] D. Nurmi, R. Wolski, C. Grzegorzczak, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, “The Eucalyptus Open-Source Cloud-Computing System,” in *CCGRID*, pp. 124–131, May 2009.
- [7] P. Murray, “Enterprise grade cloud computing,” in *WDDM’09: Proceedings of the Third Workshop on Dependable Distributed Data Management*, 2009.
- [8] European Commission: Justice and Home Affairs - Data Protection. <http://ec.europa.eu/>.
- [9] L. Youseff, M. Butrico, and D. Da Silva, “Toward a unified ontology of cloud computing,” in *Grid Computing Environments Workshop, GCE’08*, 2008.
- [10] C. Weinhardt, A. Anandasivam, B. Blau, N. Borissov, and et al., “Cloud computing a classification, business models, and research directions,” vol. 1, no. 5, pp. 391–399, 2009.
- [11] L. Qian, Z. Luo, Y. Du, and L. Guo, “Cloud computing: An overview,” in *CloudCom’09: Proceedings of the 1st International Conference on Cloud Computing*, pp. 626–631, 2009.
- [12] XACML, <http://www.oasisopen.org/committees/xacml>.