

VICSDA: Using Virtual Communities to Secure Service Discovery and Access

Shudong Chen, Johan Lukkien, Igor Radovanovic,
Melissa Tjong, Remi Bosman, Richard Verhoeven
Department of Mathematics and Computer Science
Eindhoven University of Technology, the Netherlands
+31 40 247 8204

{shudong.chen, j.j.lukkien, i.radovanovic, m.tjong, r.p.bosman, p.h.f.m.verhoeven}@tue.nl

Goran Petrovic
Department of Electrical Engineering
Eindhoven University of Technology, the Netherlands
+31 40 247 3708
g.petrovic@tue.nl

ABSTRACT

Service Oriented Architecture (SOA) is emerging as an enabling technology for sharing distributed heterogeneous resources on the network. Consequently, securing services is an increasing concern. Research issues include privacy protection for service providers, transparent access control for service consumers, secure service discovery and composition. In this paper, we present an access control approach which uses virtual communities to secure service discovery and access (VICSDA). Services grouped in virtual communities can only be discovered and accessed by authenticated community members. Meanwhile, services are autonomous to define their local access control policy. Moreover, behavior of these autonomous services is monitored in order to guarantee a better QoS provision. Using a virtual community overlay network on top of a SOA infrastructure, VICSDA can provide authentication, message confidentiality and integrity to secure service discovery and access. Better application performance can be achieved through VICSDA. We integrated VICSDA with a 3D video streaming application. This example provides us with some initial evidence that VICSDA is a viable solution to our target problems.

Categories and Subject Descriptors

C.2.4 [Computer-Communication Networks]: Distributed Systems – *distributed applications*.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Qshine'07, August 14–17, 2007, Vancouver, Canada.
Copyright 2007 ACM 978-1-59593-756-8...\$5.00.

General Terms

Design, Security, Experimentation

Keywords

Security, Service discovery and access, Virtual community, 3D video streaming

1. INTRODUCTION

Sharing of resources, data and functionality via digital networks becomes increasingly important. Through sharing, integration and cooperation of heterogeneous hardware and software components, distributed applications can be developed. Obviously, a lot of challenges derive from this sharing. On the one hand, users need to locate their requirements efficiently from the huge amount of shared items scattered over a network. On the other hand, it is necessary to protect providers' ownership and privacy. Furthermore, confidentiality and integrity of the exchanged messages is vital to both users and providers.

The Service Oriented Architecture (SOA) [1-6] is emerging as an enabling technology for sharing distributed heterogeneous resources on a network that may be under the control of different owners. In SOA, software components are wrapped into network-exposed services with explicitly described interfaces to provide functionalities; applications can be built by interconnecting services, leaving the binding until runtime. Famous standards at the moment are Web Services [2] in business environments, as well as Universal Plug and Play (UPnP) [3] and Jini [4] in the networked consumer electronics world.

SOA has some fundamental advantages over traditional distributed application systems. The functionality in a SOA can be reused to a high degree; relying on standards it provides a highly flexible and adaptable implementation for services and applications; eventually, it becomes possible to switch from a particular service to a different one without adaptations. At a

conceptual level, SOA is composed of three core roles [5]: *service registry* which acts as an intermediary between providers and requesters where services are stored, *service provider* which defines a service description and publishes it to the service registry, *service requester* which can use the service registry's search capabilities to find service descriptions and their respective providers. An instance of a service requester is a service seeker or a service user. In a SOA, there are three activities that the service provider and requester can perform: *publish, discover and invoke*. The service provider has to publish the service description in order to allow the requester to find it. The process in which the service requester retrieves a service description directly or queries the service registry for the required service is called *discovery*. That the service requester initiates an interaction with the discovered service at runtime using the binding details provided in the service description is called *service invocation*. In the remainder of the paper, the invocation activity of a service is also called *service access*.

A SOA may have a central service registry, for example the UDDI registry [6] adopted by web services. Services published in this service registry are supposed to be discoverable by all service requesters. This may lead to a privacy problem when a service provider only wants to reveal its services to trusted service requesters.

In addition, service providers and service requesters should have some guarantee that the agreements in the contract, for example, promised functionalities and QoS in the service description, are supported. That is difficult when a SOA lacks a monitoring and enforcement authority. Invocation happens between service providers and requesters in a fully distributed point-to-point way. Point-to-point cooperation is on the one hand increasing the flexibility, and on the other hand is subject to reducing the possibility for a service provider to provide contracted QoS since there is no enforcement mechanism, for instance, behaviour being monitored by a third party. An example here is that a service provider may cease its offered service without informing its users about it and a service user may try to use the service without paying (in whatever form).

There has been a lot of work in this field to solve some of these issues. One example is Secure Service Discovery Service (SDS) for service discovery and directory [7]. It uses a security model to provide authentication, data integrity, and access control. It utilizes public key encryption for authentication, shared key encryption for communication, and a hybrid model (access control list / capability) to provide flexible and scalable access control. However, it focuses only on the service discovery and has not considered security and desired performance in the service access stage as a goal. Community Authorization Service (CAS) [8] addresses three authorization problems including scalability, flexibility and expressibility in the grid computing area. It focuses on supporting the centralized specification of community policies governing collections of resources, such as who is allowed to read and write replicated data in a Data Grid. One downside of CAS is that it uses a single CAS server to control the entire community access which may have the problem of being a single point of failure as well as a potential performance bottleneck.

In this paper, we apply virtual communities to secure service discovery and access (VICSDA). We extend SOA with the concept of virtual communities to help protect the privacy of

service providers and include incentive for high QoS provision. Services are grouped into virtual communities [9]. Using the maintenance mechanisms of a virtual community, VICSDA can protect service providers' privacy with only allowing authenticated community members discovering and accessing their services. At the same time, services are autonomous and they should not give up their ability to prohibit access coming from users they do not trust. So VICSDA grants services themselves the right to deny access requests which may nevertheless satisfy the security rules of the virtual community. Furthermore, behavior of these autonomous services is monitored in order to guarantee better QoS provision. In this way, relying on a virtual community overlay on top of a SOA infrastructure, VICSDA can provide authentication, message confidentiality and integrity to secure the service discovery and access process. And a better application performance can be achieved through enforcement authority and services' behavior monitoring.

The remainder of this paper is organized as follows. In Section 2 we describe the concept of a virtual community overlay followed with a detailed operational explanation and design of VICSDA. In section 3 we present a practical application of VICSDA. Future directions are discussed in section 4. Finally we summarize and conclude in Section 5.

2. SECURE SERVICE DISCOVERY AND ACCESS IN VIRTUAL COMMUNITIES

Message communication between service providers and requesters on a network require distributed access control. In addition, some form of monitoring is provided to record service quality and to enhance the providers' incentives to share their services. In order to provide such a secure and reliable environment, VICSDA is designed. Based on the SOA approach, distributed authentication without putting much load on a centralized authentication server can be achieved. By organizing services into virtual communities mechanisms are put in place to monitor service quality and to provide access control. To subsequently describe how VICSDA works we first describe what we consider a virtual community in this context.

2.1 Virtual community

The term virtual community has been proposed in diverse ways. For example, people use computers to communicate and form friendships that form the basis of their virtual communities [10]. An interest-based virtual community is characterized on the basis of a shared interest, for instance, a Lords of the Rings fan club. A functional virtual community can be a group of users participating on a single application platform, such as the online game Ultima Online [11].

Our work focuses on virtual communities whose members interact through service invocation via digital networks or the Internet. In the remainder of this paper, when we discuss virtual communities we are not referring to any aggregation of people, but to the communication among them which is done through service sharing and cooperation.

We interpret a virtual community as a dynamic contract-based aggregation whose members have commonalities and interact via shared services by means of a digital network or the Internet for varied individual reasons. A virtual community is built on the concept of a digital network connection. It has rules that each

member has to follow. It provides services to members and has the potential to develop different applications through service cooperation and external orchestration.

Initiatives to form virtual communities are always benefit driven. To obtain benefit is the basic commonality that virtual community members have [9]. This benefit-driven can be discriminated into interest driven, application driven. For example, when a group of people with a common music interest form a virtual community they can then exchange classical music and opinions on new music. An application driven formation example is that several parties form a drug discovery virtual community. Through accessing the shared drug screening services, they can finish a drug research application without purchasing all the commercial databases and computing resources [12].

We are interested in investigating the SOA approach for composing applications from services. In its simplest form, a SOA consists of network-exposed services, discoverable and usable by requesters. One step further the services may cooperate to form distributed applications. Requesters do not necessarily take part in this service cooperation. Another party as an orchestrator does this external orchestration including service connection and binding. The collaboration should be agreed upon between service providers and requesters defining clearly and unambiguously about what is required and who is allowed to request. To address these, we extend the notion of SOA with the concept of virtual community.

Shared functionalities are first wrapped into services. These services are then organized in virtual communities by adding additional functionalities to them. From this point of view, a virtual community can be treated as an overlay network for the existing services. In this overlay services can only be exposed to authenticated virtual community members; it can define its local access control policy to filter access requests, and all the exchanged messages are encrypted. In addition, a reliable service discovery can be achieved through trustable recommendation based on past interactions. All activities including publication, discovery, invocation and cooperation are executed within the scope of a virtual community. Meanwhile, a virtual community service has the properties of a plain service: its external network interface still uses the SOA service interface and it uses SOA protocols such as the Simple Object Access Protocol (SOAP) [13] protocol and the HTTP protocol.

A virtual community overlay network of SOA services should have the following basic functionalities: member management, services management, distributed authentication for secure service discovery and access, registered member and service list maintenance, and recommendation for service selection. A detail description of these functionalities is shown in Table 1.

2.2 VICSDA

VICSDA is designed for providing a secure and reliable service communication environment. It is a facility for dual access control approach in field of secure service discovery and access. Firstly, it uses the member boundary of a virtual community to control the service discovery scope and filter the un-trusted or malicious access requests to services. Secondly, it treats services as autonomous entities. It allows services to select their local access control policy and to deny access requests coming from their un-trusted requesters. Thirdly, it addresses the problems associated with the complete autonomy of services. A

consequence of this autonomy could be that the entire application fails because service providers do not provide promised QoS provision. This is addressed in VICSDA as follows. In a virtual community, each service is granted a credit value to represent its reputation. Its reputation is related to its behavior. An authorized service user, for example, an orchestrator, can evaluate a service's QoS after using it. This evaluation affects that service's credit value. This value may increase because of a positive evaluation and may decrease when a negative one arrives. It is periodically checked, and when it is below a threshold the service will be taken out of the community. Using this maintenance mechanism, VICSDA can restrict services autonomy and then guarantee a better QoS provision to users and applications.

Table 1. Virtual community functionalities

Functionality	Task
Member Management	To deal with members' activities including registering a service provider or requester as a member and deregistering a member.
Service Management	To deal with service related activities including service publication, discovery and access.
Security Management	To protect the privacy of service providers and to provide secure service access. An example is to distribute and verify authentication of a service requester.
Credit Management	To deal with activities related to virtual community maintenance. For example, monitor and evaluate services' behaviour, help to maintain registered service and member lists, etc.
Recommendation	To make trustable recommendations for a service requester in the service selection stage. For example, to select a service with the best quality.
Orchestration	To offer a generic service with special roles that can be extended for achieving particular applications through external service orchestration.

Clearly, the design of VICSDA must rely on effective dynamic coupling of different collaborating functional components. SOA technology can support the independent infrastructure and allow the autonomous operation of its functional components. Consequently, using SOA as a fundamental infrastructure, a virtual community is formed through rewrapping services with additional functionalities. Using the UML notation [14], Figure 1 depicts the essential ingredients that together compose VICSDA.

The six basic functionalities that a virtual community has are implemented by multiple services and several data elements. *VCEntry* and *CertificateMgt* are accessible to any parties including requesters on the network and internal community members, while the other services are only available for community members. For example, *Repository*, *ServiceMgt*, *CreditMgt* and *Recommender* can only be invoked by authenticated community members.

VCEntry is a service to deal with members' actions inside a community including member registration and deregistration. *JoinPolicy* is a description of the agreed community joining policy. An example for a joining policy is that only those who supply actual personal information including IP address and e-

mail address can be approved to register as a member. When an entity satisfies the *JohnPolicy* and has been authorized to become a member, specific roles are assigned to it by the *VCEnter*. It could be a service requester to access services, or a service provider to register / deregister a service, or an administrator to change the threshold which is stored in the *CreditMgt*, or a combination of these example roles. As a member, it can execute corresponding actions according to its role while carrying a valid ticket signed by the *CertificateMgt*. Interactions of this member registration process can be found in Figure 2.

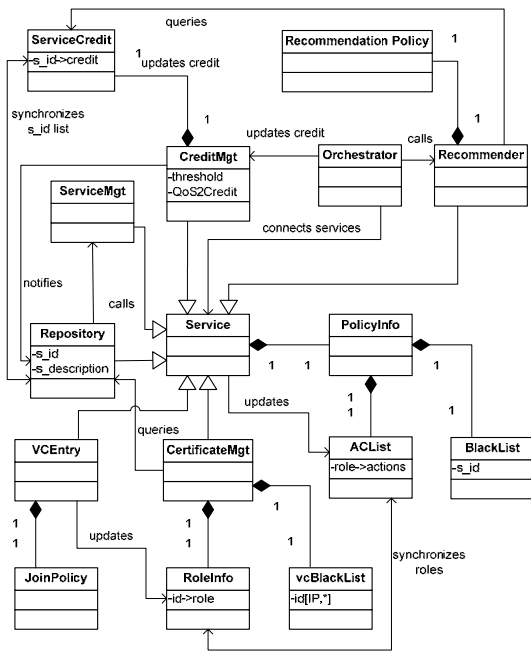


Figure 1. Ingredients of VICSDA

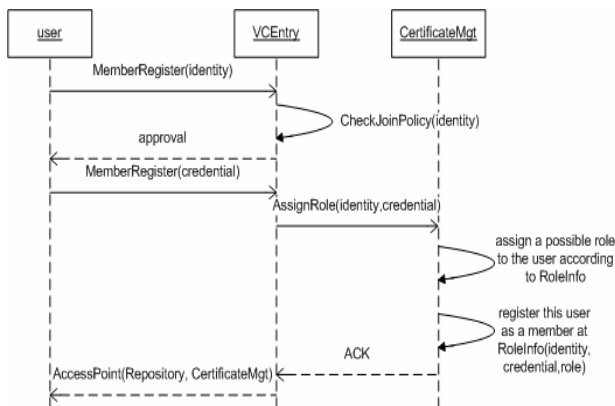


Figure 2. Member registration in VICSDA

When a member publishes services into a virtual community, those services will be registered at the *Repository* (Figure 3). The *Repository* is a service provider that is dedicated to resolve

queries. It caches service advertisements, receives queries, and performs matching between queries with available services. To execute a service publication, *ServiceMgt* will be called by the *Repository*. *ServiceMgt* can label a plain service as a community service by adding for example the *Credit*, the *ACLlist* and the *BlackList* properties to it, then advertises it to the *Repository*. *Credit* is an attribute to represent the reputation of a service related to its past interactions. A service can specify its local access control policy by editing its *ACLlist* and *BlackList*. The service assigns allowed actions to different roles. Malicious users' information can be inserted into the *BlackList*. In order to synchronize the available roles stored in a service's local *ACLlist* and the *RoleInfo* of this community, an update process can be initiated by a service provider. Figure 3 describes the detail interactions happening in a service registration process.

VICSDA uses a *CreditMgt* service to overcome the limitation of fully distributed point-to-point cooperation. The *CreditMgt* is designed to facilitate the virtual community maintenance and a contracted QoS guarantee for service cooperation. It monitors service providers' behaviour by periodically comparing their credit values with the threshold which is set by the virtual community administrator. A service providing reliable and high QoS will get its credit increased gradually while when its credit is smaller than the threshold, the *CreditMgt* will expel it from the virtual community. This service's information will be removed from the *Repository*.

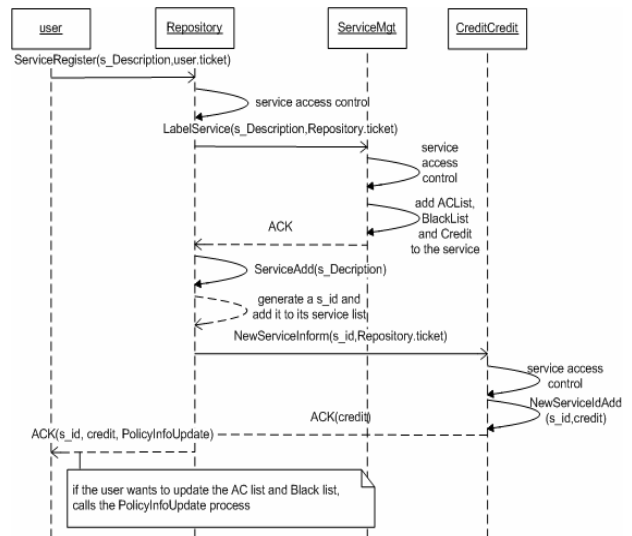


Figure 3. Service registration in VICSDA

In order to find a service, a requester issues a discovery request; in response it receives a list of matching services, for which it can call the *Recommender* for recommendations about service selection. The *Recommender* will query those candidates' credit values and make recommendations using a specific *Recommendation Policy*. In this way, VICSDA helps to execute the service discovery effectively.

The ability to provide message security during the communication is vital to provide a robust VICSDA and services. All exchanged messages in VICSDA are encrypted. Asymmetric cryptography

[15] is used for communication privacy and security. The cryptographic key pair for encryption and decryption is distributed by a trusted party namely the *CertificateMgt*.

Figure 4 illustrates how the dual access control approach for secure service discovery and access works in a virtual community overlay network. We consider the case that an orchestrator executes a service discovery at a repository as an example scenario. In this case, the orchestrator is a service requester while the repository is an accessed service. When the orchestrator submits its service discovery request to the repository, a valid ticket which is distributed by the *CertificateMgt* should be checked by the repository. This ticket is an authentication that the orchestrator is a member of this virtual community. Its role in this virtual community is indicated in this ticket. Firstly, the repository will verify the authenticity of the signature and the validity of the lifetime of this ticket. If approved, the repository will execute a second access control based on its local policy. In this stage, the repository will browse its *BlackList* to check whether the orchestrator is one of its un-trusted requesters. If not, the repository will lookup its *ACL* in order to grant the orchestrator access with authorization according to the roles in the ticket. For example, the orchestrator can invoke the service discovery action but not the service list update action. Only after the orchestrator passes the dual access control, its service discovery request can be executed at the repository side.

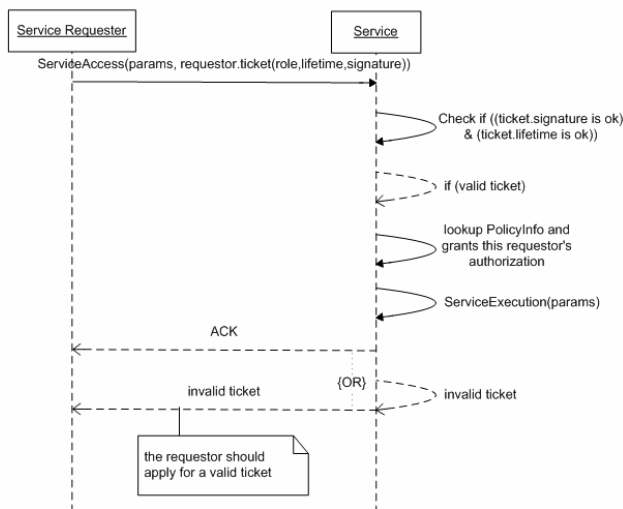


Figure 4. Secure service access in VICSDA

The same dual access control mechanism is used to each service access. Another example of its application is that after the service usage, the orchestrator can evaluate the provided QoS of the repository. In this case, the *CreditMgt* first checks the ticket of the orchestrator and then authorizes its service evaluation action. It translates the QoS evaluation into a service credit increment using the *QoS2Credit* policy and updates the corresponding service credit value.

In order to prohibit a service provider evaluating his own reputation, VICSDA defines one of the items in the *ACL* of the *CreditMgt* as that service providers cannot invoke the *QoS2Credit* action.

Alternatively, service providers or services have the right to evaluate service requesters' behaviour. In this case, service providers are autonomous entities. If a service provider treats a service requester as malicious, it can always add that requester's identity into its *BlackList*. As a result, requests coming from those who are in the *BlackList* will be rejected even if they have a valid ticket signed by the *CertificateMgt*. Suppose a provider frequently refuses providing contracted functionalities to community members, it will be expelled from the community ultimately. This is the reason why VICSDA is designed using the reputation based community maintenance mechanism.

3. INTERACTIVE 3D VIDEO STREAMING CASE STUDY

We describe here an interactive 3D video streaming application to address the feasibility of VICSDA. In particular, we integrate VICSDA dual access control with an early version of this application.

This application demonstrates how a user can change the viewpoint of a 3D video in an interactive fashion. This gives a user complete freedom as to choose where to display the video and where to control it. Figure 5 shows the physical connections between physical devices of this application.

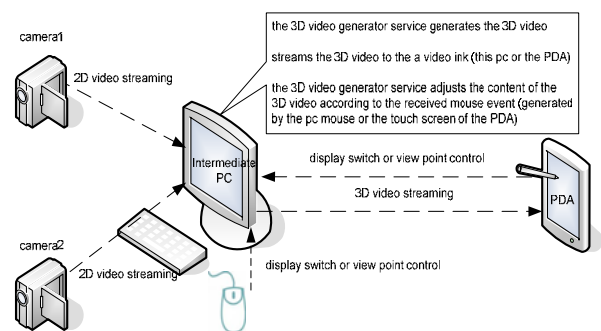


Figure 5. Interactive 3D video streaming application

A 3D video generation service generates a 3D video based on the two video streams coming from two cameras. During the video streaming a user can select a display where the video is switched to. She can bind the 3D video generation service with an arbitrary control in the environment, e.g. a mouse. She can use this control to change the viewpoint of the video. The control and the display are not necessarily at the same terminal. In this case, a mouse's event indicates her viewpoints switch requirements. These events will be received by the 3D video generation service which then correspondingly adjusts the generated 3D video's content.

An application like this one is usually developed first as a standalone program, running on a powerful platform. Mapping it to a networked context is then done using some decomposition into distributed tasks that communicate in a proprietary fashion. A problem with this is a lack of flexibility. Standards like UPnP address this by designing service interfaces such that it is much easier to develop the components independently and to let them discover each other dynamically. This leads to decomposition into well defined tasks, wrapped as services. Developing distributed

systems along these lines also leads to vulnerabilities and security problems. VICSDA solves these problems by wrapping all the components into community services and giving both users and services a ticket for dealing with each other. The new 3D video streaming application can receive a remote mouse service's events from the network which represents a user's display switch requirement or viewpoints. The remote mouse service may work on another device for example a PDA. This application addressed the following research aspects and technologies: secure service discovery and access in virtual community scope, context-aware coding in adaptive video delivery protocol [16] and a video stored somewhere in the network is displayed on the mobile device.

Concretely, we have used the following setup: two video files generated by two cameras working as the input of the 3D video generation service, one intermediate PC which provides the 3D video generation service and also acts as a sink of the 3D video stream, one PDA running as another mobile video sink, and two viewpoint control devices which are a PC mouse and the touch screen of the PDA. We reading two local files cameras represented as PC-Cameras generating two video streams of one object from different angles The architecture of this application is shown in Figure 6.

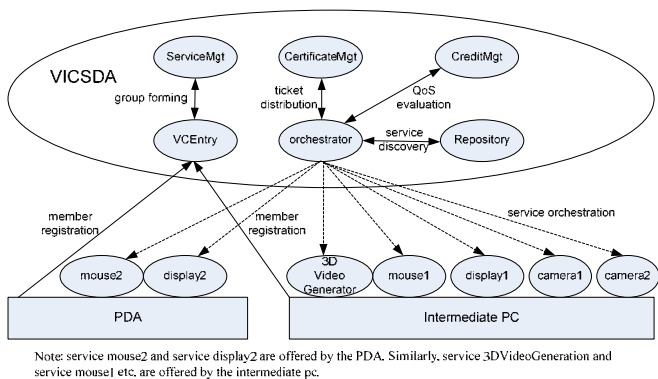


Figure 6. Architecture of the interactive 3D video streaming

All needed software components are visible as services on the network. We have modified the early 3D video generation program and wrapped it into a web service. We have also developed two mouse services. Both of them are running as web services with one running on the PC and the other running on the PDA. During the 3D video generating process, the coding of the displayed content is adapted to end users' requirements automatically. Particularly, the video signal is based on the control information of an external service: the mouse of the PC or the touch-screen of the PDA.

We use VICSDA to wrap all these services into virtual community services. So their availability is restricted to users with community credentials. We use an orchestrator to execute the 3D video streaming. It firstly discovers all required services from the repository. Then it needs to configure these services and bind them together to achieve the application. For example, it binds the 3DVideoGenerator service with one mouse service and one display service. During the course of service discovery, binding and accessing, the dual access control approach including the ticket validation and the capability granting are applied in every two communication parties. For example, the orchestrator

has to show its ticket to the repository before it invokes the actions; the 3DVideoGenerator service and the mouse service have to negotiate before they are bound together.

4. FUTURE DIRECTIONS

If VICSDA only has a single *CertificateMgt* service for community authentication then it may have the problem of being a single point of failure as well as a potential performance bottleneck. Distribution is a common solution for this, so that if the *CertificateMgt* service is overloaded or failed another peer service can start to work without affecting the system performance. To enable this functionality we are investigating the load balancing and automatic error detection / recovery related solutions. We will explore mechanisms of balancing the load of the *CertificateMgt* server and replacing it with another one to alleviate this potential problem. The best approach will depend on how to monitor the resource usage of the service and the network load. We will use a resource management service that can query a service for currently being used resources and a network load balance service that can distribute load among a set of peer services.

VICSDA utilizes the asymmetric cryptography for communication privacy and security. The decryption will take place in each message exchange. Under these circumstances, the overhead of decryption may be a vital aspect for the overall system performance. We will consider applying an efficient encryption mechanism for ticket distribution which will help to decrease the response latency of a service access process.

5. CONCLUSIONS

This paper describes the dual access control approach for secure service discovery and access (VICSDA) to solve the problems that arise in ubiquitous sharing: privacy, transparent access control and authorization. We address these problems by introducing the dual access control approach that performs fine-grained control of community policy while leaving ultimate control of local access to the services. The essence of this approach is: a) VICSDA uses the authentication functionality of a virtual community to provide service providers with privacy; b) it lets the services themselves design the local service access policy: the service can determine for itself whether it trusts a particular service requester since the community certificate authority pushes the authentication and the signature all the way to the service; c) it utilizes an auditing mechanism that through monitoring autonomous community members' behavior, avoids the potential unreliable behavior of providers. We also describe a scenario integrating VICSDA with a 3D video streaming application. This example provides us with some initial evidences that VICSDA is a viable solution to the problems we addressed.

6. ACKNOWLEDGMENTS

We thank our anonymous reviewers. Their invaluable feedback helped substantially in improving the quality of the paper. We owe our gratitude to Peter H. N. de With for his technical feedback at the implementation stage of this work. This work was supported by the research project of Freeband I-Share: Intelligent Middleware for Sharing Resources for Storage, Communication and Processing of Multimedia Data, supported by the Dutch government.

7. REFERENCES

- [1] SOA. <http://www.service-architecture.com/index.html>.
- [2] Ethan Cerami. Web Services Essentials: Distributed Applications with XML-RPC, SOAP, UDDI & WSDL. First Edition February 2002. ISBN: 0-596-00224-6, 304 pages. Publisher: O'Reilly.
- [3] UPnP Device Architecture. Version 1.0. <http://www.upnp.org/>.
- [4] Sun Microsystems. Jini technology architectural overview. White Paper. <http://www.sun.com/jini/whitepapers/architecture.html>.
- [5] H. Kreger. Web Services Conceptual Architecture (WSCA 1.0). <http://www-306.ibm.com/software/solutions/webservices/pdf/WSCA.pdf>
- [6] UDDI Version 3.0. Published Specification. <http://www.uddi.org/>.
- [7] S. Czerwinski, B.Y. Zhao, T. Hodes, A. Joseph, R. Katz. An Architecture for Secure Service Discovery Service, Fifth International Conference on Mobile Computing and Networks (MobiCom'99), Seattle, WA, pp. 24-35, 1999.
- [8] L. Pearlman, V. Welch, I. Foster, C.Kesselman, S. Tuecke. A Community Authorization Service for Group Collaboration. Third International Workshop on Policies for Distributed Systems and Networks, Monterey, CA, USA, 2002.
- [9] S.Chen, J.J.Lukkien. Freeband I-Share D1.5. 2007. Project Deliverable. <http://www.win.tue.nl/san/publications/>.
- [10] Howard Rheingold. The Virtual Community. <Http://www.rheingold.com/vc/book>.
- [11] Ultima Online. <http://www.uo.com/>.
- [12] S. Chen, L. Zhang, F. Ma. DDG: A Grid Computing System for Drug Discovery and Design. High Technology Letters, Vol.11, No.4, Dec. 2005.
- [13] SOAP Version 1.2. Published Specification. <http://www.w3.org/TR/soap/>
- [14] Fowler, Martin. UML Distilled: A Brief Guide to the Standard Object Modeling Language, Version 3.0. Addison-Wesley. ISBN 0-321-19368-7.
- [15] Housley, R., W. Ford, W. Polk and D. Solo. Internet X.509 Public Key Infrastructure: Certificate and CRL Profile. <http://tools.ietf.org/html/rfc3280>.
- [16] Goran Petrovic and Peter H. N. de With. Framework for Layered 3D Video Streaming. 27th Symposium on Information Theory. Noordwijk, The Netherlands, 2006.