doi:10.2498/cit.1001294

A Distributed Architecture for Certificate-based Delegation of Business Process Accessibility in Virtual Organizations

Amir Talaei-Khoei

Asia-Pacific ubiquitous Healthcare research Centre (APuHC), Australian School of Business,
University of New South Wales (UNSW), Australia

In this paper, a distributed architecture has been proposed in order to support an authorization service more precisely in dynamically created Virtual Organizations (VO). In comparison with other existing architectures such as Akenti, VOMS and TAS, our architecture uses certificates on top of the distributed agent architecture for managing requested resources among the VOs. The most obscure issue in distributed agents is finding the proper node that keeps the particular requested certificates

In this paper, Chord's Finger Table has been improved to add extra search abilities on the ring architecture of Chord. The process of locating keys can be implemented on the top of the improved Chord by associating a key with each data item, and storing the key/data item pair at the node to which the key maps. In this article, a theoretical analysis is presented for simulations, which shows improvement in the number of passed hops to locate keys in the proposed method in comparison of standard chord, so it's more cost efficient.

Keywords: virtual organization, business process, authorization architecture, chord protocol

1. Introduction

Effective management of networks and systems requires cooperation of people within and across organizations [25]. Today, virtual organizations are quite common, but accessing the business processes, modifying and using them are still a big challenge. In a virtual organization one of the most important issues is designing and using the business process securely. In this paper, virtual organization is a collection of collaborative entities in a distributed environment that cooperate and sometimes compete on some shared and limited resources in a specified market. The shared resource and the distributed nature of the system make the accessibility an important topic.

In this section, as an application for accessibility of business processes, we refer to awareness model of cooperative management, presented in [25]. They present collaborative graph that shows the accessibility of roles on different tasks. This graph illustrates peer-to-peer architecture of virtual organization and how business processes need to be managed by accessibility.

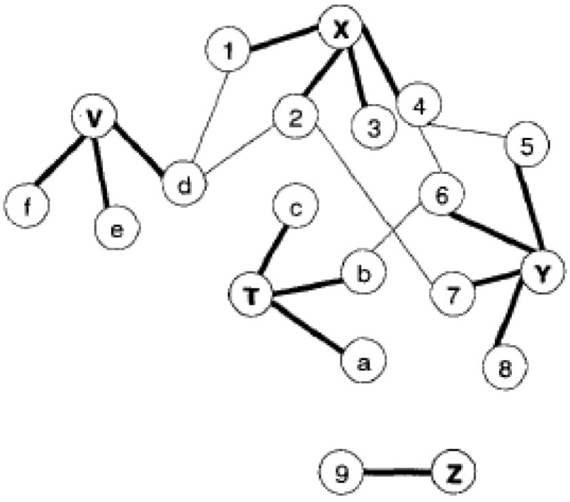


Figure 1. An enterprise collaborative graph with two processes (Application of the Method) [25].

Figure 1 shows an enterprise that has two processes, the first process has roles X, Y, and V and the second one has only Z. The first process is collaborative since there are at least two roles with task dependency. In this figure, dark-labeled circles present roles and others present tasks. Lines present artifacts, therefore there are two kinds of lines; (1) which are specified between role and tasks and (2) which are between two tasks and present shared resource for task dependencies [25]. These resources need to be securely managed to be accessed or updated. The architecture directs us to accessibility management in peer-to-peer systems.

The existing delegation systems for virtual organizations have been proposed based on Centralized Management Systems. However, in heterogeneous environments like Internet, most of the time we have to address the issue of several dynamically created Virtual Organizations, each of which independently manages a different domain of users by different authorization polices and rules.

A virtual organization architecture includes users and shared resources [1][2][3]. Users who want to work on a defined target, create a new VO in the distributed environment and shares information among the VOs. So, each VO consists of users, stakeholders and attributed authorities.

Users join a VO or leave it based on their needs. Also, stakeholders dynamically change their policies like available time or resources provided to users .In a dynamic VO users and stakeholders can participate in more than one VO at the same time. Therefore, stakeholders receive the user requests from different VOs. Also, a user can request resources from different VOs. Fine-grained authorization in virtual organization enables users to obtain their rights completely in defined areas. Users should make their requests in regions defined by stakeholders. For example, a user who has the right to execute just five business processes on target resource should not execute more than five processes.

In Section 2, authorization middleware is discussed. Section 3 illustrates Chord and some other peer-to-peer lookup methods. Section 4 is focused on some related works. Section 5 discuses all requirements and goals of the proposed method. Overview of the architecture and structure of certificates are presented in Sections 5 and 6 respectively. Finally, some discussion and results of simulation and analytical results are given in Section 7. Section 8 concludes the article.

2. Authorization Middleware

The most popular tool in nowadays virtual organization computing is called Globus Toolkit [6][11].

2.1. Globus Toolkit Features

The Globus Toolkit normally uses existing local resource mechanisms for authorization. A user is authenticated and then mapped to a local identity (e.g., a UNIX account) by a local configuration file.

This mapping also serves as an access control check: if the user is not listed in the local mapping configuration, access to the resource is denied. Once the user is mapped to a local identity, the Globus Toolkit (GT) relies solely on local policy management and enforcement mechanisms to constrain the user's actions to those allowed by local policy. This approach removes the fine-grained policy configuration and decision making from the GT services and allows the local operating system to act as a sandbox. Thus, administrators can use normal policy administration tools to configure policy. For example, a Globus Toolkit user is normally mapped to a local UNIX account. Standard UNIX file system permissions, quotes, group memberships, and so forth, are then used to configure and enforce policy.

GT uses proxy certificates to provide delegation ability, but it can't support fine-grained authorization. After performing simple authentication, the resource allows users to use all their rights. Thus, Globus cannot satisfy several requirements for fine-grained authorization in a dynamic VO environment.

Recently, more researches have been done related to fine-grained authorization. A Community Authorization Service (CAS) [6] [12] is introduced and designed as a storage for authorization information. In this model, there is a single CAS server that delegates all sets of rights to a user, and then the user can use his rights without any restriction.

2.2. Limitations of Classic GT

The classic Globus Toolkit authorization system [3] has the advantage of being easy for site administrators to understand and configure, because it uses existing local policy management and enforcement mechanisms with which the administrator is presumably already familiar. In terms of supporting a large VO, however, the GT has several shortcomings:

- Scalability: each personnel or policy change requires changing policy at each participating site;
- Lack of expressiveness: native OS methods may not be expressive enough to support VO policies;
- Consistency: different native OS methods may not support the same kinds of policies;
- Distribution: in order to maintain a consistent policy across the VO, each policy change must be propagated to each site involved. Any failure in propagation will cause an inconsistency in the policy.

3. Peer-to-Peer Lookup Methods

Virtual organizations usually follow a peer-to-peer architecture. A fundamental problem in such applications is efficiently finding a node that stores particular data. This paper uses chord, and in the rest of this section, we first overview chord and then introduce other methods and compare them with chord. For more information, readers are recommended to read [17].

Chord presents just one operation: *give a key*; it maps the key onto nodes using consistent distributed hash table (DHT). Boyvat [22] defines DHT as "a set of algorithms developed to enable us to have efficient distributed and decentralized networks." In chord, consistent hashing assigns keys to node as follows. Identifiers are ordered in an identifier circle modulo 2^m. Key k is assigned to the first node whose identifier is equal or follows k in the identifier space. This node is called successor and is defined by first node clockwise from k, if identifier is presented as circle of numbers from 0 to $2^m - 1$ [17].

Distributed Name System (DNS) provides a host name to IP address mapping [18]. Chord can provide the same service when name represents the key and IP address represents the value. DNS uses a name structure, while chord does not use any structure for names. DNS can not also find data on a practical machine, and chord can [17].

Freenet peer-to-peer storage system [19] is also decentralized, symmetric and adaptable for leaving and joining hosts (like chord). Chord has predictable time for lookup, which is not supported by Freenet [17].

Glob System [20] has wide-area location service to map object identifiers to the location of moving objects. Glob has a hierarchical, topological and geographical approach for construction of world-wide search tree. Chord without involving any hierarchy and high overload handles the world-wide search [17].

Kademlia [21] provides a DHS. It effectively treats nodes as leaves in a binary tree, where each node's position is determined by the shortest unique prefix of its ID. The lookup algorithm provided by Kademlia makes it possible to reach any desired node in logarithmic steps. Major innovation of Kademlia is that the distance between points in key space is managed by an XOR metric. The same as chord, it's symmetric as well as decentralized. Chord for neighbour selection and natural support for sequential neighbours work more efficiently [17].

Pastry [23] performs application-level routing and object location in a potentially very large overlay network of nodes connected via the Internet. Each Pastry node keeps track of its immediate neighbours in the nodeId space and notifies applications of new node arrivals, node failures and recoveries. Pastry takes into account network locality; it seeks to minimize the distance messages travel, according to scalar proximity metric like the number of IP routing hops. Pastry is prefix-based which makes it different from chord. The Pastry method seems to have more flexibility than the Chord method because "successor" is not so rigidly defined with the Pastry identifiers. Pastry, for example, takes into account network locality by adjusting nodes in its routing table. Chord cannot handle this problem because there can only be one successor. Properties, such as the identifier creation and network proximity metric, are defined by the application using the Pastry substrate. Kelaskar et al [24] present experiments and discussions that illustrate that chord has better discovery performance.

4. Related Work

The CAS [6] model has been proposed for a community authorization in the Globus toolkit. In CAS, each VO has its own CAS Server. The central administrator who is a trusted third party, delegates all the necessary rights to use resources. When a user requests rights, the CAS server delegates all sets of rights to the user, and then the user can use his/her rights without any restriction. The user's request is then executed by the resource if the stakeholder maintains a trust with the certifying CAS service and the request is authorized by the CAS server.

In VOMS5][8], group memberships and group rights are managed separately. The VOMS server like CAS manages group memberships which indicate a list of users and roles in each group. Group rights, which are local policies about group memberships, are distributed among resources. They are not stored in the central server. Based upon the roles that have been assigned to the user by the VOMS server, the stakeholder decides how much access privilege is granted to the user. So, a user obtains the VOMS server's a priori approval about his group membership and then resource's approval about group rights.

The Akenti [9] model provides restricted access to resources managed by several administrators in a distributed environment. In Akenti, the resource receives a user's request, and then passes the user identity, attributes, and requests to Akenti policy engine for an authorization decision. The resource complies with the decision of the Akenti policy engine.

The PERMIS [9] model is similar to the Akenti model. A user sends requests to a resource without authorization information, and the resource checks the user's rights through the PERMIS system. PERMIS uses a X.509 attribute certificate for the access control infrastructure.

The TAS [9] model has a ticket-based architecture based on Chord and again like PERMIS uses a X.509 attribute certificate for the access control infrastructure.

5. Requirements and Goals

5.1. Requirements

To provide fine-grained authorization in a dynamic VO environment, an authorization system must adapt efficiently to the dynamic changes of VOs and satisfy the requirements of each entity: a stakeholder, an attribute authority, and a user. A stakeholder participates in one or more VOs by sharing some or all of its resources. An attribute authority offers information about who has the membership of the VO and which resources are available in the VO. The relationship among entities of VO can vary dynamically over time, in terms of the resources involved, the nature of the access permitted, and the participants to whom access is permitted. The followings are the requirements of a fine-grained authorization system in a dynamic VO, from the viewpoint of each entity.

- Stakeholder: Since stakeholders grant their resources to users in the VOs that they join, only jobs of users in those VOs should be authorized. However, stakeholders need not know the information regarding all users in the VOs. They only need to ensure that the owner of a submitted job is a member of the permitted VOs. In addition, stakeholders should inform users of the amount of currently available resources for each VO to reduce unnecessary requests.
- VO Attribute Authority: Attribute authority plays the role of an intermediate broker between stakeholders and users. If a stakeholder offers its resource to a VO, the Attribute authority enables users in the VO to use the resource. The attribute authority also keeps a policy that is applied to users. Through this policy, the attribute authority assures that users only request the restricted resources.
- User: Users must be able to know the rights of an available resource. And he should delegate proper rights to processes to allow the processes to run. According to the policy of the stakeholder and the attribute authority, resources that a user can access change continuously in a dynamic VO environment.

5.2. Goals

This paper proposes distributed authorization service architecture, with the following services:

- Fine-grained authorization method: Although much research has been conducted on fine-grained authorization in the virtual organization, the restriction of rights when delegating user‘s rights hasn‘t been studied extensively. In addition, only a trusted server delegates all sets of rights to a user. Also, few studies have suggested a fine-grained authorization service for stakeholders or users. Thus, we focus on a fine-grained authorization method for stakeholders, users, and VOs.
- Guarantee of the authorized resources: Previous authorization architectures have hardly been concerned about guaranteeing the resource that a user has needed. So, they need another service for checking resource availability. We propose a method that the stakeholder guarantees that the previously authorized user uses the resource.

6. Overview of the Architecture

The certificate-based authorization service architecture uses certificates which contain delegated rights from each entity in a VO. The attribute authorities, stakeholders, and users in a VO issue their own certificates to delegate their rights. By sharing these certificates, the VO is constructed. The VO also changes its policy as each VO entity re-issues a certificate containing the changed policy. To share and locate certificates from various entities, we use a distributed agent system for certificate management. Agents in the distributed agent system share their certificates from the VO entities. We use a structured peer-to-peer (P2P) system among agents, so locating certificates is done efficiently. In our architecture, an attribute authority only publishes the attribute certificate for each user in the VO. Stakeholders also issue their certificates, which inform their policies to the VO. A user in the VO acquires necessary certificates from the distributed agent system in order to submit a job. And then, the user submits certificates including a user certificate to the stakeholder and runs the job after being authorized by the stakeholder. Here is an example of the proposed architecture.

In Table 1, symbols are listed to clarify architecture elements which are used in Figure 1. In the meantime, Figure 2 shows all the architecture components and relations:

- Tree resource provider centers which are introduced by Site1, Site2 and Site3.
- Two attribute authority centers that belong to VO1, VO2.
- Tree resource certificate provider center, called stakeholder, as mentioned before.
- One user from VO1 and two users from VO2.

Name	VO1	VO2
Resource	⛀	⛂
User	☺	☻
Attribute Authority Issuer	🛇	🛇
Stake-holder	⊕	⊕
Resource Certificate	R	R
User Certificate	U	U
Attribute Certificate	A	A
Active Agent	●	
Inactive Agent	○	

Table 1. List of names to symbols.

Suppose that User1 who is connected to node number zero needs a service provided by Site1 and is inquiring for a proper resource certificate from that resource. Also, suppose that Site1's stakeholder has issued and registered resource certificates on distributed agents with unique identifiers which are generated by HAS-1 hash function.

The input of hash function for generating these identifiers is the combination of service name and VO name. The resultant string is by itself a unique string, but to have an equal probability of distribution on nodes the hash algorithm is used. Also VO2 attribute authority is published and an attribute certificate for User1 is registered too. The identifier of this attribute certificate is the result of hashing of the string, which is a combination of VO name and User name.

As the first attempt for finding desired certificate by User1 from VO2, the Directory Service of VO2 is searched. If any proper service is found, that service name and the service provider VO2 name will be used for generating resource certificate identifier in common. So, the results must be the same as the generated and registered results of Stakeholder 1. Due to same used string as input for hashing function. As the next step, User1 from VO2 sends the produced identifier to agent zero, the node that is connected to it, and waits for search result. Another certificate that should be produced by User1 is the attribute certificate. This certificate identifier will be the hashed results of the string which is composed of VO2 and User1. This identifier will be passed to agent zero like a resource identifier.

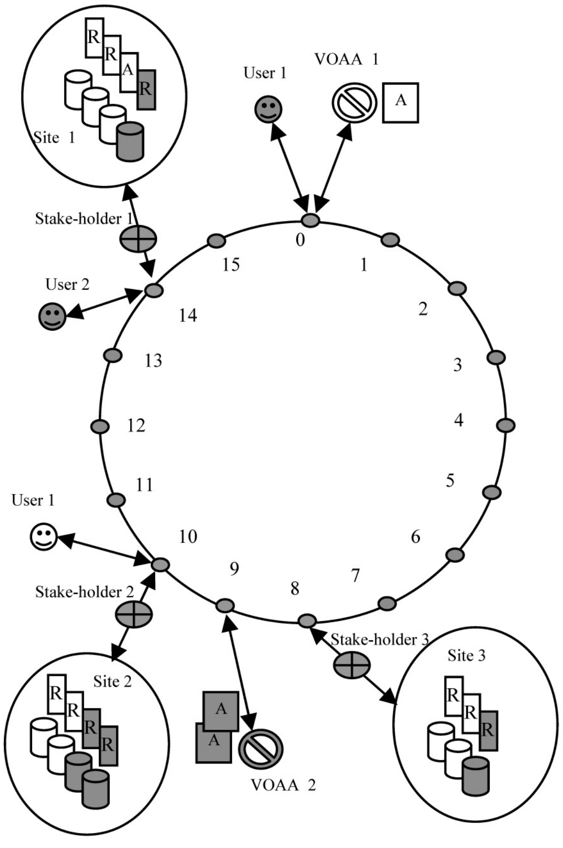
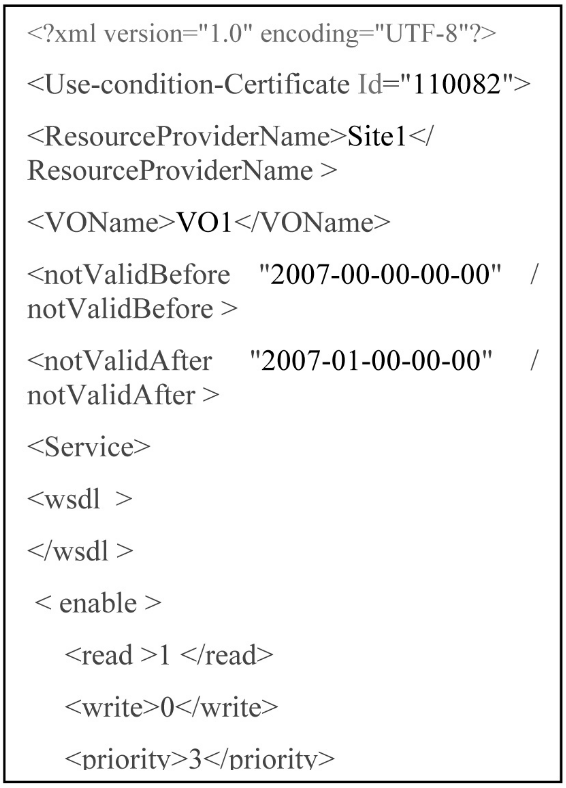


Figure 2. The proposed architecture.

After receiving attribute certificate and resource certificate from distributed agent, User1 reconstructs these two certificates as a new certificate and signs this new one in the name of User certificate. The generated user certificate is presented to Site1 and after the authentication process, User1 is allowed to use the desired service.

6.1. Certificate-based Authorization Service

The main idea of the proposed architecture is to use certificates for delegated rights in the VO. The certificate is unforgivable and exchangeable among VO entities for resource control. The certificate record is an XML object that shows the specified policy controls a resource in the specified time interval in the certificate. To protect the integrity of a certificate, each certificate record is signed by the private key of the issuer [13]. An example of a resource certificate has been given in code segment of Figure 3.



```
<?xml version="1.0" encoding="UTF-8"?>
<Use-condition-Certificate Id="110082">
<ResourceProviderName>Site1</
ResourceProviderName >
<VOName>VO1</VOName>
<notValidBefore    "2007-00-00-00-00"    /
notValidBefore >
<notValidAfter     "2007-01-00-00-00"    /
notValidAfter >
<Service>
<wsdl >
</wsdl >
 < enable >
    <read >1 </read>
    <write>0</write>
    <priority>3</priority>
```

Figure 3. Format of resource provider certificate.

6.2. Distributed Agents Architecture

The distributed agent system is a certificate management system which is independent of any site or VO. Agents provide certificate management services including certificate registration, certificate location, and certificate revocation.

We constructed an agent system to improve Chord P2P system which reduces the overhead of each service and is about 30% faster than basic Chord[14]. In the Chord, each node, which is an agent in our system, has a unique identifier from 0 to2^{m-1}. In the basic Chord, searching to find keys is continued only in one direction on the ring based of the finger tables. Finger tables have only successors. In the improved Chord, this process goes on both directions in the ring based on the finger tables. Finger tables have both predecessors and successors. Thus, the number of passed nodes is reduced by about 30% because we marked the passed nodes one third only by more information in each node. This statement is illustrated with comparison between Chord and improved Chord based on two methods: simulation and analytic model in the results and conclusion section, presented in Section 7.2 and 7.3

7. Discussion

In this section, the Chord and improved Chord are compared with each other. After that, proposed architecture is compared with Akenti architecture that is one of most famous authorization architectures for VO services. The comparison includes these three points of view:

- Security overheads on communication and encryption/decryption of messages.
- Network overheads and number of generated messages for client's requests in both architectures.
- Search time overhead of each certificate in database of both systems.

Prior to these comparisons, we implemented our proposed certificate-based authorization service.

7.1. Implementation

Agents share and provide published certificates. We implemented the distributed agent-based model on the improved Chord method, which is a structured P2P architecture. The agent performs the certificate registration and location service. In the registration service, the agents save published certificate in the local xml database and advertise the certificate's reference to other agents. In the location service, the agent finds the advertised certificate reference and provides proper certificates to users.

The agent architecture is shown in Figure 4. Axis is a Simple Object Access Protocol (SOAP) engine that is a framework for constructing SOAP message service. Agent Service is a message style web service that receives and returns certificate XML documents in the SOAP envelope.

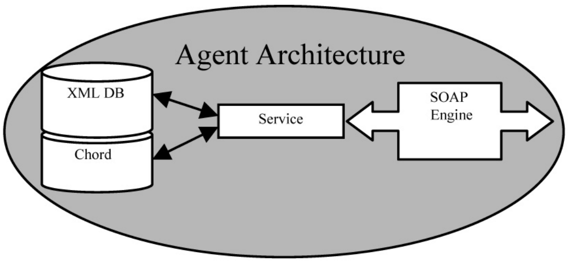


Figure 4. Internal connection.

XMLDB is designed to store and retrieve large numbers of XML documents such as certificates. Chord is a Distributed Hash Table (DHT) – based structured P2P architecture. The underlying principle of the Chord is a fast consistent hash function that is used to map keywords to some point in a logical P2P ring.

7.2. Chord vs. Improved Chord

In this section, we compare both the analytical model results and results of experiments by simulation.

7.2.1. Analytical Model Results

In this section we first calculate the average number of passed hops for Chord algorithm.

After average number of passed hops for improved Chord algorithm has been calculated and compared with Chord.

Suppose that N is the number of nodes. In the Chord we have these equations:

- The probability that the number of passed nodes be equal to one:

P(Hop=1)=1/N

- The probability that the number of passed nodes be equal to two:

P(Hop=2)=3/N

- The probability that the number of passed nodes be equal to three:

P(Hop=3)=2^{3-1}/N

- The probability of that the number of passed nodes be equal to n:

P(Hop=n)=2^{n-1}/N

Considering the above four equations, we can extract the equation below for average passed number of Chord algorithm:

$$AverageChord = \frac{\left(\frac{1}{N}\times 1\right)+\left(\frac{3}{N}\times 2\right)}{}$$

$$\frac{+\left(\frac{2^{3-1}}{N}\times 3\right)+\ldots+\left(\frac{2^{n-1}}{N}\times n\right)}{N}$$

$$= 1+6+\left(2^{3-1}\times 3\right)+\ldots+\left(2^{n-1}\times n\right)$$

Finally, we have below equation:

$$AverageChord = 7+\sum_{x=2}^{x=\log(N/2)} 2^{x}(x+1), N \geq 8$$

The same process will be repeated for calculating average passed nodes for improved Chord. Suppose that N is the number of nodes. In the improved Chord we have these equations:

- The probability that the number of passed nodes be equal to one:

P(Hop=1)=2/N

- The probability that the number of passed nodes be equal to two:

P(Hop=2)=6/N

- The probability that the number of passed nodes be equal to three:

P(Hop=3)=2×2^{3-1}/N

- The probability that the number of passed nodes be equal to n-1:

P(Hop=n-1)=2^{n-1}/N

- The probability that the number of passed nodes be equal to n:

P(Hop=n)=0

Considering the above five equations we can extract the equation below for average passed number of improved Chord algorithm:

$$Average\text{-}improved\text{-}Chord$$

$$= 2\times\left(\frac{\left(\frac{1}{N}\times 1\right)+\left(\frac{3}{N}\times 2\right)+\left(\frac{2^{3-1}}{N}\times 3\right)}{N}\right.$$

$$\left.\frac{+\ldots+\left(\frac{2^{n-2}}{N}\times n\right)}{}\right) = 2+12+2$$

$$\times(2^{3-1}\times 3)+\ldots+(2^{n-1}\times n)$$

Finally we have below equation:

$$AverageimprovedChord$$

$$= 14+\sum_{x=2}^{x=\log(N/2)-1} 2^{x+1}(x+1), N \geq 8$$

Now that both equations are calculated for Chord and improved Chord average passed nodes, the ratio of Chord to improved Chord is shown below:

$$Ratio = \frac{7+\sum_{x=2}^{x=\log(N/2)} 2^{x}(x+1)}{14+\sum_{x=2}^{x=\log(N/2)-1} 2^{(x+1)}(x+1)}$$

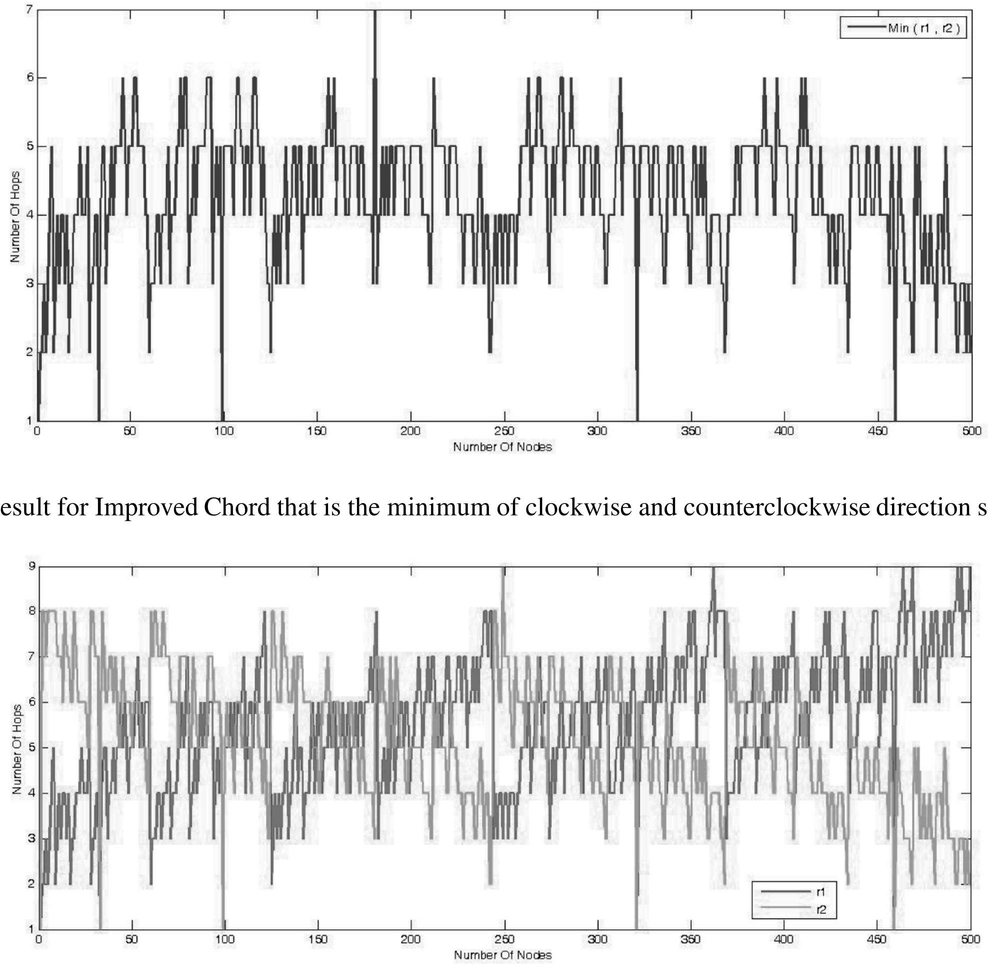
The results of simulations are shown in Table 2. Analytical results show that we have reduction in the number of average passed nodes about 34% for $N = 16$, or $m = 4$, toward 12% for $N = 1024$ or $m = 10$.

	Average passed	**Average passed**	**Average passed**	**Average passed**
Chord	3.18	5.04	7.01	9
Improved Chord	2.37	4.09	6.02	8
Improvement	34%	23%	16%	12%

Table 2. Analytical results (comparison of Chord and Improved Chord algorithms).

7.2.2. Simulation Results

Chord and Improved Chord algorithms have been simulated and Figures 5a and 5b show the results of simulation. In our simulation, the number of nodes is 500 and the number of identifiers is 5000. Suppose that the starting nodes for searching identifiers are always a defined node such as node 1. There are two parameters r1 and r2 that indicate the number of hops in clockwise and counterclockwise directions on the ring respectively. In fact, r1 is the result for basic Chord algorithm. Both r1 and r2 in Improved Chord algorithm are shown in Figure 8. The minimum passed hops that are results of both r1 and r2 are shown in Figure 5a. As Fig-



5a. Result for Improved Chord that is the minimum of clockwise and counterclockwise direction search.

5b. Results for both r1(clockwise) and r2(counterclockwise) direction search.

Figure 5. Comparison of Chord and Improved Chord.

	Max passed nodes	Average passed nodes	Max passed nodes	Average passed nodes	Max passed nodes	Average passed nodes	Max passed nodes	Average passed nodes
Chord	4	2.625	5	3.466	7	4.508	9	5.504
Improved Chord	3	2.125	4	2.666	5	3.341	7	4.294
Improvement	33%	23%	25%	30%	40%	34%	28%	28%

Table 3. Simulation results (comparison of Chord and Improved Chord algorithms).

ure 5b shows the maximum passed nodes has been reduced from 9 in basic Chord to 7 in improved one. More details on comparison of two methods are shown in Table 3. Considering the average passed nodes we have had about 30% reduction in number of passed over nodes.

7.3. Proposed Architecture vs. Akenti Architecture

As we discussed before, in this section we compare the architecture proposed in this paper with Akenti architecture. This comparison stands on three aspects as follows.

7.3.1. Security Overheads

Three main security rules are discussed below:

Server Secure Socket(SSL) – This mechanism uses Server Secure Socket (SSL) with GSI certificates, which are based on X.509 certificates. SSL is lightweight because it does not involve any XML manipulations. Also, except the initial handshake, SSL uses a symmetric cipher, which is known to be much faster than an asymmetric cipher because it transports layer security and does not work, if the connection includes multiple hops, as a feature supported by SOAP.

XML-Signature – XML-Signature is a standard for digital signatures for XML documents. The usage of XML-Signature with SOAP messages is described in WS-Security[15] specification. With XML-Signature, each message is signed with X.509 certificate (GSI certificate). It ensures the integrity of a message, but it does not support replay-attack prevention. As opposed to the WS-SecureConversation, which will be described later, this mechanism is stateless and does not need any initial handshakes. Thus, it is suitable for a single invocation of a service.

WS-SecureConversation – is a relatively new protocol to establish and use secure contexts with SOAP messages. First, a secure context is established between a client and a server. Once the security context is established, the following messages are signed using the XML-Signature standard. It is faster because it uses a symmetric key to sign messages, but it requires additional round trips to establish a connection.

Three main methods of protecting message integrity are discussed before. Figure 6 that is borrowed from [16] shows the comparison between SSL and XML-Signature methods. Here, the focus is on both methods: GT(httpg w/o

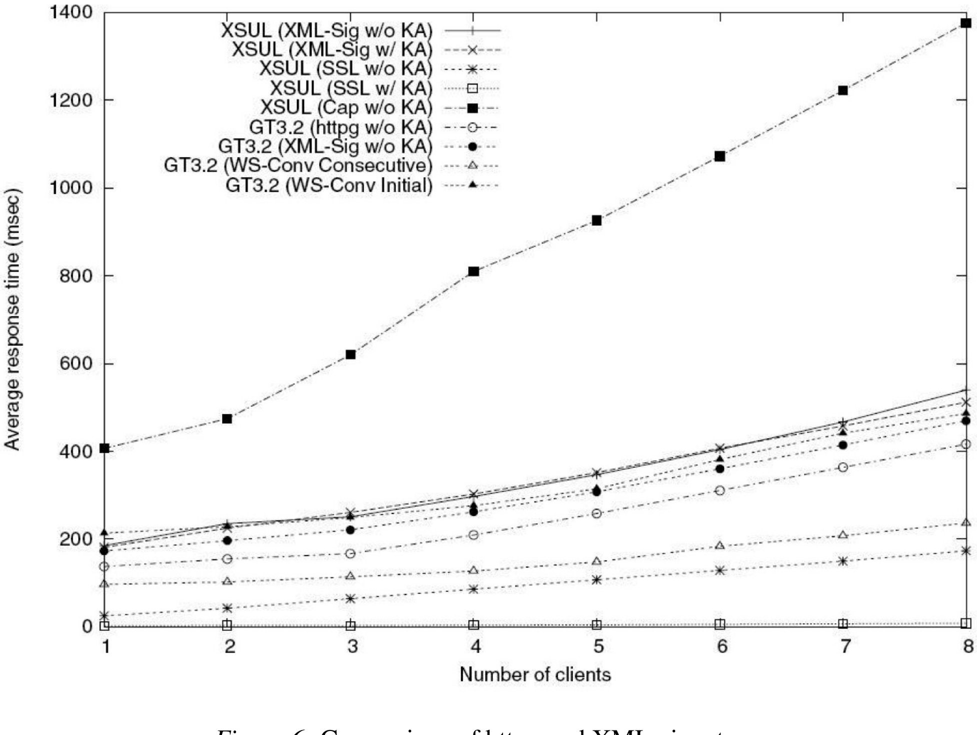


Figure 6. Comparison of httpg and XML-signature.

KA) and GT(XML-signature w/o KA) that are employed in Akenti and the proposed method. As the diagram shows, migrating to XML-signature methods have reduced response time by about 10% in a simple echo server and client environment.

7.3.2. Network Overheads

Network overhead is an important factor for evaluating a network-based system like Akenti and the proposed architecture. In this paper the number of generated messages for every client request is considered as a parameter for network traffic. By considering Figure 7, the number of messages in Akenti architecture is calculated as below (Figure 7a):

$$NumMessages = 2 + 2 \times N$$

The number of messages for one client request in proposed architecture is calculated by the following equation (Figure 7b):

$$NumMessages = 3 + \log_2^{N-1}$$

Figure 8 shows the ratio of messages in both systems for different number of nodes. It is clear that the increasing number of nodes heavily impacts the ratio.

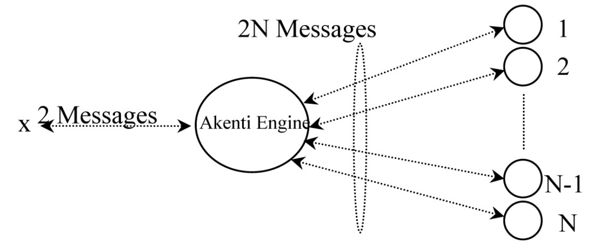


Figure 7a. The Akenti architecture.

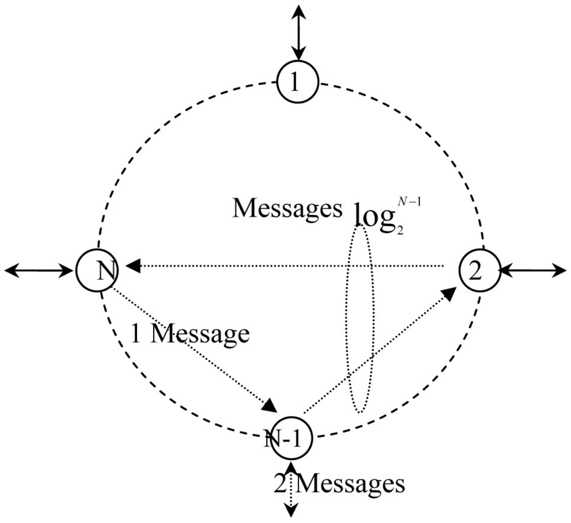


Figure 7b. The proposed architecture.

Figure 7. Architecture of Akenti and the proposed model based on the number of messages.

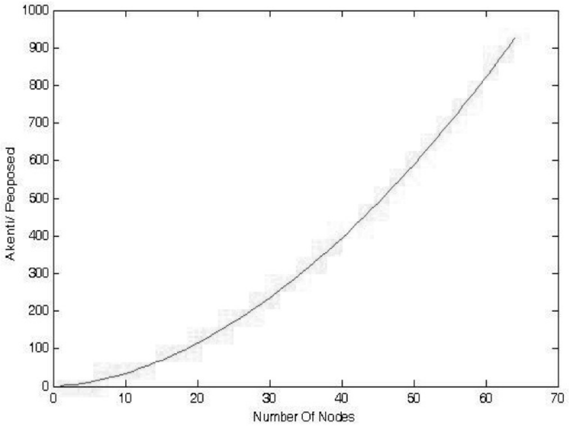


Figure 8. Ratio of number of messages (Akenti/proposed).

7.3.3. Database search time overhead

One important feature for the architecture is overhead in search time. The Akenti system is not scalable regarding the number of messages. Thus, the number of agents in Akenti architecture has a limitation that causes problems like increasing number of certificates on each agent while the proposed architecture is fully scalable and there are no limitations on number of agents. So, when the number of certificates is increased, there is a possibility of increasing the number of nodes. This would cause reduction of certificates on each agent and also reduction in search time.

8. Conclusion

Virtual organizations often implement business processes in distributed peer-to-peer architecture. Shared resources and artifacts make accessibility of business processes a challenging issue. One applicability of this issue is awareness of peers in a collaborative business process, when different peers work on dependent tasks.

This article proposes a distributed architecture for authorization of business processes in dynamic virtual organizations. This architecture supports: (1) Fine-grained authorization, (2) Guarantee of the authorized resources. The architecture tries to improve Chord's Finger Tables with adding extra search abilities on the ring architecture of Chord.

In this aspect, Chord's Finger Table has been improved and Chord has been added extra search abilities. The evaluation stands on (1) comparison of Chord and Improved Chord presented in this paper (2) comparison between proposed and Akenti architectures. The first comparison has been done by analytical model and simulation experiments. The second one has been discussed disruptively in security, network overload and database search time overhead criteria.

Results of the theoretical analysis, simulations and experiments show that using improved Chord method achieved 30% reduction in number of passed hops for locating the keys. Hence, Improved Chord is cost efficient, scalable, with lower communication cost and about 30% faster than the basic Chord.

References

[1] I. Foster, C. Kesselman, J. Nick, S. Tuecke, The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. *Open Grid Service Infrastructure WG (GGF)*, June 2002.

[2] I. Foster, C. Kesselman, S. Tuecke, The Anatomy of the Grid: Enabling Scalable Virtual Organization. *International Journal of Supercomputer Applications*, pp. 200–222, 2001.

[3] G. Laccetti, G. Schmid, A Framework Model for Grid Security. *Future Generation Computer Systems*, No. 23, 702–713, 2007.

[4] I. Foster, C. Kesselman, The Globus Project: A Status Report. *Proceedings of the 7th Heterogeneous Computing Workshop*, pp. 4–19, March 1998.

[5] R. Alfieria, R. Cecchinib, V. Ciaschinic, L. dell'Agnello, Á. Frohnere, K. Lorenteyf, F. Spatarog, From gridmap-file to VOMS: managing authorization in a Grid environment. No. 21, 549–558, 2005.

[6] L. Pearlman, V. Welch, I. Foster, C. Kesselman, A Community Authorization Service for Group Collaboration. *Proceedings of the IEEE 3rd International Workshop on Policies for Distributed Systems and Networks*, 2002.

[7] M. Thompson, W. Johnston, S. Mudumbai, G. Hoo, K. Jackson, A. Essiari, Certificate-based Access Control for Widely Distributed Resources. *Proceedings of the 8th USENIX Security Symposium*, pp. 215–227, August 1999.

[8] R. Alfieri, R. Cecchini, V. Ciaschini, L. dell'Agnello, Á. Frohner, A. Gianoli, K. Lõrentey, F. Spataro, VOMS, an Authorization System for Virtual Organizations. *European Across Grids Conference*, pp. 33–40, 2003.

[9] D. W. Chadwick, A. Otenko, The PERMIS X.509 role based privilege management infrastructure. *Future Generation Comp. Syst. 19(2)*, pp. 277–289, 2003.

[10] B. Kim, S. Hong, J. Kim, Ticket-based fine-grained authorization service in the dynamic VO environment. *Proceedings of the 2004 workshop on Secure web service*, 2004.

[11] The Globus Alliance, `http://www.globus.org`.

[12] V. Welch, F. Siebenlist, I. Foster, J. Bresnahan, K. Czajkowski, J. Gawor, C. Kesselman, S. Meder, L. Pearlman, S. Tuecke, Security for Grid Services. HPDC-12, *IEEE Press*, June 2003.

[13] XMS Signature, `http://www.w3c.org/Signature`.

[14] I. Stoica, R. Morris, D. L. Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, H. Balakrishnan, Chord: A Scalable Peer-to-peer Lookup Protocol for Internet Applications. *IEEE/ACM Transactions on Networking*, Vol. 11, No. 1, pp. 17–32, February 2003.

[15] A. Nadalin, C. Kaler, P. Hallam-Baker, R. Monzillo, Eds., (2004, Mar.) Web Services Security: SOAP Message Security 1.0 (WS-Security 2004). [Online]. Available: `http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss`

[16] S. Shirasuna, A. Slominski, L. Fang, D. Gannon, Performance Comparison of Security Mechanisms for Grid Services. *Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing (GRID'04)*.

[17] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, H. Balakrishnan, Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. *IEEE/ACM Transaction on Networking*, 2003.

[18] P. Mockapetris, K. J. Dunlap, Development of the domain Name System. *Proceedings of Special Interest Group on Data Communication*, 1988.

[19] I. Clarke, A Distributed Anonymous Information Storage and Retrieval Systems. Master thesis. *University of Edinbergh*, 1999.

[20] A. BAKKER, E. AMADE, G. BALLINTIJN, I. KUZ, P. VERKAIK, I. V. D. WIJK, M. V. STEEN, A. TANENBAUM, The Glob Distributed Network. *Proceedings of USENIX Annual Technical Conference*, 2000.

[21] P. MAYMOUNKOV, D. MAZI'ERES, Kademlia: A Peer-to-peer Information System Based on the XOR Metric. *Proeedings of 1st International Workshop on Peer-to-peer Systems (IPTPS)*, 2002.

[22] B. BOYVAT, Kenosis: A P2P RPC system using the Kademlia DHT. *Telecommunications Software and Multimedia Laboratory*, Finland.

[23] A. ROWSTRON, P. DRUSCHEL, Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-peer Systems. Middleware, Springer-Link, 2001.

[24] M. KELASKAR, V. MATOSSIAN, P. MEHRA, D. PAUL, M. PARASHAR, A Study of Discovery Mechanisms for Peer-to-peer Applications. *Proceedings of the Second IEEE/ACM International Symposium on Cluster Computing and the Grid*, 2002.

[25] S. BENFORD, J. BOWERS, L. E. FAHLEN, J. MARIANI, T. RODDEN, Supporting Cooperative Work in Virtual Environments. *The Computer Journal*, vol. 37, Aug. 1994, pp. 653–668.

Received: June, 2008
Revised: January, 2009
Accepted: September, 2009

Contact address:
Amir Talaei-Khoei
APuHC
University od New South Wales
Sydney, NSW 2052
Australia
e-mail: amirtk@student.unsw.edu.au

AMIR TALAEI-KHOEI joined Asia-Pacific ubiquitous Healthcare research Centre (APuHC) in 2008, as a part of Australian School of Business at University of New South Wales (UNSW), where he works closely with computer scientists, business experts and health professionals. Prior to that, he did a short-term research on business modelling at Systemic Modelling Lab, Ecole Polytechnique Federale de Lausanne (EPEL), Switzerland. He is doing hid PhD in Information Systems at UNSW and achieved his master in Software Engineering of Distributed Systems from Royal Institute of Technology, Sweden. Amir is interested in formalization and modelling approaches to bridge mobile agents, artificial intelligence and Computer Supported Cooperative Work (CSCW). As a part of his research, he applies the theory to different actual scenarios in healthcare e.g. games for home-based rehabilitation, disaster management, and mobile health monitoring.