A DECENTRALIZED APPROACH FOR CONTROLLED SHARING OF RESOURCES IN VIRTUAL COMMUNITIES

E. Bertino
Dipartimento di Informatica e Comunicazione
Universitá degli Studi di Milano
bertino@dico.unimi.it

E. Ferrari[1], A.C. Squicciarini[2],
[1] *Dipartimento di Scienze Chimiche, Fisiche e Matematiche*
Universita' degli Studi dell'Insubria, Como
[2] *Dipartimento di Informatica e Comunicazione*
Universitá degli Studi di Milano
elena.ferrari@uninsubria.it; squiccia@dico.unimi.it

Abstract A Virtual Community is a composition of heterogeneous and independently designed subsystems, focusing on large-scale resource sharing. One of the key aspects of virtual community management, is how to enforce a controlled and selective sharing of resources in such a dynamic and decentralized environment. The traditional way is to have all the access requests mediated by a trusted third-party. However, this approach has the drawback that the third party can become a bottleneck for the whole system. By contrast, in this paper we propose a decentralized approach to virtual community management that relies on a controlled sharing of rights and duties among community members. In the paper, besides dealing with community policy specification we provide a framework able to manage all the phases of the the community life.

Keywords: Virtual communities, decentralized systems, selective and controlled resource sharing, policy specification.

1. Introduction

A Virtual Community is a composition of heterogeneous and independently designed subsystems, focusing on large-scale resource sharing, in-

novative applications and in some cases high performance computation. The sharing that we refer to is the direct access to computers, software, and data emerging in fields like science, industry and engineering. Several open issues need to be addressed, such as how to manage access policies to coordinate resource sharing, how to establish a community, how to ensure that member communities respect community policies and so on. Protocols to define how to establish sharing relationships between participants must also be enforced: each member entering into a community has to agree on what it is permitted to do and also what it is obliged to do. Resource sharing must be carefully controlled, with resource providers and controllers defining clearly and carefully what is shared, who is allowed to share and the conditions under which the sharing occurs. Resource sharing is therefore conditional, each provider has to define and publish the conditions under which it makes its resources available. Such conditions (or policies) must always be consistent with the community policies, regulating the whole organization. Further, once resource sharing is defined, the member is obliged to give access to its services according to the stated rules. Since the need to set up a collaborative sharing of resources is fundamental to many diverse disciplines and activities, our work focuses on suggesting a broad class of strategies to establish an efficient virtual organization. The goal of our work is thus to define an overall architectural framework for a collaborative environment, characterized by a flexible and dynamic structure where participants share resources in an efficient and decentralized way. The principle we apply to address decentralized control is based on the concept of *sharing* rights and duties among all members of the community. In particular, we introduce the concept of *witness,* that is, a community member that can act as a trusted third party monitoring the exchange of resources between two parties. Different from other proposals [4, 6] the witness can dynamically change and is not empowered to control all community issues, thus it cannot become a bottleneck for the whole system. Decentralization of responsabilities is obtained through the use of administrative credentials. Indeed, in addition to the witness, we also introduce the concept of *community guard,* that is, a member responsible of accepting or refusing the joining of new members. Due to lack of space, in the paper we mainly focus on the basic components needed for enforcing a controlled sharing of resources in a virtual community. Related work (e.g., [4]) mainly focus on how to manage decentralized policies for heterogeneous resources that are not under a centralized control. However, none of them addresses issues as members entitlements, delegation of entitlements, violation detection, and sanctioning mechanisms [5]. By contrast in this paper we deal with some of these

aspects and devise strategies to setup communities in an effective and decentralized way. The remainder of this paper is organized as follows. Next section presents the basic elements of a virtual community. In particular, Section 2.3 presents the policy language we have developed to encode community and local policies. Section 3 deals with processes underlying a community. Finally, Section 4 concludes the paper.

2. Basic Elements of a Collaborative Environment

Formally, a collaborative environment *CE* can be modelled as a tuple $\langle S, \mathcal{AC}, R, NS \rangle$, where S is the set of **subjects** belonging to the community, R the set of shared **resources,** and $\mathcal{AC}$ represents the set of **administrative credential types** defined throughout the community. Finally, *NS* is the **normative state,** that is, the set of policies, directions and rules regulating the community. We assume that our framework supports a key-management infrastructure such as VersaKey [7], where each subject is actually identified by its public key certificate. Moreover, each subject is qualified by means of credentials. Credentials are digitally signed certificates issued by trusted third party authorities stating properties and attributes of their owners. In what follows, we illustrate the basic building blocks of a virtual community.

2.1. Administrative credentials

Subjects of a *CE* can act as service providers as well as service consumers, and also, under certain conditions detailed in the following sections, community controllers. To regulate the assignment of roles across the community, the community also relies on a set of pre-defined administrative credentials $\mathcal{AC}$ that correspond to specific roles that need to be played by one or more community members for the correct functioning of the community. The number of subjects to which each administrative credential is assigned can vary from 1 to k, where k denotes the whole population, on the basis of the rules defined in the normative state. Unlike traditional credentials, administrative credentials are issued by the community itself when the subject subscribes to the organization. Both conventional and administrative credentials are expressed using an XML based language [2]. The administrative credential types we refer to are the following:

Resource provider. It is the credential assigned to all the subjects sharing their resources across the community. A resource provider can either directly manage disclosure and access to its resources, or

delegate this task to one or more resource managers, by issuing administrative credentials.

Resource Manager. It is the credential assigned to all the subjects entitled to manage a resource by a resource provider. Resources managed by a resource manager can belong either to the community or to a third party that delegates resource administration. The corresponding credential will specify the name of the subject, a link to the profile of the resource it manages, and the owner of the resource who delegates the resource management.

Witness. This credential type is assigned to subjects entitled to monitor that a specific resource request by a subject to a service provider has been correctly processed according with both local and community policies. A witness must then be able to detect possible community laws violation as they occur in the processes it witnesses. Witnesses can also be in charge of exercising additional functions, for instance, they may also randomly control sharing processes that they do not directly witness.

Community guard. It denotes the members empowered to accept or refuse the joining of new members. A community guard has the task of eventually issuing administrative credentials to the applicants. Moreover, it is in charge of checking compatibility of applicant local policies with community laws and, under certain circumstances, certifying the validity of local policies.

The above set of administrative credential types can be extended when needed. Each subject may possess one or more administrative credentials. The set of members possessing a specific administrative credential can share a special key to exercise administrative functions. Once the member receives one of the administrative credentials and the corresponding key, it is authorized to execute all the associated functions. The whole description of rights and duties associated with each administrative credential is formalized into a set of XML documents published in the community repository. In the following given a subject $s \in S$ and an administrative credential type $ac \in \mathcal{AC}$ we denote by $belong(s, ac)$ the fact that s is associated with a credential of type ac.

2.2. Community resources

The key element characterizing a community is the set of resources shared by its members. Resources typically have different enquiry mechanisms and capabilities that vary according with resource characteristics.

Since different policies can be adopted to regulate each type of resource we first need to provide a taxonomy of resource types. However, because of the rapidly growing number of collaborative applications and the rapid evolution of distributed systems the adopted resource characterization, borrowed from [3], is partial and open to update. The set CR of resource types we refer to consists of the following elements: CR={ *Computational Resources, Storage Resources, Network resources, Code Resources, Catalogs*}. To make policy specification easier we assume that each resource is univocally identified and has a set of characterizing properties that can be exploited in the specification of policies. For instance, a storage resource can be characterized by space availability, disk bandwidth, the types of data that can store, the mechanisms used for file transfer and backup. Like for subjects, resource properties are collected into credentials associated with the resources themselves. Resource credentials are issued by community guards when the corresponding resource is included into the set R of resources shared across the community, and they are stored at resource providers site. In the following, when no further specification is added, with the term credential we refer to subject credential, whereas we use the term resource credential to refer to a credential associated with a resource. Resources may either belong to community members or to the community itself. Resources belonging to specific members are called *local* resources, whereas resource belonging to the community are called *community* resources. Local resources are under the owner control, however their administration can be eventually delegated to other members via administrative credentials.

Example 1 *Suppose that a University has established a collaboration with a research center and a laboratory to cooperate on a common research project. Suppose moreover that the project is sponsored by EU and that the consortium has bought hardware and/or software instruments with EU funds. Such resources are an example of community resources. Moreover, suppose that the participating laboratory makes available its server for storing community data. The server is then an example of local resource.*

Resources can be further classified into *on-duty* and *on-choice* resources, depending on whether the corresponding providers are obliged to share the resource or can deliberately choose to share them. When a community resource is added one or more members are chosen to act as providers, and therefore they are in charge of administering the resource. All community resources are on-duty, since they must be granted to community members and can not be released for use on discretion of

their managers. By contrast, local resources can be either on-duty or on-choice, depending on the statute in force and on member availability.

2.3. Virtual community policies

Sharing resources across a virtual community must be regulated by both *community* and *local* policies. Local policies are basically expressed in term of properties and capabilities that requesters have to possess in order to obtain resources. Such capabilities are stated as conditions against the requester credentials. Community policies are access control policies regulating access to resources belonging to the community itself. Besides local and community policies, sharing of resources is also regulated by *community directions,* which are high-level instructions defining minimal conditions that resource providers have to satisfy while granting local and community resources within the community. Both community policies and directions are part of the normative state of the community, as explained in Section 2.4. Directions can be of two different kinds, *positive* or *negative.* Positive directions quantify the resources that must be shared, specifying who must be granted the access and, eventually, the frequency or the direction validity. By contrast, negative directions are used to specify denials such as services not permitted across the community or subjects that must not be allowed to receive some resources. Since directions only give some general instructions, resource providers are allowed to enforce more specific policies as long as they do not conflict with any community direction. A community direction can be formally defined as follows.

Definition 1 (Community direction). *Let* $CE = \langle S, \mathcal{AC}, R, NS \rangle$ *be a virtual community, and let CR be the set of resource types characterizing R. A community direction* **cd** is a *tuple of the form:*
$\langle typeR, resq, temp, credset, sign \rangle$ *where:*
- *type* $R \in CR$ *denotes the type of resources to which the direction refers to;*
- *resq (resource quantification) specifies the set of conditions that providers of resources of type typeR must satisfy;*
- *temp is a temporal expression denoting a time interval or a periodic expression, specified according to the formalism proposed in [1];*
- *credset is a set of credentials identifying the subjects to which the direction refers to;*
- *sign* $\in$ *{positive, negative} specifies whether the direction is negative or positive.*

Resq is expressed as a set of conditions against resource properties specifying the features of the resource that has to be granted. Such conditions

are expressions of the form "*property_name op a*", where *property_name* is a resource property (contained in the corresponding profile), *op* is a comparison operator, and *a* is a constant or an expression compatible with the property value. Note that it is not required to constraint all the elements of a resource credentials. Properties that are not constrained can be autonomously regulated by local policies of resource providers. The *Credset* component is a set of credential type names denoting credentials needed by subjects in order to be able to access the resource under the specified conditions. Finally, the time interval component defines the period the direction has to be enforced, and/or the frequency (e.g., every Monday) the resource must be available for sharing. *Credset* and *temp* are optional elements. When they are not specified they can be locally regulated by resource providers.

Example 2 *Suppose that a university grants both students and professors 10 gb of disk storage during the academic year. Moreover, suppose that students are not allowed to store file containing gif images and can not use fast internet providers during week-ends. The corresponding directions are formalized as follows:*

- ⟨`diskStorage, size = 10gb, [1/1/2003, 1/1/2004], {students, teachers}, positive`⟩
- ⟨`diskStorage, datatype = gif, [1/1/2003, 11/1/2004], {students}, negative`⟩,
- ⟨`Network, bandwidth = 256 Kb, [Saturday, Sunday], {students}, negative`⟩,

The second and the third directions are examples of negative directions constraining resources available for students.

We assume that community directions are enforced for all community providers. However, Definition 1 can be easily extended to keep track of the service providers to which the direction refers to. We assume that community directions are always consistent, that is, it is not possible that community directions are in conflict among themselves. This means that we assume that consistency of directions is checked each time a new direction is added or an update occurs to an existing direction.
Policies regulating disclosure of resources can be either *strong* or *weak*. *Strong* policies regulate disclosure of on-duty resources, that the members are obliged to share. Strong policies are digitally signed by a community member entitled to perform such a task (typically a guard or a community founder), which signs them when the corresponding resource is added to the list of on-duty resources and the corresponding resource credential is issued.

Example 3 *Referring to Example 2, consider a mass storage provider obliged to share 10 gb of its hard disk. Since the sharing of resource is a condition for membership, the corresponding policies will be strong and local.*

Each time a *strong* policy for a local resource is entered or needs to be modified, it must be validated by the signature either of a guard or of a community founder,which checks its consistency with the community directions. Updates can be executed on provider initiative or to comply with new community directions. Referring to Example 2, if the direction changes asking for 15 gb of disk storage, all the corresponding policies that conflict with this new direction have to be updated by the providers and validated by one administrator entitled to sign strong policies. Similarly, community strong policies are updated upon proposal by one of the managers of the resource, but the updates need to be validated by the whole set of managers.

Weak policies are local policies defined by providers for governing on-choice resources. Such policies govern disclosure of resources that the provider is not obliged to grant. Providers can autonomously update weak policies without asking for any validation by community administrators. They only have to inform the other members whether an update to the policies occurs broadcasting a message to the rest of the community. In case of a not correct enforcing of weak policies, sanctions will be lighter than in case of no correct enforcing of strong policies.

Example 4 *Suppose that the server of Example 3 makes available 25 gb of its hard disk. The 15 gb offered in addition to the 10 gb that the server is obliged to grant are regulated by local weak policies.*

Policies are formally defined as follows.

Definition 2 (Policy). *Let* $CE = \langle S, \mathcal{AC}, R, NS \rangle$ *be a virtual community. A policy* **p** *for CE is a tuple of the form:*
$\langle r,$ *rescond, subjcond, grade, time, type*$\rangle$, *where:*

- $r \in R$ *denotes a resource ID;*
- *rescond is a list of conditions on resource credentials, specifying the features of the granted resource;*
- *subjcond is a list of credentials and/or credential properties, stating credentials and conditions against them that must be satisfied in order to obtain access to* r*;*
- *time denotes the validity period of the policy;*
- *grade* $\in$ *{strong, weak} specifies whether the policy refers to an on-duty, or on-choice resource;*

- *type* $\in$ {*local, community*} *denotes whether resource* r *belongs to a member or to the community itself.*

The component *rescond* of a policy specifies the resource actually shared, constraining the properties of the released resource or the features of the service granted. Since each resource is identified by a credential the list of properties consists of the attribute and tag names contained in the corresponding XML credential. The component *subjcond* specifies the subject credentials and conditions on them that requesters need to possess in order to obtain r. Such conditions are expressions of the form *"attribute_name op a",* where *attribute_name* is an attribute name, *op* is a comparison operator, and a represents a constant or an expression compatible with the attribute value. The last two components of a policy specification specify the grade and the type of policy and must be always specified. By contrast, components *time* and *subjcond* can be omitted from the policy specification. If the *subjcond* component is omitted it means that the resource may be granted to all the community members, whereas if *time* is not specified, it means that there are no temporal constraints for granting the resource.

Example 5 *With respect to the community direction of Example 2, suppose that a storage provider makes available 15gb of its disk storage, identified by code DS1, for teachers and 10 gb for students. The corresponding policies are the following:*
1) ⟨`DS1, size = 10 gb disk, {students, teachers(grade = full)}, strong, local`⟩; *2)* ⟨`DS1, size = 5 gb disk, teachers(grade = full, dep = biology), weak, local`⟩.
The first is an example of strong policy since it regulates the portion of disk storage the server is obliged to grant. By contrast, the second policy is an example of weak policy. The policy states that only full professors of the biology department can access the 5 gb of memory in addition to the amount granted by the previous policy.

Community directions and policies therefore work together to grant a correct functioning of the whole community. Directions have higher priority than local policies and therefore in case of conflict they take the precedence. For instance, a community direction can oblige network providers to provide a fast connection every day from 8 a.m. to 10 p.m. Each storage provider is empowered to autonomously define under which conditions it grants the service but it should assure the continuity of the service during the specified hours. Intuitively, a policy is in conflict with one or more directions when its enforcement can cause a violation of the direction. More precisely, a policy and a direction may conflict when:

- The policy forbids a resource access to a member which is entitled to access it by a positive direction.
- The policy allows some of the accesses permitted by a positive direction to a member, but the conditions specified in the policy are more restrictive than those stated in the corresponding positive direction.
- The policy allows resource accesses that are explicitly prevented by a negative direction.

Example 6 *With respect to Example 2 consider the following policies: 1.* ⟨**PROV1**, **bandwidth = 256Kb**, {**students**}, [**Monday-Saturday**], **strong**, **local**⟩*; 2.* ⟨**DS1**, **8 gb disk**, {**students**}, **strong**, **local**⟩*. Suppose that the first policy is specified by an internet provider, stating that it grants fast connection to students from Monday to Saturday. This policy conflicts with the last negative direction of Example 2 which denies the access on Saturday. Moreover, suppose that the storage provider of Example 5, updates the first policy reducing the disk storage availability from 10 to 8 gb. The resulting policy will be in conflict with the first positive direction of Example 2 but not with the negative one, which forbids storing gif files.*

When a conflict is detected, the provider is obliged to rewrite the conflicting policy. If it refuses to comply with this obligation it is consequently sanctioned, usually with the revoking of the provider credential for the involved resources. Note that since weak policies do not correspond to providers obligations they can never be in conflict with any positive direction. Finally, community policies community directions, typically when a positive direction is updated influencing the validity of the community policy. Community policies also may need to be updated in order to be correctly enforced. As discussed earlier, in order to be modified such kind of policies need to be validated not only by a single community manager, but by all the managers administering the same resource.

2.4. Normative state of the community

Previous work [5] introduced the concept of normative state as a set of access permissions of community members, and their obligations to provide access to their services. We extend such concept defining the normative state as a set of information about the community structure, regulating not only resource management but also policy enforcement. More precisely, the normative state contains community directions as well as policies regulating access to community resources. Additionally, it includes all those kinds of information that help the members to actively participate to community life. Such information concern aspects

like the management of the community, sanctioning mechanisms and violation punishments. The normative state can be divided into three sections corresponding to three different kinds of information concerning respectively resource sharing, management structure and enforcement mechanisms.

Resource sharing. The core of such section is the collection of community directions (described in Section 2.3) giving instructions concerning policy specifications for the shared resources. Community policies are also included, regulating access to those resources belonging to the entire community. Moreover, this section collects summary information on the resources shared within the community. More precisely it contains an entry for each shared resource specifying the owner, the type of resource and whether the resource is governed either by strong or weak policies (i.e., the member is obliged or not to share the resource). Finally, for each resource the section also lists the member entitled to manage (e.g. Resource manager) and control accesses to the resource itself.

Management information. This section collects information defining how authorities and responsibilities are, or can be, distributed within the organization. For what concerns how authorities are distributed this section keeps track of the administrative credentials assigned to each subject. Moreover it specifies rights and duties associated with each administrative credential type and the criteria to assign administrative functions to each member. To this purpose, several approaches can be adopted. A possible approach is constraining the ratios of members that must be authorized to exercise administrative function to the whole population. Another criteria might require that each member must be associated with at least one administrative credential, or establish the minimum percentage of members that must be assigned to each administrative credential. The specified constraints impacts the degree of decentralization enforced within the community. Intuitively, the more members are authorized to exercise administrative functions the more decentralized will be the resulting community.

Sanctioning mechanisms and violation detection. The last section of the Normative state defines how to detect and manage cases when members fail, or do not comply with community laws. More precisely, such section contains a classification of the possible violations with the related sanctioning mechanisms, and a description of the devised mechanisms and protocols to detect the violations.Unlike the description of the devised mechanisms, which are expected to be standardized for all the communities, the classification of illegal actions strictly depends on the reference scenario and the type of environment considered.

3. Community management

To explain how we realize a decentralized system enforcing a controlled and selective sharing of resources we now briefly detail the management process underlying a community. The steps we focus on are *community setup,* that is, how a community is started, *new member association,* that is, how new members can subscribe to a community, and, finally the process of leaving the community, that is, *leave of a member.*

- **Community setup** Several issues have to be addressed in order to setup a new community. First, a community is said to be established when the minimal conditions to correctly execute sharing of resources hold. Operatively, community founders have to publish a first version of the normative state, defining the set of administrative credentials and choosing the founders that will receive such administrative credentials. Moreover, resource credentials specifying resource properties must be issued to the resource managers for each resource of the set R of shared resources. The first issue facing community founders is then to decide key aspects of the normative state that better fit the community goals. In order to define a framework that is as adaptable and flexible as possible we do not define a unique type of community. Indeed, our framework supports the definition of several community types, regulated in different ways, and can be properly customized according to specific community requirements and needs.

- **New members association** One controversial issue about how to join a community regards the entities with whom the requesters have to interact. In centralized systems such task is performed by a specific entity which has the control of the whole community. To avoid centralized approaches where reliability of the whole community is entrusted to a single entity in our framework this role is played by a set of members referred as *community guards* (or guards, for short) whose specific task is to perform the joining phase. This phase is actually carried out as an interaction between the applicant and a guard. The aim is to verify requester properties in order to accept or refuse the subscription. Moreover, community guards eventually sign local strong policies of the applicant, in order to certify policies consistency with community directions. A guard can also assign administrative credentials, on the basis of evaluation of applicant competences and capabilities. The phase starts when a requester contacts a subject of the community to join it. If the receiver is not a community guard the

request is forwarded to a member having the *community guard* credential.

- **Leaving the community** There are two different ways to leave the community: voluntary leave or forced leave. According to the first, a member announces that it is leaving forwarding a leaving message to all community members. The remaining members update the normative state, and delete on cascade all information about the leaving member. Moreover, if the leaving member was either a community administrator or a resource provider, it is deleted from the corresponding lists, and all the administrative credentials it owns are revoked. The second way is the heaviest form of sanctioning provided by the community and consists of banning. As well as for voluntary leaving, to exclude a member all information concerning it must be deleted from the normative state. Note that if the member was an administrator and the community makes use of a community public key for signing administrative credentials a new key must be sent out according with the adopted key management system [7]. The remaining participants can then use the new key for exercising administrative functions, but not the member which just left.

4. Conclusions

In this paper we have presented a flexible framework for managing virtual communities whose goal is resource sharing based on selective and flexible policies. In the paper we have mainly focused on the language to specify policies and directions, and on the steps to deal with the various phases of the community life. The work presented in this paper is part of an on going research project. We are currently working on XML policy specification and on web services for policy management. Also, we are developing mechanisms to check local policy correctness against community policies. Finally, another interesting area to explore concern protocols and strategies to share resources among community members in a flexible a decentralized manner.

References

[1] E. Bertino, C.Bettini, E.Ferrari, and P.Samarati. An Access Control Model supporting Periodicity Constraints and Temporal Reasoning. *ACM Transactions on Database Systems,* vol 23, n. 3, pp. 231–285, September 1998.

[2] E. Bertino, E. Ferrari, A. Squicciarini $\mathcal{X}$-TNL: An XML-Based language for Trust Negotiation. *Proceedings of the 4th International Workshop on Policies for Distributed Systems and Networks,* Como, Italy, June 2003.

[3] I. Foster, C. Kesselman, S.Tuecke. The anatomy of Grid-Enabling Scalable Virtual Organizations. *International Journal of Supercomputer Applications,* 15(3), 2001

[4] L.Pearlman, V. Welch, I.Foster A Community Authorisation for Service Group Collaboration *Proceedings of the 3rd International Workshop on Policies for Distributed Systems and Networks* page 50-59, Monterey California, USA, June 2002. IEEE.

[5] B. Sadighi Firozabadi, M. Sergot. Contractual Access Control. *To appear in the proceeding of Security Protocols, 10th International Workshop,* Cambridge, UK, 2002.

[6] B. Sadighi, M. Sergot, O. Bandman. Using Authority Certificates to Create Management Structures, April 2001. *Proceedings of the 9th International Workshop on Security Protocols,* Cambridge, UK.

[7] M. Waldvogel, G. Caronni, D. Sun et. al. The VersaKey Framework: Versatile Group Key Management Department of Computer Engineering, ETH Zurich. Technical Report TIK-57, 1998.

[8] World Wide Web Consortium. Extensible Markup Language (XML) 1.0, 1998. Available at `http://www.w3.org/XML`