February 2009

Cloud Computing in Virtual Environments

Kristen Hardwick
Clemson University, hardwic@clemson.edu

John Fisher
Clemson University, jfisher@cs.clemson.edu

Ben Sterrett
Clemson University, bsterre@clemson.edu

Christine Minor
Clemson University, mminor@clemson.edu

Sebastien Goasguen
Clemson University, sebgoa@clemson.edu

Cloud Computing in Virtual Environments

Kristen Hardwick
Clemson University
hardwic@clemson.edu

John Fisher
Clemson University
jfisher@cs.clemson.edu

Ben Sterrett
Clemson University
bsterre@clemson.edu

Christine Minor
Clemson University
mminor@clemson.edu

Sebastien Goasguen
Clemson University
sebgoa@clemson.edu

ABSTRACT

In this paper we present the basis of a new middleware service that provisions clouds for virtual organizations (VOs).

This service makes use of a virtual environment's inherent ability to render objects to represent clouds with real clouds. These clouds are created on demand by avatars and tagged to provide a rudimentary semantic that can be used for searching. Clouds are then loaded with an inventory that contains objects and scripts used to access remote resources. Compute resources, sensor networks, and visualization services can be part of the cloud's inventory. Second Life is used to implement this cloud computing service. The authorization mechanism of Second Life and an external database managed by our cloud service is used to restrict access to clouds based on avatar roles and group membership.

We argue that this service can be used effectively by a VO to provide a very interactive experience for its members as well as potential collaboration between multiple VOs. Cloud computing takes a very figurative meaning in our work since we literally create clouds in the environment and manage their ownership, access and capabilities. We believe this innovative work brings together grid computing, social networking and virtual environments in a very attractive and understandable way.

Keywords

Virtual worlds, collaborative research, cloud computing, services, middleware, virtual organizations.

INTRODUCTION

Multi-User Virtual Environments

Virtual worlds are "computer-based simulated environment[s] that users can inhabit and in which they interact with others" [6], these worlds and Multi-user Virtual Environments (MUVEs), have gained popularity in recent years. They are most commonly used for social interaction, often in the format of a game like World of Warcraft or EVE Online. However, MUVES are being used more often for purposes other than entertainment. For example, in June 2007, "Second Life's virtual characters exchange[d] an average of $1,700,000 (U.S.) daily" [2].

MUVEs are also being used for teaching and research. This is because of the benefit of such activities in a social environment. MUVEs are appealing to researchers because it allows them, through their avatars, to interact with others in the virtual environment even though they may be on opposite sides of the globe.

Many MUVEs allow users to access online resources from within the environment using HTTPRequests or XML-RPC calls. This allows in-world avatars to manipulate resources directly, and for the creation of innovative ways to display results. However, virtual worlds come with a sense of anonymity that is difficult to duplicate with real life meetings. It may be difficult to hold an avatar accountable for mischievous behavior when real-world resources are being accessed, so full access to secure resources is rarely granted. In order to achieve the full potential of accessing real world resources, it is necessary to have authentication protocols embedded in the environment.

The cloud concept described in this paper was implemented in a specific MUVE, Linden Lab's Second Life, but it can be adapted for any MUVE that allows user scripts. Second Life users are always required to log in before accessing the virtual environment; however, there is a need for additional authentication when real world resources are involved. All Second Life avatars should not have the same privileges for all resources, and there needs to be a way to differentiate. One way to solve this problem is to store the objects that access resources in another object, represented as a cloud. Avatars that wish to use the resources first access the cloud, which performs an authorization process. If the resources are public, the cloud will not need

to check the avatar's permission level. However, if the resources are private, authorized users can create a group and claim a cloud, which will then authenticate based on group membership and roles. The cloud computing service is represented in the world as actual clouds, and users must interact with each of the clouds in order to access the resources that it contains.

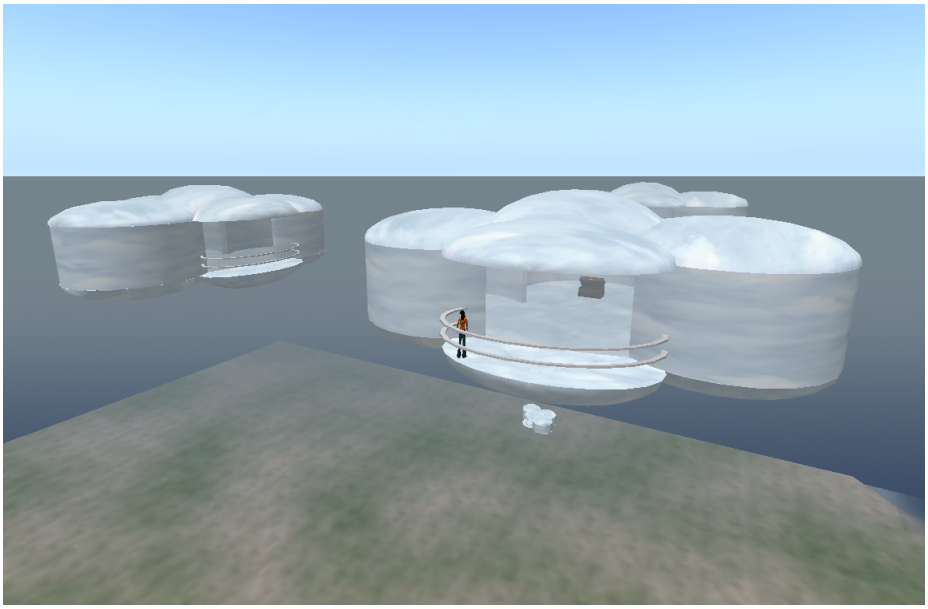

Figure 1. A View of the Cloud Objects

There are two main pieces to the cloud computing service. The first is a main cloud where the avatar can access all of the general functions of the service. The second piece is the set of all clouds that contain the actual resources that belong to each of the VO's.

In this paper is a description of a service that solves the problem of authentication in a virtual environment. Section two discusses two previous efforts toward this type of authentication, while section three gives a general overview of the functionality of the cloud computing service. In the fourth section, specific implementation details and code samples are shown. Finally, section five draws some conclusions from this work.

PREVIOUS WORK

The authentication and authorization challenge exist even without the involvement of virtual worlds. One technique commonly used in grid computing infrastructures is the Virtual Organization Membership Service (VOMS). A virtual organization is "a collection of people in the same administrative domain" [9]. The VOMS is used for "managing authorization data within multi-institutional collaborations" involving grid applications [14]. Using the VOMS system, an administrator sets up a database connecting specific users to their roles and privileges in the Virtual Organization (VO). When a user sends a request to access the VO's resources, the VOMS Server is accessed and the privileges are checked. If an unauthorized user attempts to access the VO's resources, they are denied access because they have no privileges in the database.

Second Life is often used as a learning tool. Some universities have classes that only meet in Second Life, with a class project requiring students to work together to build a complex Second Life object [12]. Many universities, even those that don't use Second Life, also use tools like Moodle and Blackboard in order to facilitate the passing of information between teachers and students. Sloodle allows for the combination of Moodle with Second Life so that students inside the virtual world can access their assignments and grades [13]. This type of application requires some technique of authentication and authorization because the avatar needs to be linked to his real world identity.

PROPOSED CLOUD SERVICE

Main Cloud

The first time an avatar interacts with the main cloud, he has four options to choose from (as shown in Figure 2): an option to search, to list the current clouds, to list the current groups, and to register. The search option (1) will allow the avatar to search any available clouds, which represent the resources for all of the virtual organizations, for the specified tag. If any are found, it will return a list of matches and the option to get more information about each one. If an avatar wishes to see a list of all clouds, he would select ListClouds (2), which would return a list of all of the clouds that have been created. If the avatar is interested in one of the displayed clouds, he can either choose to view more information about it or to teleport to its location in order to view the resources. Similarly, to view a list of all created groups, ListGroups (3) can be used.

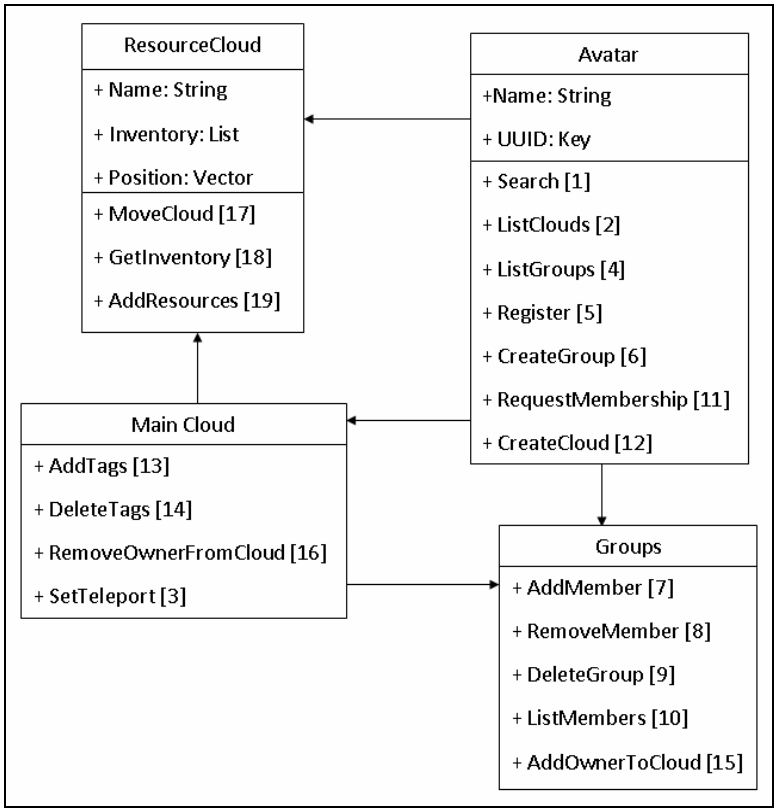

Figure 2. Avatar Functions

In order to access the most important functions of the cloud, an avatar must Register (4). The registration process is very straightforward for the avatar, requiring nothing more than for the option to be selected. After registering, the avatar can create a group (5) for his specific VO, or request membership into an existing VO (10). Once the group is created, the avatar can add new members (6), remove members from the group (7), and delete the group (8). Additionally, anyone can view a list of the members of a particular group (9).

The most important action that a registered avatar can take is to create a new resource cloud (11). This function will allow a separate cloud object to be created so that the avatar will have a place to store the resources that he wishes to share. Tags describing the resources that the cloud is to contain can be added by the cloud's creator when it is formed, or by anyone that accesses the cloud after it is formed (12). This tagging ability adds a social networking element, since avatars are able to discover common interests and new resources using the clouds. Unsatisfactory tags can be deleted at any time using the DeleteTag option (13).

If a VO's resources are to be private, the avatar can use the AddOwnerToCloud (16) function to connect a group that he has founded to the cloud. When a cloud has a group owner, only members of that particular group will be able to access the cloud's resources. If an avatar that is not a member of the group attempts to access the resources, he will be prompted with an option to request membership into the group, and the founder of the group will be able to screen members as necessary.

If a private resource is to be made public, the founder of the group that owns the cloud can use RemoveOwnerFromCloud (14) to remove the private restrictions. After this function is called, anyone will be able to access the resources within the cloud.

Resource Cloud

A resource cloud does not have the same functions as the main cloud. When an avatar creates a resource cloud, he has the option to move it around the region. This will allow one virtual organization to own multiple clouds, and organize them together in one particular section of the region. The main function of this cloud is for an avatar to access the resources that are stored inside. If the cloud is public, or if the cloud is private and the user has the proper privileges, touching the cloud will create a folder containing the resources in the avatar's inventory. If the avatar attempts to access a private cloud and does not have the proper membership privileges, he is prompted with an option to request membership into the group. If membership is requested, a message is sent to the founder of the group.

Resources

Any avatar with the proper privileges can add new resources to the resource cloud. These resources can be Second Life notecards, scripts, or even other objects. To add a resource, the avatar can drag it out of his inventory and onto the resource cloud. Some examples of resources that can be made accessible through the cloud service are sensor data, computing resources, visualization resources, or any other applications that use a Web service as an interface.

Figure 3. Second Life Object Accessing Atmospheric Sensor Data

Figure 3 shows a Second Life object that receives information from sensors that output weather information. In one mode of the object, information regarding the temperature, wind direction and speed, humidity, pressure, and other atmospheric conditions are shown. In another mode of the object, the user types a zip code and the information from the sensors is parsed to give less detailed weather information for that area. This object is an example of a resource that would be stored in a public cloud. Access to the object does not need to be restricted based on user identity, but it would benefit from the ability to be tagged and represented as a social networking entity.

Another example is the Condor object shown in Figure 4. Condor is a scheduling system used with computing resources in order to run high performance computing jobs. The Condor object used in Second Life allows the avatar to access Clemson University's Condor queue and to view information about jobs and resources (represented by the blocks). This type of object

would be stored in a private cloud, because access to the Condor queue would need to be authorized. In order to use this object with the cloud service, the manager of the Condor system at Clemson University would create a group, claim ownership of the cloud that stores the resource, and then add the necessary avatars as members in the group. This way, any avatars that are not members of the group will not be able to manipulate Clemson University's Condor queue.

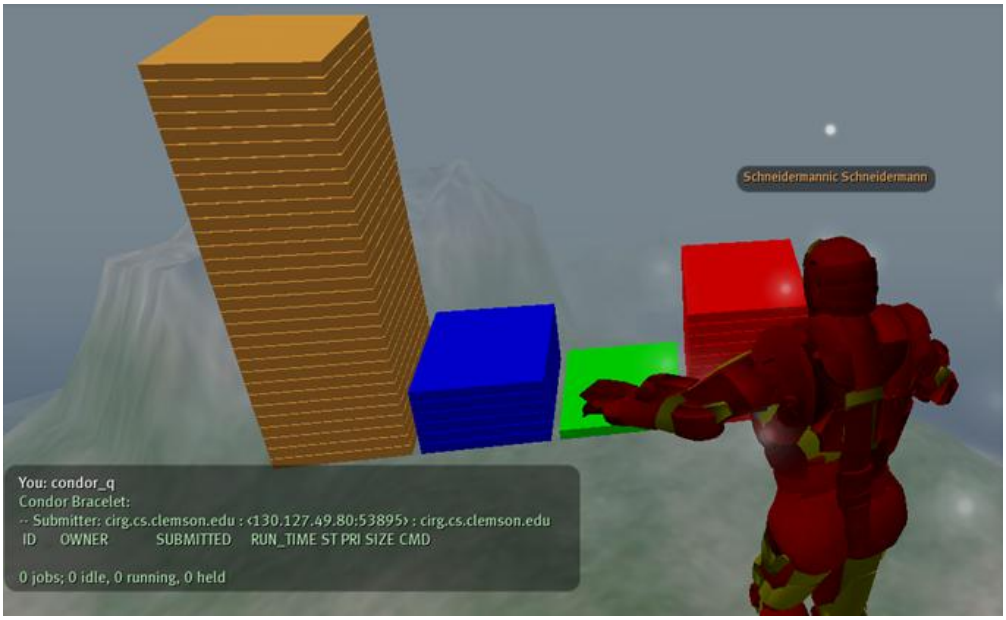

Figure 4. Object Accessing Real World Condor Resources

IMPLEMENTATION

In Second Life, when an avatar wishes to interact with an object, there are two ways to do so. The first is to physically touch the object, and the second is to use special keywords on the chat channel that the object has been scripted to recognize. The object described in this paper uses both of these methods.

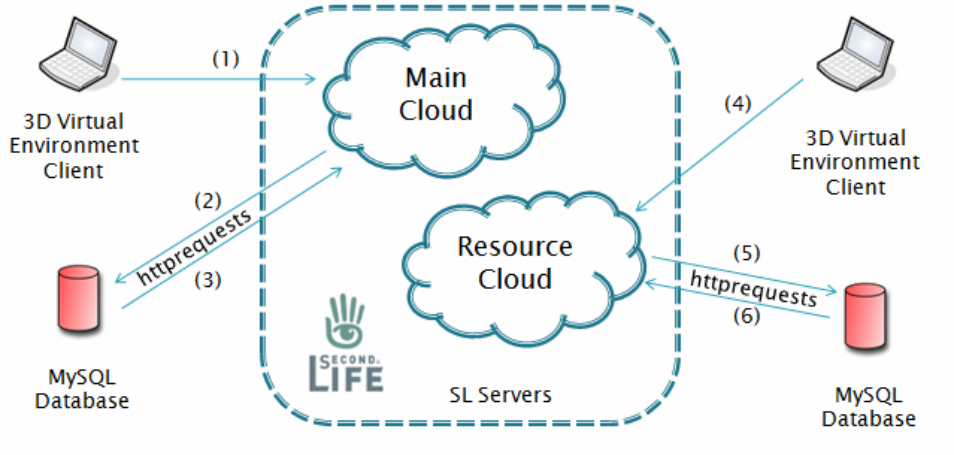

Figure 5. System Interaction

The Second Life scripts are written in Linden Scripting Language (LSL), and all the information used for the cloud service is stored in a MySQL database, as shown in Figure 6. This database is accessed using a PHP script and HTTPRequests. When

an avatar registers, his name and unique UUID are stored in the Avatars table. This is necessary because it allows the avatar to join and create groups and to create clouds.

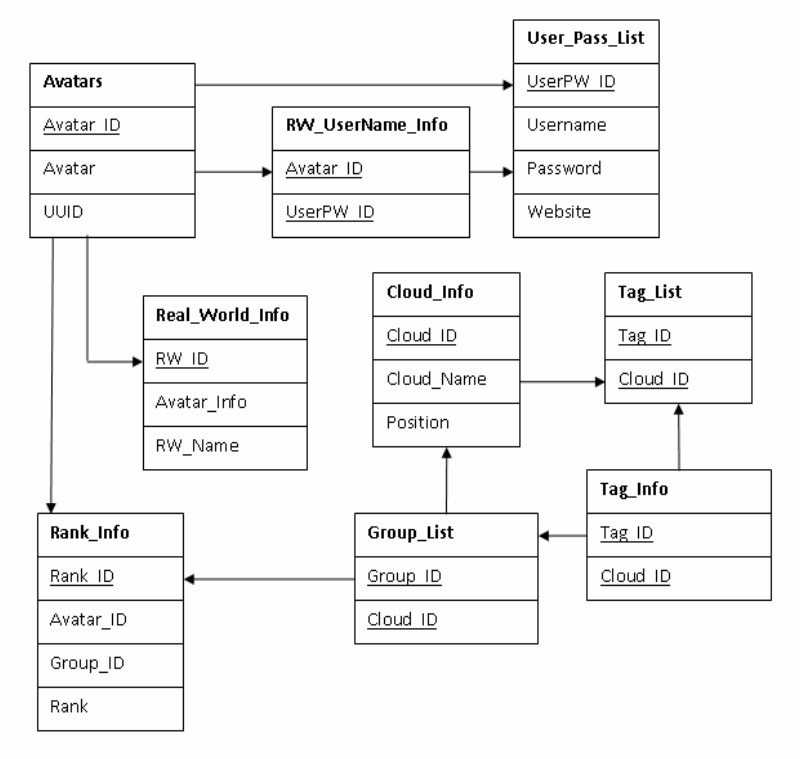

Figure 6. ER Diagram

New clouds are created using the in-world chat window. The format for the user is "Create [CloudName] [tag1,tag2,tag3]", with [CloudName] being the name of the cloud, and [tag1,tag2,tag3] being any tags that should be associated with the cloud and its resources. The tag list is optional at the time of cloud creation, and it can be edited at any time after the cloud has been created. Once this information is submitted, the script in the main cloud parses the information and performs an HTTPRequest, as shown in Figure 7, to the PHP script.

```
else if(llList2String(currentResponse,0) == "CreateCloud"){
  target = (vector)llList2String(currentResponse,1);
  llRezObject("RezzedCloud",target, ZERO_VECTOR, ZERO_ROTATION, 0);
  llDialog(userid, "Cloud created. Feel free to move it.",["OK"],channel);
  llMessageLinked(LINK_SET,7,(string)target,"");
} ...
if(llList2String(parsedCommand, 0) == "Create"){
  cloudName = llList2String(parsedCommand, 1);
  newTags = llList2String(parsedCommand,2);
  requestID = llHTTPRequest(webAddress, [HTTP_METHOD,"POST",HTTP_MIMETYPE,
     "application/x-www-form-urlencoded"], "task=CreateCloud&position=" +
     (string)position + "&cloudName=" + cloudName + "&tags=" + newTags);
}
```

Figure 7. Cloud Creation – LSL

When the PHP script receives the request, it inserts the information into the database and returns the cloud's position. The LSL script then receives the position and creates a new cloud in-world at those coordinates. The resource cloud is equipped with a script that automatically updates the position in the database when the cloud is moved, so the avatar is free to move the cloud to a different location.

The chat window is also used when a group is created, with the format being "CreateGroup [groupName]", where [groupName] is the name of the group. Again, after the information is passed in from the user, an HTTPRequest to the PHP script is initiated and the information is stored in the database, as shown in Figure 8. When the Rank information is entered, the person creating the group is called the founder. This gives that avatar all of the rights of the group, including adding and deleting members and claiming ownership of clouds.

```
$query  = "Select Avatar_ID from Avatars where Avatar = '$name'";
$result = mysql_query($query);
while($row = mysql_fetch_array($result, MYSQL_ASSOC)){
   $avatarID = $row['Avatar_ID'];
}
if($avatarID == ""){
   echo "AvatarError";
}
else{
   $query = "Select Group_ID from Group_Info where Group_Name =
      '$groupName'";
   $result = mysql_query($query);
   while($row=mysql_fetch_array($result, MYSQL_ASSOC)){
      $groupID = $row['Group_ID'];
}
                                ...
$query = "Insert into Group_Info (Avatar_ID,Group_Name) values"
   . " ('$avatarID','$groupName')";
$result = mysql_query($query);
$query = "Select Group_ID from Group_Info where Group_Name =
   '$groupName'";
$result = mysql_query($query);
while($row = mysql_fetch_array($result, MYSQL_ASSOC)){
   $groupID = $row['Group_ID'];
}
$query = "Insert into Rank_Info (Avatar_ID,Group_ID,Rank) values"
   . " ('$avatarID','$groupID', 'Founder')";
$result = mysql_query($query);
echo "CreateGroup";
```

Figure 8. Group Creation - PHP

To claim ownership of a cloud, the founder must use the format "AddOwnerToCloud [groupName] [cloudName]", where [groupName] is the name of the group claiming ownership and [cloudName] is the name of the cloud. Once this is complete, the cloud has been made private, and any access to this cloud must be authorized.

In the resource cloud, the main function is to give the inventory to an avatar that requests it. If the cloud is private, the first step is to check the permissions of the avatar, as shown in Figure 9. An HTTPRequest is initiated, passing the cloud name and the avatar's name to the PHP script. The PHP script accesses the Rank_Info table and searches through the list of that group's members for the avatar's ID. If it is found, the message "Granted" is passed back to the LSL script. However, if it is not found, an error message is passed back. If the cloud is public, there will be no matching group ID for that cloud in the Rank_Info table, so there will be no need to search for the avatar's ID.

```
$query  = "Select Cloud_ID from Cloud_Info where Cloud_Name = '$cloudName'";
$result = mysql_query($query);
while($row = mysql_fetch_array($result, MYSQL_ASSOC)){
   $cloudID = $row['Cloud_ID'];
}
$query = "Select Group_ID from Group_List where Cloud_ID = '$cloudID'";
$result = mysql_query($query);
while($row = mysql_fetch_array($result, MYSQL_ASSOC)){
   $groupID = $row['Group_ID'];
}
if($groupID != ""){
   $query = "Select Avatar_ID from Avatars where Avatar = '$avatarName'";
   $result = mysql_query($query);
   while($row = mysql_fetch_array($result, MYSQL_ASSOC)){
      $avatarID = $row['Avatar_ID'];
   }
   $query = "Select Rank_ID from Rank_Info where Group_ID = '$groupID'
      and Avatar_ID = '$avatarID'";
   $result = mysql_query($query);
   if(mysql_num_rows($result) > 0) {
      echo "Granted";
   }
   else{
      $query = "Select Group_Name from Group_Info where Group_ID = '$groupID'";
      $result = mysql_query($query);
      while($row = mysql_fetch_array($result,MYSQL_ASSOC)){
         $groupName = $row['Group_Name'];
      }
      echo "PermissionError.." . $groupName;
   }
}
else{
   echo "Granted";
}
```

Figure 9. CheckPermissions – PHP

When the script receives the HTTP response, it will either give the inventory to the avatar, or give the avatar an option to request membership into the group. If the avatar decides to request membership, an instant message containing the avatar's name is sent to the founder asking for approval. The founder can then communicate with the avatar and complete whatever screening process is necessary. If the founder decides to admit the avatar, he can do so by using the chat format "AddMember [memberName] [groupName]", where [memberName] is the avatar to be added and [groupName] is the name of the group.

Avatars can also request membership proactively, by using the following chat format: "RequestMembership [groupName]", where [groupName] is the name of the group.

To make a cloud public again, the founder must use the format "RemoveOwnerFromCloud [groupName] [cloudName]", where [groupName] is the name of the group claiming ownership and [cloudName] is the name of the cloud. Once this is complete, the cloud has been made public, and there will be no further authorization of access.

CONCLUSIONS

Authorization in virtual worlds is a difficult issue to address. The owner of the restricted resources cannot expect a stranger to give up any important information about himself because no one would be willing to use the object. However, the owner has a reasonable expectation of keeping his resources safe from threats. After designing this cloud service, we have learned that cloud computing is feasible in virtual environments because resources can be attached to a cloud and access can be managed based on the avatar's identity. Also, because of the ability to tag clouds and resources, the cloud becomes a social networking entity. Future work in this area will involve maintaining the mapping between avatar and real life identity.

REFERENCES

1. Benford, Steve, Chris Greenhalgh, Tom Rodden, James Pycock. (July 2001) Collaborative Virtual Environments, *Communications of the ACM*, 79-85.
2. Bray, David A., Benn R. Konsynski. (November 2001) Virtual Worlds: Multi-Disciplinary Research Opportunities, *SESSION: Research Contributions*, 17-25.
3. Hendaoui, Adel, Moez Limayem, and Craig W. Thompson. (January 2008) 3D Social Virtual Worlds: Research Issues and Challenges, *IEEE Internet Computing*, 88-92.
4. Jaquet-Chiffelle, David-Olivier. (2002) Authentication and/or Identification Through the Virtual World, *Stork Cryptography Workshop*.
5. Kemp, Jeremy and Daniel Livingstone. (2006) Putting A Second Life "Metaverse" Skin on Learning Management Systems, *Second Life Community Conference*.
6. Kumar, Sanjeev, Jatin Chhugani, Changkyu kim, Daehyun Kim, Anthony Nguyen, Pradeep Dubey, Christian Bienia, Youngmin Kim. (August 2008) Second Life and the New Generation of Virtual Worlds, *Computer*, 46-53.
7. Lee, Chia Yao and Matthew Warren. (2007) Security Issues within Virtual Worlds such as Second Life, *Australian Information Security Management Conference*.
8. Mennecke, Brian, Roche, Edward M., Bray, David A., Konsynski, Benn, Lester, John, Rowe, Michael and Townsend, Anthony M. (2007) Second Life and Other Virtual Worlds: A Roadmap for Research, *28th International Conference on Information Systems (ICIS)*.
9. Niinimaki, Marko, John White, Wim Som de Cerff, Joni Hahkala, Tapio Niemi, Mikko Pitkanen. (2004) Using Virtual Organizations Membership System with EDG's Grid Security and Database Access, ***15th International Workshop on Database and Expert Systems Applications***.
10. Scheffler, Martin, Jan P. Springer, and Bernd Froehlich. (2008) Object-Capability Security in Virtual Environments, *IEEE Virtual Reality Conference*.
11. Second Life. http://www.secondlife.com.
12. Second Life in Education. http://sleducation.wikispaces.com/educationaluses.
13. Sloodle. http://www.sloodle.org/moodle.
14. Virtual Organization Membership Service. http://www.globus.org/grid_software/security/voms.php.
15. Wacker, Arno, Gregor Schiele, Sebastian Schuster, and Torben Weis. (2008) Towards an Authentication Service for Peer-to-Peer based Massively Multiuser Virtual Environments, *IEEE Virtual Reality Conference*.