

# Role-based Access Control for Collaborative Enterprise In Peer-to-Peer Computing Environments

Joon S. Park and Junseok Hwang  
School of Information Studies  
Syracuse University  
Syracuse, NY 13244-4100  
+1 (315) 443-1668, 4473

[\[jspark, jshwang}@syr.edu](mailto:{jspark, jshwang}@syr.edu)

## ABSTRACT

In Peer-to-Peer (P2P) computing environments, each participant (peer) acts as both client and content provider. This satisfies the requirement that resources should be increasingly made available by being published to other users from a user's machine. Compared with services performed by the client-server model, P2P-based services have several advantages. However, wide-scale application of P2P computing is constrained by limitations associated with the especially sophisticated control mechanisms needed between peers. To overcome these limitations, we introduce a controlled P2P computing architecture by extending the concept of Web services to the peer-to-peer level through a generic middleware. Specifically, in this paper we tailor our approach to support RBAC. Although our approach supports both brokered and purist P2P models, all of the policy decisions can be made on the peer side, because policy information is transferred from the policy servers to the corresponding peers through metadata that peers can understand. Each peer makes the access control decision based on the enterprise, the community, and the peer policies without asking other components. This approach supports RBAC services for collaborative enterprise in P2P computing environments, not only within one community but also within inter-communities. Furthermore, it also supports peers' autonomous decisions without causing policy conflicts. The broad dissemination of our approach would enable P2P technology to be applicable to more reliable and efficient services, providing controlled communications between peers.

## Categories and Subject Descriptors

D.4.6 [Operating Systems]: Security and Protection – *access controls, information flow controls.*

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.

SACMAT'03, June 1-4, 2003, Como, Italy  
Copyright 2003 ACM 1-58113-681-1/03/0006...\$5.00.

## General Terms

Management, Security.

## Keywords

Role-based Access Control, Security, Peer-to-Peer Computing.

## 1. INTRODUCTION

Expanded use of the Internet and technological advances are eliciting changes in how individuals manage and share their resources. The Internet has been primarily a client-server-based system, where it is challenging to keep resources up-to-date and consistent among all of the devices. Furthermore, the client-server model has service limitations in heterogeneous computing environments. In recent years, a new Peer-to-Peer (P2P [Clark02, P2P01, P2P02]) model has emerged, perhaps best illustrated by Napster™ [Nap]. In this P2P workplace, each participant (peer) acts as both client and content provider. This satisfies the requirement that resources<sup>1</sup> should be increasingly made available by being published to other users from a user's machine. Compared with services performed by the client-server model, P2P-based services have several advantages. In P2P environments, users can manage heterogeneous resources residing in various platforms and perhaps in different policy environments. A P2P-based resource management model can provide higher resource availability due to the distributed nature of P2P computing. Unlike the client-server model, each peer, as a service provider, can define the resources provided, the service levels, and their conditions. Furthermore, each peer, as a service requestor, selects one of the available service providers (peers) based on the service levels and conditions. That is, there can be multiple peers for the same resources that are proposing different service levels or conditions (e.g., different service fees). Furthermore, P2P-based service provides higher utilization of Internet service resources (e.g., bandwidth) because of its distributed routing architecture.

However, wide-scale application of P2P computing is constrained by limitations associated with the especially sophisticated *control* mechanisms needed between peers; most current peer-to-peer

---

<sup>1</sup> Resources can be contents, services, applications, or computing power. In this paper, we focus on visible resources (e.g., files) in P2P computing environments.

technologies simply focus on sharing services rather than on controls between the peers. One of the key requirements for the control mechanisms is access control. This becomes more critical when peers join or leave a community, which represents a set of peers sharing some resources. In this paper, we introduce an approach for providing the strong and efficient access control mechanism, Role-based Access Control (RBAC [PKF01, PSA01, SCFY96]), to P2P computing environments. Our approach is transparent to peers and supports RBAC mechanisms dynamically based on each peer's current context. In our approach, we use a middleware to provide controlled P2P computing environments by means of machine-understandable metadata based on Extensible Markup Language (XML [BPS00]). In this paper, to clearly illustrate our approach, we use an electronic publication project as an example scenario. However, we believe our approach can be used with other P2P applications that specifically require controlled communications within a community and across inter-communities.

The rest of this paper is organized as follows. In Section 2, we summarize and analyze the existing technologies that are related to our work and describe how we apply those technologies for supporting our requirements. In Section 3, we describe the generic architecture for our controlled P2P computing environments, using a middleware. In Section 4, we discuss how to extend the generic architecture with RBAC, and Section 5 concludes this paper.

## 2. RELATED WORK

### 2.1 Web Services

Web services provide a means for application-to-application communication over the Internet [Shi02]. Since the Internet comprises heterogeneous applications and computing platforms, Web services naturally bring the issues of interoperability and extensibility of those various applications, platforms, and frameworks. Currently, W3C is working on standard architecture and requirement studies and is developing a standard document on this new concept of services on the Internet [WS]. To support such interoperable architecture and processing models, Web services have evolved as a combination of several standard sets of technologies such as XML, WSDL (Web Service Description Language [CCMW01]), UDDI (Universal Description, Discovery and Integration [UDDI]), and SOAP (Simple Object Access Protocol [SOAP]).

The architecture of Web services is based upon the interactions between three primary components: the service provider, service registry, and service requestor. These components interact using publish, find, and bind operations. The service provider is the business that provides access to Web services and publishes the service description in a service registry. The service requestor finds the service description in a service registry and uses that information to bind to the services. Service discovery defines a process for locating service providers and for retrieving service description documents, and it is a key component of the overall Web services model. Service discovery is a very broad concept, which means that it is unlikely to have one solution that addresses all of its requirements. The UDDI (Universal Description, Discovery and Integration) specification addresses a subset of the overall requirements by using a centralized service discovery model. WS-Inspection [BBMNP01] is another discovery

mechanism that addresses a different subset of requirements and uses a distributed usage model. The WS-Inspection specification is designed around an XML-based model for building an aggregation of references to existing Web service descriptions, which are exposed using standard Web server technology. In our research, we extend this Web service concept to the peer-to-peer level (each peer is both service provider and requestor) through a generic middleware.

### 2.2 P2P Models

Actually, the definition of P2P computing is not yet clear. In general, the technology enables a computing environment where peers (users' end-systems) connect to each other without centralized connection points (such as servers in the client-server model). Each peer has resources that it provides to other peers. One of the best-known applications of P2P computing technology is the music file-sharing service, but we can apply the technology to many other applications.

The control of P2P networks can be implemented in various ways. Basically, there are two different models for P2P connection control: "brokered" and "purist."

By the brokered P2P definition, peers communicate through the direct connection for resource sharing, while they use a centralized path for control decisions. In this model, the control path keeps the client-server model, while the data path is on the P2P networks. (This is why some people call the model "hybrid.") One of the best examples of this is the Napster [Nap] file-sharing network. Generic control functions such as search and indexing are performed in the brokered server located in the P2P networks. The status and any changes of user activities are kept and relayed through the network's centralized server. Other P2P networks such as the Groove Network [GN], Kazaa [Kaz] and Blubster [Blu] fall into this brokered type of model.

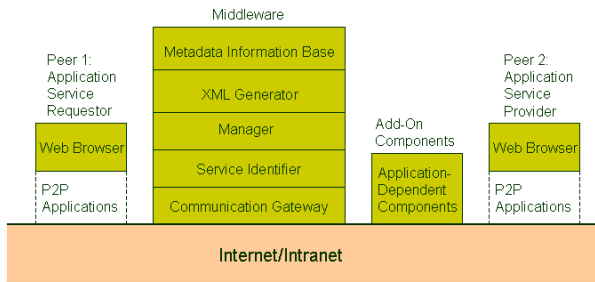
By the purist P2P definition, peers communicate with each other through a direct connection for both resource sharing and control decisions, which allows open architecture of P2P networks. In this pure P2P control, each peer node should have control over its self-organization, routing, and other control functions for managing P2P networks. Gnutella [Gnu, RF02] is a popular P2P network for deploying the control functions in this way. In the Gnutella Network, each node manages the membership and conduct search to form P2P networks. There is no central server to coordinate or relay these functions. Unlike the brokered approach, the peer nodes send and receive queries and responses among them directly to control their P2P networks. For instance, Gnutella peers build an overlay network by forging mesh connections with a set of neighboring peers. The initiation starts with the flooding of queries to their neighbors. Such purist P2P networks include Morpheus [Mor], Limewire [Lim], FreeNet [CHS02, FN], JXTA [JXTA], and Publius [Pub].

In this paper, we introduce an integrated model that supports autonomous decisions and centralized controls. We extend the brokered P2P model with RBAC to support a centralized access control, not only within a community based on the community policy but also inter-communities based on the enterprise policy. Furthermore, we also extend the purist P2P model for peers' autonomous management. Although our approach supports both brokered and purist P2P models, all the policy decisions can be made on the peer side, because policy information is transferred

from the policy servers to the corresponding peers through metadata, which peers can understand.

### 3. GENERIC ARCHITECTURE FOR CONTROLLED P2P COMPUTING ENVIRONMENTS

We design and develop a middleware [HALSM02] platform that works as a broker between peers, providing controlled P2P computing environments. The generic middleware architecture comprises five major overlaid modules as illustrated in Figure 1: Communication Gateway, Service Identifier, Manager, XML Generator, and Metadata Information Base. We call this architecture generic because those five components can be commonly used for any controlled P2P applications. To support application-dependent requirements (RBAC in this paper), we can integrate those components with some add-on components in the architecture. This implies that our middleware approach can be used, not only for a particular application, but also for other applications that require controlled and transparent computing P2P services. The middleware refers to add-on service components that are necessary for providing automatic and transparent processes to peers. In this paper, we add policy servers to the generic architecture to enforce RBAC in P2P computing environments.



**Figure 1. Generic Architecture for Controlled P2P Computing Environments.**

Peers (users) use Web browsers as their primary interface to services. Each peer can communicate with other peers by using P2P applications (software) installed in its machine. The P2P applications can be integrated as plug-ins with the peers' Web browsers, providing transparency to the peers. (The users' interface can be still Web browsers.) This enables more distributed and autonomous P2P services. For instance, if a peer has installed a P2P application that can read and understand the community's security policy, the peer can make policy decisions autonomously. The security policy is originally defined in a policy server and transferred to the peers (usually providers) in metadata (XML) form via the middleware. Without the corresponding P2P application in the peer's machine, the peer needs to ask for decisions from an external decision maker. In our description below, we focus on the RBAC (Role-based Access Control) policy, assuming that peers have installed the corresponding P2P applications in their machines.

The left side of Figure 1 contains a peer (requestor) who requests resources, and the right side contains a peer (provider) who provides the response for a specific request. All the communications between a peer and the middleware are through the Communication Gateway, which supports various communication protocols such as HTTP and SOAP. The Communication Gateway provides a window to the Service Identifier for receiving service requests from peers. The middleware is designed to support various types of service transactions; therefore, the identification of service types needs to be embedded in the middleware. The Service Identifier contains WSDL (Web Service Description Language) documents, which offer a description of the types of services managed by the middleware. Information concerning port types, service types (e.g., Web services, Open Grid Services Architecture services (OGSA [FKNT02]), Context Aware services, etc.), and message types is contained inside a WSDL document. In this paper, typically, the resource transaction services for P2P content management will be identified in the form of Web Services.

The Manager facilitates provisioning capabilities for the use of resources and also controls usage of resources on behalf of a peer. Alternatively, especially with our approach in this paper, we can distribute those decisions to peers by using corresponding P2P applications. In this case, policy decisions can be made autonomously by the peers. The Manager elements allocate and manage resource-sharing among different peer systems (communities) and networks with different resource definitions and identifiers. A Manager provides a mechanism for managing resources within its domain, where it has direct control. A domain consists of one or more communities. It also provides a mechanism to manage resources with other Managers' domains. A Manager supports resource allocation and provisioning at the system boundary among multiple peers.

If a peer, as a provider, wants to provide its resource to other peers (say, Alice wants to share her chapter with editors in our example), the peer first needs to access the middleware through the p2p applications in order to generate metadata about the resource. In the middleware, the metadata is automatically generated by the XML Generator and stored in the Metadata Information Base. The XML Generator helps to establish machine-understandable metadata for existing resources (including files, peers, security policies, ontologies, charging and account information, and other constraints) to be retrieved and managed. Ideally, the XML Generator converts any existing non-XML data to XML standards. To provide autonomous services to peers, resource descriptions and policies that can be read and understood by the corresponding P2P software, is indispensable. To generate the metadata automatically, a peer needs to provide some support information about its resources (e.g., author's name, keywords, date, category, data format, version, etc.) that will be used for resource search by other peers. Each peer may need to provide its background information to generate metadata for its profile. Furthermore, the XML generator creates metadata about policies to the resources, referring to external components such as policy servers. If the metadata provides sufficient information in a machine-understandable language, the peers will be able to make a decision autonomously, based on the information in the metadata (role-based access control in this paper).

The Metadata Information Base is a distributed database that maintains metadata information for the resources. The application service requestor (Peer 1 in Figure 1) can search other peers' resources, based on the metadata stored in the Metadata Information Base. The same resources may be located in many different peers. Peer 1 can see a list of peers and conditions for the resource it is looking for. After Peer 1 selects one of those peers (say, Peer 2) in the list, the middleware between peers looks up the policy servers and generates policy metadata that are related to the request. Then, the middleware sends the metadata to the corresponding peers (usually, the provider, Peer 2 in our example). Finally, when Peer 1 connects to Peer 2 and requests services (for instance, file download), Peer 2 can make a decision using the corresponding P2P application that understands the policy metadata.

## 4. RBAC-ENABLED P2P COMPUTING ENVIRONMENTS

### 4.1 Access Control Requirements

Let us imagine a system that supports individuals (peers) with appropriate services in a virtual common workplace. We consider this common workplace as a community in a P2P computing environment. If a P2P application supports many users as peers, we should consider more efficient and secure access control mechanisms for both inside and outside peers. This becomes more serious when multiple communities are involved in the same collaborative enterprise. Some of the peers may have common functionality in one community while each peer may be involved in multiple communities. A peer's privileges need to be changed based on its context for the enterprise. Peers do not need privileges before they join or after they leave the corresponding community. There can be more complex cases for providing minimum privileges to the right peers at the right time. Therefore, we need an efficient mechanism to control "Which peer has which privileges for which resources under what conditions?" Technically, we could use the conventional identity-based access control mechanism for this purpose. This could work for a small project, where a small number of peers are involved and the job assignment is stable. However, for a large system that supports many peers from different organizations, the identity-based access control mechanism is inefficient and too complicated to manage, because the direct mapping between peers and privileges is transitory.

To solve this problem, we should separate the mapping between peers and privileges through common job functionalities such as roles. Usually, functionality-privilege mapping is more stable (of course, it can be changed when necessary) than peer-functionality mapping, because job responsibilities for a collaborative project do not change frequently, while peers' job functions change quite often. The system makes access control decisions based on the peers' job functions instead of on their identities. This provides an efficient access control mechanism to the system and resolves the scalability problem. To support RBAC among different communities in P2P environments, we need to provide role ontology [Fen01] for such a collaborative enterprise. For instance, there should be a way to have decision makers understand that the Authors role in Community A and Editors role in Community B can refer to the same job function and therefore require the same privileges.

### 4.2 RBAC Policies Applied to Peers

In this subsection, we tailor the generic architecture for controlled P2P Computing Environments (described in Section 3) to support RBAC. To enforce RBAC for enterprise projects in P2P computing environments, we add application-specific service components such as enterprise and community policy servers to our generic architecture described in Section 3.

For brevity, we describe our approach based on a collaborative publishing enterprise in P2P computing environments where multiple communities are involved. If we consider the job functionalities of peers as roles, we can use those roles for access control decisions. Suppose Community A uses the Authors, Editors, and Illustrators roles and Community B uses the Writers, Reviewers, and Distributors roles. Each peer needs different roles in different communities. A peer can be assigned to multiple roles in the same community in other applications. The same role may be associated with different privileges in different communities. Analogously, the role of Manager in one company probably has different privileges than in another company. Conversely, different roles in different communities may have the same permissions. For instance, in our example, the Authors role in Community A may have the same permissions as the Writers role in Community B. This is supported by the concept of role ontology in our work.

For a collaborative enterprise in P2P computing environments, we basically identify three different policies: enterprise, community, and peer policies. A community policy defines the community's URA (User-Role Assignment), PRA (Permission-Role Assignment), constraints, and role-hierarchy. Since we have two different communities in our example, we need two different community policy sets, one for each community. An enterprise policy defines the enterprise's URA, PRA, constraints, role-hierarchy, and role ontology that apply to all the participating communities and peers. Each peer defines its own peer policy, including the peer's PRA and constraints. Figure 2 shows the three RBAC policies applied to a peer, Chris, in our example.

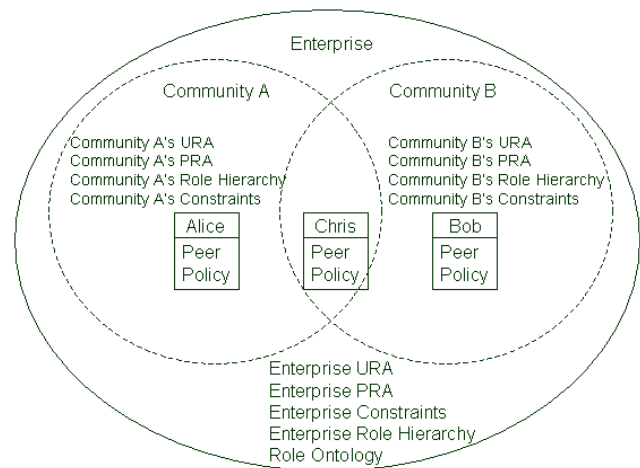


Figure 2. RBAC Policies Applied to Peers.

An enterprise policy is enforced by a centralized mechanism that applies to all the participating communities (including the peers in the communities), while a community policy is enforced by a centralized mechanism within the community (supports community-level autonomy). These two mechanisms are based on the brokered P2P model (described in Section 2). To support peer-level autonomy, we allow each peer to define its peer policy in its machine through the corresponding P2P software. For instance, a peer, Chris, can define his own PRA and constraints, which should not conflict with his current community or global policies. This mechanism is based on the purist P2P model (described in Section 2). If a conflict occurs between policies, enterprise policies (EP) are superior to community policies (CP), which are superior to peer policies (PP) unless a specific policy priority is defined.

The URA defines which peer is assigned to which roles in which communities. Suppose Alice wants to access Bob's article, but does not know initially which peer has the article. The following example shows how we can federate the enterprise, community, and peer policies to support RBAC for a collaborative work in P2P environments. Note that this example does not give a formal representation of a comprehensive set of policies. The purpose of this simple example is to illustrate the relationships between the policies and to show how we enforce them on the peer side (service provider in this case).

#### [Enterprise Policy]

Enterprise URA:

E\_URA1. Alice is assigned to the Enterprise Member role.

E\_URA2. Chris is assigned to the Enterprise Auditor role.

Enterprise PRA:

E\_PRA1. The Enterprise Member role can join any communities that are participating in the enterprise.

E\_PRA2. The Enterprise Member role can search resources in any communities that are participating in the enterprise.

E\_PRA3. The Enterprise Auditor role can access any resources in any participating communities in the enterprise.

Enterprise Role Hierarchy:

E\_RH1. Enterprise Auditor > Enterprise Associate > Enterprise Member

Enterprise Constraints:

E\_Constraint1. A peer cannot join Community A and C at the same time.

Role Ontology:

RO1. The Editors role in Community A and the Reviewers role in Community B are interchangeable.

:

#### [Community A's Policy]

Community A's URA:

CA\_URA1. Alice is assigned to the Editors role.

Community A's Role Hierarchy:

CA\_RH1. Editors > Authors

:

#### [Community B's Policy]

Community B's PRA

CB\_PRA1. The Reviewers role can access Bob's article.

:

#### [Bob's Peer Policy]

Bob's PRA:

Bob\_PRA1. The Editor role can access Bob's contact information.

Bob's Constraints:

Bob\_Constraint1. Do not keep personal files in the shared directory.

:

The operational scenario (Alice wants to access Bob's article, but does not know initially which peer has the article), based on the above policies, is as follows. Before Alice starts this operation, we assume that all the related resource description metadata were generated by the XML Generator and stored in the Metadata Information Base, as we described in Section 3.

1. Alice can join a community (either Community A or B), since she has the Enterprise Member role in the enterprise (E\_URA1 and E\_PRA1). Suppose Alice joins Community A. The middleware authenticates<sup>2</sup> Alice and checks if she is assigned to the Enterprise Member role.
2. Alice can search Bob's article through the resource description metadata stored in the Metadata Information Base. The middleware allows this operation because Alice has the role required (Enterprise Member) to use its search service (E\_URA1 and E\_PRA2).
3. Alice reads a list of peers who have the article she is looking for and decides which peer she is going to connect to. Suppose she selects Bob in Community B as a resource provider in the list.

---

<sup>2</sup> The authentication mechanism is not described in this paper, since it can be supported by any existing authentication technologies in different ways.

4. The middleware pulls Alice's role information in Community B, referring to the policy servers and sends it to Alice in XML format. Since Alice is assigned to the Editors role in Community A (CA\_URA1), which is interchangeable with the Reviewers role (RO1) in Community B, she is also assigned to the Reviewers role in Community B.
5. The middleware looks up the corresponding policy servers and generates policy metadata that is applied to Bob's article (e.g., CB\_PRA1). The middleware sends the policy metadata to the provider (Bob).
6. Alice connects to Bob, presenting her role in Bob's community (Community B) described in XML. When Alice requests the article from Bob, Bob checks if Alice has the required role to access the article in his machine by reading the policy metadata that he received in step 5 and the role information presented by Alice. Since Alice has the required role, Reviewers, Bob allows Alice to access the article (by E\_URA1, RO1, and CB\_PRA1).
7. As Bob defined in his Peer Policy, he gives his contact information to Alice (Bob\_PRA1) as well.

The above operational scenario is just an example of many possible cases. Detailed procedures may vary based on applications and implementations. We introduced the user-pull mechanism to retrieve the peer's role information in this example, however the server-pull mechanism is also possible [PSA01]. Basically, each peer (Bob in the above example) makes the access control decision based on the enterprise, community, and peer policies, without asking other components. This approach supports a centralized RBAC not only within a community but also inter-communities. Furthermore, it also supports peers' autonomous decisions without causing policy conflicts.

## 5. IMPLEMENTATION STRATEGY

In our approach, the community and enterprise URAs are maintained in a centralized manner, while the PRAs are maintained in a distributed manner. This brings several administrative benefits. First, it provides efficient maintenance, because all the URA mappings for the participating peers in the enterprise are in the same place. URA needs to be frequently changed in real applications so centralized URA maintenance provides effective real-time update and synchronization. Second, it supports easy coordination between URAs, such as inter-community constraints, for the enterprise. For instance, if we need to enforce separation of duty by means of the peers' roles in different communities, we can easily deal with the URAs in the same place. Third, it supports a better performance (fast and accurate search) in finding a peer's URA information. For instance, when we find all the roles assigned to a peer in the enterprise, it is faster and more accurate to search one place that has all the information, rather than to look for multiple distributed places. In real P2P environments, a URA search across communities is frequently requested. Furthermore, it is reasonable to distribute community PRAs to their corresponding community policy servers, because PRAs are usually stable and do not require inter-community interdependencies as URAs do. Usually, we do not need to find all the permissions associated with a role in the enterprise. One could claim that the separation-of-duty support is

difficult through distributed PRAs, but the same effect can be achieved through centralized URAs. Therefore, we believe it is a good strategy to centralize URAs while distributing PRAs to corresponding places in P2P computing environments in which many communities participate.

Currently, we are implementing our idea in JAVA; the middleware with the five major overlaid modules (Communication Gateway, Service Identifier, Manager, XML Generator, and Metadata Information Base), P2P software for peers, and policy servers. In our implementation, a peer can communicate with other peers using a Web browser. Metadata is generated by the XML Generator and stored in the Metadata Information Base. The control metadata is transferred from the Metadata Information Base to a peer's P2P software, where policy decisions are made, via SOAP (Simple Object Access Protocol). The P2P applications can be installed as plug-ins in the peer's Web browser, providing transparency to the user.

## 6. CONCLUSIONS

In this paper, we have introduced a controlled P2P computing architecture by extending the concept of Web services to the peer-to-peer level, using a middleware. Particularly, we have tailored our approach to support RBAC. We have also introduced implementation alternatives addressing optimum strategies. Our approach enables a peer to make the access control decision autonomously based on the enterprise, the community, and the peer policies without asking other components. The broad dissemination of our approach would enable P2P technology to be applicable to more reliable and efficient services, providing controlled communications between peers. Finally, although we have applied our approach to visible resources (e.g., files) in P2P computing environments in this paper, in our future work we will apply our idea to invisible resources, such as computing powers, applications, or services that peers can provide to other peers.

## 7. ACKNOWLEDGMENTS

Our thanks to the support from the New York State Center for Advanced Technology in Computer Applications and Software Engineering (CASE) and Systems Assurance Institute (SAI) at Syracuse University.

## 8. REFERENCES

- [BBMNP01] K. Ballinger, P. Brittenham, A. Malhotra, W. Nagy, S. Pharies. *Web Services Inspection Language (WS-Inspection) 1.0*, October 2001.
- [Blu] *Blubster*, <http://www.blubster.com/>.
- [BPS00] T. Bay, J. Paoli, and C. Sperberg-McQueen. *Extensible Markup Language (XML) 1.0*, W3C Recommendation. October 2000; <http://www.w3.org/TR/REC-xml>.
- [CCMW01] Christensen, E., Curbera, F., Meredith, G. and Weerawarana., S. *Web Services Description Language (WSDL) 1.1*. W3C, Note 15, 2001, [www.w3.org/TR/wsd1](http://www.w3.org/TR/wsd1).
- [CHS02] I. Clarke, S. Miller, T. Hong, O. Sandberg, and B. Wiley. *Protecting free expression online with Freenet*. IEEE Internet Computing , 6(1), January/February, 2002 , pp. 40–49.
- [Clark02] D. Clark. *Face-to-Face with Peer-to-Peer Networking*. IEEE COMPUTER, 34(1), January 2001, pp. 18-21.

- [Fen01] Dieter Fensel. *Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce*. Springer Verlag, ISBN: 3540416021, August 2001.
- [FKNT02] Foster I., Kesselman, C., Nick, J. and Tuecke, S. (2002). *The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration*. 4th Global Grid Forum, Toronto, Canada.
- [FN] *FeeNet*, <http://www.freenet.com>.
- [Gnu] *Gnutella*, <http://www.gnutella.com>.
- [GN] *Groove Network*, <http://www.groove.net/>.
- [HALSM02] Junseok Hwang, Praveen Aravamudham, Elizabeth Liddy, Jeffery Stanton, Ian MacInnes. *IRTL (Information Resource Transaction Layer) Middleware Design for P2P and Open GRID Services*. 36th HICSS, Big Island of Hawaii, USA.
- [JXTA] *Project JXTA*, <http://www.jxta.org/>.
- [Kaz] *Kazaa*, <http://www.kazaa.com>.
- [Lim] *Limewire*, <http://www.Limewire.com/>.
- [LS99] O. Lassila and R. Swick. *Resource Description Framework (RDF) Model and Syntax Specification*. W3C Recommendation, February 1999; <http://www.w3.org/TR/REC-rdf-syntax/>.
- [Mor] *Morpheus*, <http://www.Morpheus.com/>.
- [Nap] *Napster*, <http://www.napster.com>.
- [P2P01] *Proceedings of the First International Conference on Peer-to-Peer Computing (P2P'01)*. Lingköping, Sweden, August 27 - 29, 2001.
- [P2P02] *Proceedings of the Second International Conference on Peer-to-Peer Computing (P2P'02)*. Lingköping, Sweden, September 05 - 07, 2002.
- [PICS] Platform for Internet Content Selection (PICS), <http://www.w3.org/PICS/>.
- [PKF01] Joon S. Park, Myong H. Kang, and Judith N. Froscher. *A Secure Workflow System for Dynamic Cooperation*. IFIP 16th International Conference on Information Security (IFIP/SEC 2001), Paris, France, June 11-13, 2001.
- [PSA01] Joon S. Park, Ravi Sandhu, and Gail-Joon Ahn. *Role-based Access Control on the Web*. ACM Transactions on Information and System Security (TISSEC), 4(1), February 2001.
- [Pub] *Publius*, <http://www.publius.com>.
- [RF02] M. Ripeanu, A. Iamnitchi, and I. Foster. *Mapping the Gnutella Network*. IEEE Internet Computing, 6(1), January/February 2002, Pages: 50–57.
- [SCFY96] R. Sandhu, E.J. Coyne, H.L. Feinstein, and C.E. Youman. *Role Based Access Control Models*. IEEE Computer 29 (2), February 1996.
- [Shi02] C. Shirky. *Web services and context horizons*. IEEE Computer, 35(9), September 2002, pp. 98–100.
- [SOAP] SOAP: *W3C Simple Object Access Protocol Version 1.1*, May 8, 2000 <http://www.w3.org/TR/SOAP/>
- [UDDI] *Universal Description, Discovery and Integration*, <http://www.uddi.org/>.
- [WS] *Web Services Architecture*, <http://www.w3.org/TR/2002/WD-ws-arch-20021114>.