Collaborative and Secure Resource Management with Distributed Agents

Yugyung Lee, Younki Cho, Jerry Stach, E. K. Park
Computer Science Telecommunications
University of Missouri-Kansas City
5100 Rockhill Road, Kansas City MO, 64110
{yugi, ycho, stach, ekpark}@cstp.umkc.edu

Abstract

As an increasing number of people communicate and collaborate computationally over the Internet, the need for a collaborative environment that facilitates secure and reliable access to large quantities of distributed resources has become manifest. However, the breadth of resource distribution makes it difficult for Internet users to acquire the necessary resources for collaborative computing. We have developed a framework for an efficient infrastructure that provides a reasonable mapping between users and resources distributed over the Internet. The proposed framework contributes: (1) an authority mechanism that can be applied to independent and widely distributed resource-providers; (2) a resource hierarchy for partitioning global resource spaces into manageable subspaces; and (3) systematic management of dynamic groups of users and resources running on different sites. This paper describes the schema and processes for resource partition, mapping, negotiation, and forming a resource domain in our framework.

1 Introduction

As the Internet facilitates communication among people, they are willing to collaborate through the Internet to achieve their common goals. The current trend of collaborative computing is toward an increased use of resources from distributed sites within a shared computing environment. Collaborative and distributed computation (CDC) may involve many processes and large amounts of data, geographically distributed over multi-organization and networks.

However, the breadth of resource distribution makes it difficult for Internet users to acquire the necessary resources for collaborative computing. Collaborative users require resources to be categorized based upon their individual interests. Furthermore, due to the relatively insecure Internet infrastructure, there is a high risk of exposing valuable and sensitive resources to unauthorized users. In order to build a collaborative environment, which provides fast access for large quantities of distributed data and a reliable use of computational resources, some issues should be addressed. Due to the complexity of the distributed resources spread over the Internet, it is not easy to attain required resources whenever they are needed. Also, due to the diversity of the Internet users’ needs, it is difficult to acquire the resources corresponding to their interests. The collaborative and distributed computing service scheme requires distributed systems and applications to be integrated not only within a community internally but also externally with strategic partners or other communities. Due to the insecure Internet infrastructure, there is a risk of valuable and sensitive resources being exposed to unauthorized users.

The goal of this study is to design a framework for a CDC system providing a reasonable mapping between resources and users in a secure and efficient way. In this paper, we specifically describe resource mapping schema required for building such framework. The schema includes construction of resource and user graphs, partition of the graph into resource domains/user domains, and dynamic mapping between the user domains and the resource domains. These schema includes challenging problems such as partition and mapping known as NP-hard problems [6]. Our resource and user partitions are designed based on building resource/collaborate graphs and constraint-based partition method. Our resource mapping schema are designed based on Economic and Sociological models supporting the processes of negotiation, agreement, contract and group forming and management. This paper is organized as follows: Section 2 describes our motivation. Section 3 introduces the architecture of our CDC system. Section 4 focuses on the detailed schema to build the resource domains. Section 5 contains our conclusions.

2 Motivation

2.1 Distributed Resource Management
The Internet resources are huge in size and complex in structure. Understanding the relationships between resources and also identifying the requirements for their use are very difficult. There might be different types of resources including abstract resources (i.e., CPU and scheduling disciplines) and specific resources (i.e., data, instrument, storage capacity, communication channel). Each resource must have a set of requirements for their use. There may be different granularities of resource hierarchy and diverse requirements. Thus, a mapping several different resource into diverse users is a challenging task.

Recently, software agent frameworks have gained much attention as counterparts of static distributed object frameworks. Many researches have been carried out to apply agent technologies to the areas of e-commerce, information retrieval, interactive interface, education, resource management, etc. Among many agent definitions, the essential properties were reported by [22]: Autonomy, Learning, Mobility, Persistence/Flexibility, Active/continuity, Pro-activeness, and Social ability (communication, collaborative).

The mobile agent paradigm has several distinctive advantages. Generally, mobile agents are being contemplated as an alternative to traditional client-server computing based on the matter of fact of their social ability, intelligent decision making capabilities, and their concurrent execution and mobility capabilities. The mobile agents have the capability to reduce network use, increase asynchrony between clients and servers, and introduce concurrency. In client-server applications, the servers usually provide a public interface with a fixed set of primitives. The clients might need and change their high-level functions that compose of these primitives over a certain period of time. A client can maintain its own interface at the server side by using the mobile agent rather than the server changing its interface frequently to support the requirement of each client. Because the mobile agents also can execute concurrently, this feature allows parallel activities. A client can distribute its task among multiple agents to provide parallel computing or fault tolerance.

Many researches have been carried out on agent-based network management and control environments. Wehmayer et al. pointed out that without taking on the potential advantages of software agent, rapid service deployment, logical configuration flexibility and management scalability will not be achieved in the future network [21]. The agent-based performance management system [2] exhibits constrained mobility and the cooperative management of TCP/IP-based networks [7]. A tightly coupled, distributed, co-operative, multi-agent system provided a solution for the network traffic control and dynamic reconfiguration [21]. Multiple, simple, homogeneous, software process (referred as Ants), are allowed to randomly roam the physical network configuration [8]. The agents are working for congestion control in a circuit switched environment distributed network control, where fixed co-operative message passing control entities are used, leads to complex co-orientation strategies when viewed as a single system.

Somers [17] developed a multiple agent-based system, called HYBRID, for resource management and system reconfiguration in a distributed manner. In particular, they used a hierarchy of autonomous agents who have local decision-making capabilities, but co-operate to ensure system-wide objectives are maintained. The hierarchical structure provides allowing high-level agents to inherit their capabilities through the hierarchy and co-operating between peer agents. The HYBRID system consists of a three-layer agent architecture, responsible for a particular region of network and independent agents operate within the layer of the architecture. Our system design was inspired by the HYBRID in terms of their three main features: control distribution, hierarchy and autonomous management.

![Figure 1. Resource Agent Hierarchy](image)

Our approach attempts to partition the resource spaces into manageable subspaces and organize the partitions into a hierarchical structure which can employ distributed resource management. We propose two primary structures, (1) a tree structure for automated brokering and management of distributed resources, (2) a ring structure for agent negotiation, which will support efficient coordination of shared resources of the tree (See Figure 1). For enabling communities to share valuable resources in a secure way, we use a mobile agent to negotiate and represent the diverse requirements of services and users. An agent is specialized in representing either a set of related resources or a user, the former is called Provider Agent (PA) who knows the primary functionality of resources', their usage requirements and access policies, and related resources and required computing environments. The latter is called Consumer Agent (CA) that delegates the users; requirements and privileges and the tasks to be accomplished in the CDC environment, or the partners or collaborators. Groups are managed under
an inheritance-based hierarchy from which memberships are generated and distributed according to ring or tree structures. The hierarchical management implies the control process and communication structure of distributed computing. Our framework is robust in that group membership is dynamic during collaboration and is extensible.

2.2 Policy based Management

The distributed and collaborative computing often requires the resources from multiple computing environments resided in different physical locations. The Internet becomes a virtual repository of resources managed by organizations that have different management paradigm and policies. Due to the diversity of distributed resources, underlying network and their providers/users, the Internet is heterogeneous in its nature. As a matter of fact, Internet users may have different criteria for accomplishing their task: for instance, some users may want to perform it through collaboration and sharing resources with others while others do in an exclusively isolated environment. Some users may have an authority to access highly confidential resources while others are not. Similarly, the Internet providers demand different compliance for the usage of their resources. Compared to the centralized systems, distributed management systems are more complicated to construct, operate, and maintain.

In this paper, we address some important issues including how to represent diverse requirements of users and providers and also how to specify a resource management infrastructure for a natural mapping between providers and users. We adapt a policy-based group management mechanism which can support an unified representation of the requirements and policies [3]. The natural mapping is established through negotiations and agreements on policies among the participant agents, and forming a resource sharing group. Then, the resource management is conducted through the group. Mediator agents (SRA, Section 3) are introduced for efficient interactions among the agents during negotiation and resource sharing and a resource hierarchy among the agents is introduced for effective management and resource partition. The hierarchy becomes a framework for supporting interoperability between the various resources and users by a better understanding of related requirements and policies. Imposed over the resource hierarchy, a well-defined scope and enforcement policy of resource management enables distributed and unified requirements management for a secure and collaborative environment.

2.3 Secure Computing Environment

A distributed and collaborative computing environment has been built on top of Internet. The Internet is not secure. The growth of the Internet raises many concerns on network security and also focuses on development of open and flexible security architectures for ensured communication privacy, authentication, and authorization. Since these mechanisms are often obscurely related, there has been some interest in the ability to express and qualify the level of support for security and quality of its service. A secure, collaborative, distributed framework must specify: how to determine the security level and access level; how to describe the security policies to the resource providers and users; how to construct a secure negotiation between the resource providers and users.

The resources in the CDC are represented as a collection of separate and largely unrelated domains and the policies specify the privileges and permissions necessary to perform the assigned duties and the requirements to use the resources for the duties. It is not easy to identify the relationships between resources, between users, and between resources and users. The design of our resource hierarchy based on policies is followed by Harroud et al. [9]. They designed a hierarchy of policy-based classes, in which classes are used to represent specific policies and the policy inheritance is defined between the general and specific classes. Second, the security issue in our CDC is based on Certificate management and Revocation list control [18]. In their approach, shared security keys, group keys, are used to secure group-based communication, to exchange authorization credentials which specifies permission or restrictions (policies) of usage of resources, and to be updated according to frequent membership changes. Traditional security systems are designed to enforce one particular security policy. Under the object-oriented policy in Java, policies are tailored to their specific operational needs. The major concerns of authorization systems are (1) who can access what operations on which object, and (2) how this can be achieved. The traditional access control uses a set of an access control list, which defines what type of operations can be performed on the resources. The access control list (ACL) specifies in a list <resource (object), user (subject), permissions (action)> who is user (subject), what are resources (object), what kind of privilege for the object can be obtained by the subject. For instance, Discretionary Access Control (DAC) allows privileges to be granted to other subjects by the object owners and Mandatory Access Control (MAC) with a well-defined security policy [1]. In reality, it is not easy to enumerate all the objects and describe in a simple access control list. Furthermore, the resources and the users in a CDC environment are diverse in terms of their granularities, resource types, and requirements.

It is necessary to review some security techniques developed by agent community (Ajanta [10], Agent-Tcl [15], Ajanta [10], Voyager [12], etc). The security model in Agent-Tcl [15] separates untrusted code from trusted code, with clear and simple boundaries between environments.
having different security properties and provides mechanisms for implementing a variety of security policies. Ajanta [10] extended RMI Security model to use an access control list (ACL) to network resources. The Ajanta uses proxy server-based resource access control at application-level. As a code verification approach, Java runtime environment, which includes a “byte-code verifier” to ensure the validity of Java-based mobile agent. Using timeout concept, a mobile agent has been roaming the network past its relevant life span can be destroyed, or returned to its origin as the Ajanta system. Voyager [12] provides secure network communication over a protocol based on an encrypted and authenticated channel.

Our CDC system supports a group-based authentication and access control. A secure group is formed through a policy-oriented negotiation between resource consumers and providers according to their access level and requirements. During the negotiation, a collective and unified view on security policies is constructed and can also be dynamically updated according to changes on the requirements of resources and participants. The integration of security and quality of service into a unified system is a challenging task because it employs distributed and heterogeneous resource management. This results in significant reduction on security overheads. Our approach can be differentiated from others such that security and collaboration are established by formation of resource sharing groups based on natural interactions such as policy-based negotiation and agreement.

3 Resource Agent Hierarchy

The primary goal of this study is to build a virtual and logical environment where consumer and provider agents can share their valuable resources and collaborate each other for their common goals. A resource agent hierarchy will be built to organize and manage the partitioned resource domains and provide a framework mapping between diverse application resources and specific user requirements. The resource agent hierarchy has two primary structures: (1) a tree structure for automated resource mapping and management of distributed resources, and (2) a ring structure for agent negotiation, which will support efficient coordination of shared resources of the tree.

Our architecture is a three-tiered hierarchy (Figure 1). The authority of management is distributed over the tree: The root authority in the hierarchy is an ultimate owner of all the resources, and delegates direct management of some of its resources to lower-levels. The first tier is composed of a general secure resource agent (GSRA) that has a general view of resources and policies. The second tier is composed of a regional secure resource agents (RSRAs) that has an intermediate view of resources and policies. The bottom tier has resource domains composed of a secure resource agents (SRA), provider agents (PAs) and consumer agents (CAs).

These agents (GSRA, RSRA, and SRA) are involved in the management of resource domain and verification of the agents. They have privileges of making decisions either to grant or deny the agents’ requests. Agents in the higher-level (e.g., GSRA) provides manages more general services/resources so that their policies are more general than those in the lower level. The GSRA controls the hierarchical communication protocol and provides directory services, which are based on a certificate repository for the verification of resource domain entry, authority, and resource policies. The GSRA has a collective view of resource regions such as the specific policies, available resource information, and the relationship between regions. The RSRA and SRA have authorities similar to GSRA’s, but they have smaller management scopes and lower abstraction of resource policies and requirements than the GSRA. The RSARs and SRAs manage specific resources and access policies of resource regions and resource domains, respectively. The GSRA inherits the attributes and policies to the lower level RSARs and SRAs. The inheritance hierarchy simplifies the communication and access control for the resource sharing in the collaborative and distributed environment. The agents in the bottom tier are mobile agents representing diverse users and heterogeneous resources. Mobile agents are adequate for designing of effective communication and coordination mechanisms to support inter-agent negotiation among available resources and services. Through the negotiation and agreement among agents, a resource domain is constructed with a SRA and resource agents. The SRA is a coordinator which initiates the negotiation, supports for an agreement among participated agents, forms a resource domain and manages the resource domain.

The group-based access control for agent coordination is available through a framework constructed on top of a combined structure of a tree-based and ring-based topology. Our distributed management based on resource agent hierarchy delivers important security services and enhances an efficient coordination between sharable resource members. Through a well-defined scope and enforcement policy of resource management, we have achieved decentralized and secure resource management for a large distributed environment.

4 Resource Mapping Using Distributed Multi-Agents

Our unified mechanism maps the diverse requirements of users with heterogeneous policies of resources using three important features: (1) agents represent diverse policies from both sides (consumers and providers), (2) policy-based natural interaction (negotiation) to achieve an agree-
ment on policies, and (3) an organizational structure (sharing group based on the agreement) for simplifying communication and maximizing utilities.

A three-step approach is used to construct a resource domain (the bottom layer of the resource agent hierarchy (Figure 1)). The first step is to represent the resource space/the user space in a resource graph/user graph, which is an acyclic graphs composed of a set of resources/a set of users and a set of relationships between resources/users. The second step is to partition the resource graph/user graph into manageable subspaces and delegate an agent, called a provider agent/consumer agent, to manage a subspace. A resource agent type, called a provider agent, is specialized in representing a set of related resources. The provider agent knows the primary functionality of resource, its usage requirements and access policies, and related resources and computing environments. Another agent type, called a consumer agent, is specialized to represent a user or a group of users. The consumer agent knows the requirements and privileges of the users, the tasks to accomplish, and the collaborators of the tasks. The goal of the partition is to minimize the communication between the provider/consumer agents and to maximize the resource utility within the provider/consumer agent. The third step is to construct a collaborative resource space, called a resource domain. The resource domain is a virtual framework which provides for independent and widely distributed provider agents to assert their authority over diverse consumer agents, and in which they collaborate with each other for their common goals, resource sharing.

First, we describe the detailed schema and processes to support an efficient mapping between requirements of resource consumers and resource providers in our framework (Sections 4.1 and 4.2). Second, we discuss the processes of reaching to an agreement through the negotiation on acquiring required resources and forming a resource sharing group (Section 4.3). In our model, the initiation of control is from an agreement between consumer agents (CAs) and provider agents (PAs). We are particularly interested in how to optimize the negotiation process (with minimum negotiation process and an optimal mapping between CAs and PAs) and what kind of communication and coordination mechanism among the agents are required. The details of the proposed architecture including agent specifications, group-based security and management, and communication protocol, are available in [3].

4.1 Representing Resources using Provider Agents

The resources in the Internet are heterogeneous and diverse. Their requirements and accessing policies are also not clearly defined so that acquiring the Internet resources tends to be complex and time consuming processes. Thus, we need focus on how to represent the requirements of the resources for their safe and fast access. For their efficient use, it is necessary to define the property and requirement of resource types and the relationships between resources. In this paper, the resources are restricted to application programs, their required information and data distributed through Internet. We can construct a resource graph based on the identified resource types and the relationships. The resource type is determined by a set of attributes associated with resource. The attributes, that determine the resource type, are Goal (G), Cost (C), Security level (SL), Distribution mode (DM), and Priority (P). The resource relationship determines how resources are related through specific relationships including Excluded (Ex), Required (Rq), Compatible (Ct), and Shared (Sr). Among those relationships, Excluded is an example of negative relationships while Required (Rq), Compatible (Ct), and Shared (Sr) are examples of positive relationships. The negative relationship type is represented by a negative number and the positive relationship types are represented by a positive number and prioritized according to the degree of its tendency: \( Rq > Sr > Ct \).

Formally, the resource graph is represented by directed acyclic graphs (DAGs), i.e., \( G = (V,E) \) where \( V \) is a set of vertices representing a resource type, \( RT = \{Hard, Soft, Positive, Negative\} \), determined by a set of attributes, \( RA = \{G, C, SL, DM, P\} \), and \( E \) is the set of edges representing a resource relationship between resource types, \( RR = \{Ex, Rq, Ct, Sr\} \). According to the attributes and relationships, three types of resources can be determined as follows:

- Required resource (Rr) =
  \[ \forall i,j \in R \]
  \[ Cond_R ( \sum^n Rq(i,j) > \sum^n Sr(i,j) > \sum^n Ct(i,j) \land \neg Ex(i,j)) \]
  \[ Cond_E (G_i = G_j, C, SL, P, DM) \]

- Shareable resource (Sr) =
  \[ \forall i,j \in R \]
  \[ Cond_R ( \sum^n Sr(i,j) > \sum^n Rq(i,j) > \sum^n Ct(i,j) \land \neg Ex(i,j)) \]
  \[ Cond_E (G_i = G_j, C, SL, P, DM) \]

- Exclusive resource (Er) =
  \[ \forall i,j \in R \]
  \[ Cond_R (Ex(i,j) \land \neg Rq(i,j) \land \neg Ct(i,j) \land \neg Sr(i,j)) \]
  \[ Cond_E (G_i = G_j, C, SL, P, DM) \]

The constraints are defined by the resource attributes and resource relationships. The strength of constraints are decided by the property of resources, their comparability and requirements with other resource types. Depending on the
strength of constraints, two types of constraints are determined: soft constraints and hard constraints. In a model we proposed, Required resource and Exclusive resource are hard constraints and Sharable resource is a soft constraint. The type of constraint is used as a criterion of the graph partition which will be discussed later. For the further simplification of a resource space, we partition the resource graph into several sub-graphs. Some issues should be considered for this process: how to partition the resource graph in the way of reducing the constraint conflict and minimizing communication cost between sub-graphs.

Our partition schema considers not only the quantity of partitioned graphs (such as the number of nodes or edges) but also the quality of graphs (such as comparability, accessibility, and dependence between nodes). Our partition schema extends the weighted graph partition method to incorporate resource constraints of the weighted graph. Our schema is similar to Oliveira’s because the attaching utility values to different criteria are used for multi-criteria evaluation according to the expressions of the attributes’ values weighted by their corresponding utility values [13]. As described previously, the constraint type of related resources is determined according to accumulated weight of nodes and edges in the resource graph. For instance, the value of hard constraints may be higher than that of the soft constraints.

There are three types of vectors to represent the weight of the resource graph: a node weight vector, an edge weight vector, and a graph weight vector. The node weight vector, representing the degree of constraints associated with each node in the graph, stores accumulated weights computed from the normalization of each attribute’s weight of a node. Especially, the goal attribute in the node vector is important to verify whether or not two resources have a common goal. Similarly, the edge weight vector represents the degree of each edge’s weight of constraints in the graph by the accumulated edge weights of each relationship’s weight. Our graph weight vector stores accumulated weights determined by quantitative partition schema. Our quantitative partition schema is designed by the physical features followed by Koppler et al.’s [11]. In their partition schema, load balance, communication, and connection are considered as major measures and the measurements are based on several features of weighted graph: internal node and edge number, border node number, cut edge number, and neighbor node number, neighboring degree and ratio. In our partition schema, accumulated weights integrating from these three vectors (node, edge, graph vectors) become a criterion to partition a resource graph.

Due to the space limitation, a simple example of resource graph partition will be demonstrated. This example (Figure 2) shows only our partial partition schema according to relationships between resources in a resource graph. The weight edge vector shows the relationships between resources, including Ex, Rq, Ct, and Sr. For the negative relationship, Ex, we use -1 as a weight and for the positive relationships, Rq, Ct and Sr (according to Rq > Sr > Ct), according to their weights respectively. First, we partition the resource graph according to Excluded which is the strongest negative relationship. Thus, two resources related through an Excluded relationship should not exist within a same resource domain. In Figure 2-B, the initial graph is partitioned into two subgraphs {{A, C}, {B, D, E, F, G}} according to the Excluded relationship. Second, a threshold value computed from the sizes of nodes and edges (according to quantitative criteria) determine whether or not the graph should be further partitioned. Let us assume that we have a threshold value, ≤ 3. A subgraph with resource types {A, C} becomes the third resource domain (Resource Domain 3 in Figure 2). We need further partition into the subgraphs with {B, D, E, F, G}. For this partition, the required relationship, which is the highest positive relationship, is selected as a partition criterion. The Required relationship between the nodes B and F is first recognized and then a node G which has in/out-edge to those nodes. As a result, the nodes D and F become the first resource domain and the nodes B, F, and G become the second resource domain.

Figure 2-D shows the Resource Domains: Domain 1 = {D, F}, Domain 2 = {B, E, G}, Domain 3 = {A, C} and their local/global relationships. The resource graph partition simplifies the resource domains and their relationships. The global relationships determine the interactions between the resource domains and their constraints are specified as a set of resource domain policies. As an example, the policy of Domain 2 specifies an exclusive relationship with the Domain 3, while the policy of Domain 2 specifies a sharable relationship with the Domain 2.

Each resource domain is composed of related resources description and their requirements and policies on usage. A
provider agent represents a resource domain. The provider agent is an expert in a usage of the resources: which computing environment is required, when to use, how to use and in which order, what conditions to be satisfied before/ during/after for their use, and who an valid user is. Those are specified as forms of policies written in a data structure, called Resource Certificate (RC). The RC represents the resources available in the provider agent and becomes a part of Resource Domain Certificate (RDC) in Table 1. In Section 4.3, we will discuss the RDC as a collective representation combining consumers' and providers' perspectives.

4.2 Representing Users Using Consumer Agents

Now, we discuss how to represent diverse users’ requirements and privileges in a CDC environment. A user space can be represented by user graphs composed of user types and relationships between the classes. A user space seems to be simpler than the resource space. The user class has several attributes such as Goal (Gl), Role (Rl), Privilege (Pv), Task (Ts), Membership (Mb), and Collaborator (Cl). The relationships between user types are defined as Collaborate (Col) and Excluded (Exc). User types, according to the attribute values, are classified as Individual user (Iu), Group user (Gu), Exclusive user (Eu), and Collaborative user (Cu). Formally, the user graph is represented by directed acyclic graphs (DAGs), i.e., G = (V, E) where V is a set of vertices representing user type, UT = {Iu, Gu, Eu, Cu} determined by a set of attributes, UA = {Gl, Rl, Pv, Ts, Mb, Cl}, and E is the set of edges representing user relationship between user types, UR = {Col, Exc}.

A user graph is partition into smaller sub-graphs according to identified user types and the relationships between user types. The schema to partition the user graph is similar with the resource graph partition. The detailed partition process was described in [3]. We use a software agent, called a consumer agent, to represent a partitioned user graph. The consumer agent represents users’ role (Rl) and performs users’ task (Ts) possibly with its collaborators (Cl) if any. A privilege is an access right to the resources and a role is collection of such privileges [20]. Roles (Rl) are defined to specific duty requirements. A role provides a job, a set of actions and a collection of responsibilities and functions. User authorization to a role grants the user access to the privileges (Pv) defined in the role. Those are sources of verification in distributed authorization and access control paradigm. It is necessary to verify whether the user has the appropriate privileges, whether the user has actually transferred the privileges to the consumer agent and whether it is the delegate eligible for the request. Only an authorized user achieves the duty requirements associated with the role.

The consumer agent acts on behalf of the users with the detailed description on users (role, goal, membership, etc), which are specified as a set of policies in a data structure, called Consumer Certificate (CC). The consumers' policies in the CC are used for mapping with the providers’ in RC during negotiation and become a part of RDC (resource domain certificate, Table 1).

4.3 Construction of Resource Domain through Negotiation

In the distributed and collaborative computing environment, it is essential to have suitable mechanisms for an adequate control over the behavior of authorization systems. Our authentication and access control is based on an economic model; negotiation, agreement, and forming sharing groups. Various researches are undergoing on negotiation. In [4], a cooperative society was constructed according to roles, responsibilities, and preferences of the agents. Raiffa's negotiation model [16] is a bilateral multi-issues negotiation model. Faratin et. al [5] further extended the bilateral model to multi-lateral model.

Our negotiation model is based on OMG's multi-lateral negotiation [14] and the interactive negotiation [19]. The first model supports many parties involving negotiation and the second model supports multiple attributes involved in negotiation. In the negotiation process for forming a resource domain, more than one provider agent and more than one consumer agent may be involved. Since the mapping is between multiple consumer and provider agents on multiple issues (policies), it is a multi-party and multi-issue negotiation. The similarity between their needs and requirements initiates the negotiation while their diversity creates a continuous interaction until they reach to an agreement. The whole process until a resource domain is formed is called negotiation. The goal of the negotiation is to reach a mutual agreement on the policies of the domain among participants. The basic assumption under our approach is that the agreement through interactive negotiation results in an optimal mapping between resources and users.

In fact, there is no guarantee that the mapping always reaches to an agreement and the whole process tends to be complex and time-consuming. In order to establish a reasonable and efficient mapping between the consumer and provider agents, a policy-based supervised negotiation model is used. We introduce a mediator, called a secure resource agent (SRA, who supports the negotiation process for enhanced intensive interactions and a reasonable mapping according to the agents’ conditions and requirements.

Throughout interactive negotiation between CAs and PAs, a collective and comprehensive view of resource access policies is constructed and they reach to an agreement. An agreement can be established according to negotiation on quantitative and qualitative policies of a resource do-
Table 1. Resource Domain Certificate (RDC)

| PA | the identification of resource providers |
| CA | the identification of resource consumers |
| RC | description of resource providers |
| CC | description of resource consumers |
| Ag | agreed policies among participants |
| Pp | prioritized PAs and CAs |
| TAG | constrained/unconstrained agreement |
| To | duration which the agreement is valid |
| Ct | the cost for the resource |
| Pt | Actions to be taken when either PAs or CAs violate the agreement |

Algorithm Policy-Based-Negotiation (SRA: Agent) {
    /* Domain resource policies */
    DRP = Domain_Resource_Policy (D);
    /* Provider agents in the domain */
    DPA = Domain_Provider_Agent (D);
    /* Candidate Consumer agents for the domain */
    DCA = Domain_Consumer_Agent (D);
    /* Generate a certificate based on policies from PAs */
    RDC = Certification (SRA, DRP);
    /* Initialization of Agreement variable */
    Agreement = False;
    /* Initialization of Timeout variable */
    Timeout = False;
    /* Multicasting of initial certificate to PAs */
    Multicast (SRA, RDC, DPA);
    /* Repeat until an agreement is established. */
    While (Not Agreement) {
        /* request message to CAs for a negotiation on the RDC */
        Resp = Request_Negotiation (DCA, RDC)
        /* in case the request for certification negotiation is approved */
        if (Approved (SRA, Resp)) then
            /* digital signing on the RDC with SRA, DCAs, and DPAs */
            Cert = Sign_on_Certification (SRA, RDC, DCA, DPA)
            /* determine if the public key is identical to original one. */
            Verify_signatures (SRA, RDC, DCA, DPA);
            /* Verify the certificates are unrevoked. */
            Verify_revocation (RDC);
            /* multicast the final certificate */
            Multicast (SRA, RDC, DCA, DPA);
            /* Reset the Agreement variable */
            Agreement = True;
            /* construct a RD among SRA, CAs, PAs */
            Form_Resource_Domain (SRA, DCA, DPA, RDC);
        else (Timeout) then
            /* Send a message to participants on the termination */
            Terminate_Negotiation_Session (SRA, DPA, DCA);
            Exit; /* Terminate the negotiation process */
        else /* policy negotiation by changing requirements */
            DRP = Update_Group_Policy (SRA, DRP, DCA, DPA);
            /* reset the Timeout variable */
            Timeout = Update_Timeout (Timeout);
        end
    }
}

As mentioned previously, resource space and user space are already partitioned and their requirements are specified in Resource Certificates (RC) and Consumer Certificates (CC). Thus, the negotiation in this phase (communication and local constraints resolution) is significantly simplified. In this stage, we only focus on the issue of global communication and constraint resolution based on the Certificates (RC and CC) represented in a set of policies. A set of resource policies are described in quantitative and qualitative attributes and specific to a certain resource domain. The policies considered during the negotiation are an integration of two perspectives, RC and CC, to specify the policies applied to the resource sharing participants (SRAs, CAs, PAs). The policies describe the conditions/privilege of the resources usage and group/individual access. During the negotiation, every participant should agree on a set of domain policies represented in a form of Resource Domain Certificate (RDC).

Figure 3. Forming Resource Domains

We now describe a negotiation process between the participants, SRA, CAs and PAs to form a resource domain. In particular, the negotiation processes are supported by SRA: (1) conduct a resource conference by inviting consumer and provider agents available in resource pools and initiating the interactions, (2) construct the resource domain policies based on proposals from the participants, (3) form a resource domain according to agents agreements on the domain policy, (4) provide an efficient and secure communication protocol between members and (5) manage a resource group to support the resource domain policy update and group management. For our example (Fig-
through a resource partition process (Section 4.1), three resource domains are identified \{PA1, PA3\}, \{PA2\}, \{PA4\}. Three SRAs (SRA1, SRA2, SRA3) broadcast an invitation message to the consumer agents (CA1, CA2, CA3, CA4, CA5) in the consumer pool. The invitation message contains a resource domain name, a list of invitees, and Resource Domain Certificate (RDC). The RDC specifies initial domain policies partially generated by the Provider Agents. The CAs respond the SRAs with a reply message stating their participating willingness. After receiving participants' join message, the SRA checks the applicants' qualification based upon the resource domain's initial policies and decides the number of consumer agents who can join into their domain and the way to negotiate the policies and requirements of the domain. During the negotiation, a number of interactions are required until the participants reach to an agreement on the policies. In order to reduce unnecessary interactions during negotiation, a mediator-based negotiation by SRAs facilitates an optimal negotiation: minimum interactions, maximum agreement, and restricted negotiation period using a timeout. Finally, resource domains (logical ring), among the consumer and provider agents who accept the invitation, are formed. At the end of negotiation, three resource domains \{\{PA1, PA3, CA2, CA4, SRA1\}, \{PA2, CA1, CA5, SRA2\}, \{PA4, CA3, SRA3\}\} are formed and each Resource Domain is associated with a Resource Domain Certificate (Table 1). Policy-Based-Negotiation describes the detailed negotiation process. In the algorithm, our multiprotocol approach is used to pass a message to multiple members within a resource domain. The protocol is one of our local communication protocols and used for efficient broadcast using resource domain group. Refer [3] for our detailed security schema and communication protocols.

5 Conclusion

In this paper, we focus on distributed management for an efficient mapping between diverse resources and their users. We build an unified framework to map the diverse requirements of users with heterogeneous policies of resources. To do this, we introduced representation and partition schema for resource and user spaces, facilitated agents to represent diverse requirements from consumers and providers, developed algorithms to support policy-based negotiation and an agreement on policies, and designed hierarchical and ring structures for simplifying communication and maximizing utilities.

References