# Membership Management based on a Hierarchical Ring for Large Grid Environments

**Tae-Wan Gu*, Seong-Jun Hong*, Saangyong Uhmn*, and Kwang-Mo Lee***

**Abstract:** Grid environments provide the mechanism to share heterogeneous resources among nodes. Because of the similarity between grid environments and P2P networks, the structures of P2P networks can be adapted to enhance scalability and efficiency in deployment and to search for services. In this paper, we present a membership management based on a hierarchical ring which constructs P2P-like Grid environments. The proposed approach uses only a limited number of connections, reducing communication cost. Also, it only keeps local information for membership, which leads to a further reduction in management cost. This paper analyzes the performance of the approach by simulation and compares it with other approaches.

**Keywords:** *P2P, Membership Overlay, Membership Management, Hierarchical Ring*

## 1. Introduction

Grid environments provide the mechanism to share heterogeneous resources for processing jobs [1]. As the number of hosts in a Grid running complex applications increases, it should be constructed in a decentralized manner to avoid bottleneck. In this regard, the P2P model has the potential to improve the scalability of Grid environments. That is, the intrinsic capability of the decentralized structure of P2P networks can be adapted to solve the scalability problem of the Grid environments. Moreover, considering the deployment of information services and the search for specific resources, the P2P model can be very effective for many Grid environments [1].

However, the membership management currently deployed in Grid environments is provided in a centralized fashion as in the Open Grid Service Architecture (OGSA)[2] and Web Service Resource Framework (WSRF)[3], because most of the resources are owned by research institutes or public organizations. The centralized architecture would not be adequate because of the scalability problem in large environments. Thus, there have been many attempts to utilize the approaches for P2P networks in Grid environments [4–7].

In this paper, we propose a hierarchical ring-based membership management approach called *HRing*. It can be considered as an application of the P2P model to maintain membership in Grid environments. It maintains only partial information about the membership at each node rather than all of it, which improves the scalability and reduces the management cost. In addition, its hierarchical structure reduces the network traffic as well as the convergence time.

Manuscript received October 16, 2006; accepted March 3, 2007.
This research was supported by Hallym University Research Fund, HRF-2000-45.
**Corresponding Author: Tae-Wan Gu**
* Dept. of Computer Engineering, Hallym University, Chuncheon, Korea (taewani,teferi,suhmn,kmlee@hallym.ac.kr)

The remainder of the paper is organized as follows. In Section 2, we present related works about membership management in Grid environments and P2P networks; in Sections 3 and 4, we describe our management approach; in Section 5, we analyze the performance of the proposed method, and present the experimental results in Section 6. Finally, we present our conclusion in Section 7.

## 2. Related Work

An example of the conventional centralized approach to services in Grid environments is Index Services used in the Globus Toolkit 3(GT3)[8]. There is one Index Service per virtual organization (VO). For large Grid environments, multiple Index Services can be constructed hierarchically. A similar approach is used in the WSRF-based Globus Toolkit 4. However, this type of model provides little flexibility and is not suitable for large Grid environments. In Index Service, it generates large overhead when processing frequent user requests, which causes bottleneck [8].

There is a similarity between P2P networks and Grid environments in the sense that recent Grid environments have been constructed on a huge distributed model. Iamnitchi et al. discussed the similarity and also analyzed the potential of using the P2P model for Grids in [9]. In [10], Talia et al. compared the Grid and P2P networks and argued that these two systems will converge in terms of their concerns, as Grid scales and P2P networks address more sophisticated application requirements.

P2P networks can be broadly categorized into two classes: unstructured architecture [11, 12] and structured architecture [13–15]. The former approach allows members to join and leave the network freely without global overlay planning. The latter maintains highly structured overlays and utilizes Distributed Hash Tables (DHT) to process user requests. The latter approach is less flexible because it uses

a static centralized index [16].

An example of the unstructured approach, which uses the so-called heartbeat message at every node, is presented in [17]. This message is sent by each node and received by all others; it contains membership information known to each node. In this architecture, each node maintains a membership directory independent of those of the other nodes, which is one of the All-to-All (A2A) approaches. Therefore, there is significant overhead in maintaining the membership information [18].

In [16], a Gossip type is used for the membership service, which is similar to the work of [17]. Each node maintains a random set of neighbors named *partial view*. In addition, it sends its partial view to its neighbors. Upon receiving it, the neighbors can update their own partial views if they are different. Although this approach is successful in small scale networks with a high bandwidth, it can often take a considerable amount of time to converge to the stable state of the membership.

In [18], Zhou et al. propose a hierarchical approach that addresses the limitations of the two above-mentioned approaches. The membership service is provided by a membership hierarchy with a virtual tree structure. It is designed to deliver membership information effectively from a node on the upper level to those on the lower level. However, one of the limitations of this approach is that it has a single point of failure. If a node on the upper level leaves the network, it may cause a critical problem for the entire network.

## 3. Hierarchical Ring Membership Management

### 3.1 Overview

The membership is represented as a graph $G=<V, E>$ of connected rings, where $V$ is a set of nodes $\{v_1, v_2, \ldots, v_n\}$ in the membership, and $E=\{<v_i, v_j>|v_i, v_j \in V\}$ is a set of connections between two nodes, where $|E|=m$. Each node $v_i$ can have up to 3 out of 4 kinds of connections to other nodes which are $out_{main}$, $out_{sub}$, $in_{main}$ and $in_{sub}$, respectively. As the name implies, $out_{main}$ and $out_{sub}$ mean the outgoing connections and $in_{main}$ and $in_{sub}$ are the incoming ones. Every node $v_i$ in the membership has $out_{main}$ and $in_{main}$, and possibly one of the others, i.e. $out_{sub}$ or $in_{sub}$, but not both of them. Fig. 1 shows an example of the graph for the membership.

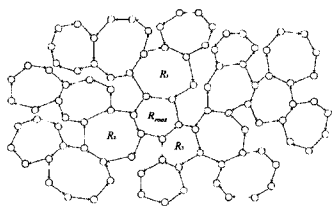


**Fig. 1.** An example graph of the HRing

As shown in Fig. 2, there is a special ring, $R_{root}$, which is the initial ring of the membership and its level is 1. The ring $R_{root}$ is connected with several other rings of level 2, and so on. Also, we define a ring on level $i$ as the $R_{parent}$ of a ring on level $i+1$, and a ring on level $i+1$ as the $R_{child}$ of a ring on level $i$. Fig. 2 illustrates this relationship in detail.

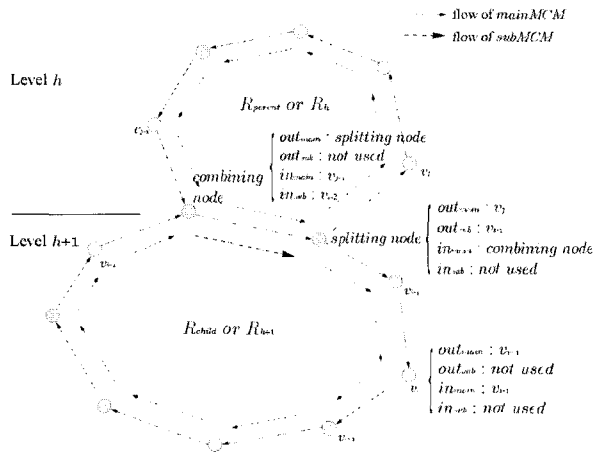



**Fig. 2.** The relationship between two rings in the HRing

In addition, we will use NEXT($v_i$) to refer to $v_i$+1, the next node of $v_i$ in the path of the MCM transmission, and PREV($v_i$) to refer to $v_{i-1}$, the previous node of $v_i$ in the same ring. The *splitting node* is a starting node and the *combining node* is a finishing node of the ring $R_{child}$. We will use $NEXT_{sub}(v_i)$ to refer to the next node connected by $out_{main}$ of $v_i$, and $PREV_{sub}(v_i)$ for the previous node in $R_{child}$ connected to $v_i$ by $in_{sub}$.

The message called MCM (Membership Control Message) is generated at each node in the membership and sent through the outgoing connection. All nodes transmit their MCM through $out_{main}$. In addition, the splitting node generates an MCM for $R_{child}$ and transmits it through $out_{sub}$. The MCM through $out_{main}$ is called $MCM_{main}$, and the MCM through $out_{sub}$ is called $MCM_{sub}$.

The other important element of the management is the threshold value $k$, to create $R_{child}$. When a node $v_{new}$ tries to join the membership, it contacts a node $v_{contact}$. We assume that there is a service to find $v_{contact}$ and that $v_{contact}$ has only $in_{main}$ and $out_{main}$. The new node measures the Round Trip Time, $RTT_{new,contact}$, to $v_{contact}$ and compares it with $k$ from $v_{contact}$ to determine whether it joins the ring, $R_h$, or a new ring, $R_{h+1}$. The ring will be created if $RTT_{new,contact}$ is greater than $k$.

Every node maintains the information about the members in the same ring only. In addition, it uses $out_{main}$ to transmit and $in_{main}$ to receive membership information, except the splitting node and the combining node. The splitting node transmits $MCM_{sub}$ through $out_{sub}$ and the combining node receives $MCM_{sub}$ through $in_{sub}$. Because of the usage of two connections and local information, this management provides efficiency and scalability for the Grids.

### 3.2 Design of the Node

Each node, $v_i$, has the following 3 components.

- ***Connection Handler* (*CH*)** : processes join/Leave operations. When a new node participates in the membership, it measures $RTT_{new,contact}$ and executes a Join operation. If a node wants to leave the membership, it executes a Leave operation.
- ***Message Handler* (*MH*)** : sends or receives MCMs. When a node participates in the membership, it generates an MCM and transmits it through $out_{main}$, and if it is a splitting node, it also generates $MCM_{sub}$ and sends it through $out_{sub}$. The MCM is transmitted asynchronously using UDP. Also, $MH(v_i)$ verifies that an MCM is received correctly through $in_{main}$ , and also through $in_{sub}$ if it is a combining node. It compares the received MCM with its own, and modifies its own if necessary.
- ***Event Handler* (*EH*)** : There are several kinds of events which can occur at the node, such as the joining or leaving of a node, membership graph adjustment, and missing an MCM(*NO_MCM*), and so on. If one of these events occurs at a node $v_i$, $EH(v_i)$ multicasts the event to all the nodes in its own ring and *SplitNode* in $R_{root}$ which is a splitting node for the branch containing $v_i$.

Also, each node, $v_i$, keeps the following information.

- **Depth of the ring, Depth($v_i$)**: indicates the depth of the ring to which the node $v_i$ belongs. This value is used to keep the structure balanced. If $v_i$ receives an MCM or hierarchy adjustment message, each node updates it with the received value.
- **A node in $R_{root}$, *SplitNode***: indicates the splitting node in $R_{root}$ for the branch of $v_i$ to notify the events.

For sending and receiving an MCM, the message timeout, *MCM_TIMEOUT*, is defined to detect an event, *NO_MCM*. If a node $v_i$ does not receive an MCM through $in_{main}$ within *MCM_TIMEOUT*, it initiates an event *NO_MCM*. We assume that the value of *MCM_TIMEOUT* is configured by the administrator based on the network statistics.

### 3.3 Membership Control Message

MCM is a control message for sharing the membership information among the nodes in the same ring only and is processed by $MH(v_i)$. It is sent by every node in a ring at a predefined time interval $t$ through $out_{main}$. It contains the following information. First, it contains the identification of the splitting node (*SplitNode*) in $R_{root}$ which is used by $EH(v_i)$ to notify the events. Second, it contains the depth of the ring of $v_i$, Depth($v_i$). Third, there is a list of nodes (*ListOfNodes*) in the ring. When a new node participates in the membership, the contact node appends the new node to this list. Fourth, it has MCM flag (*flag*) for the splitting node to distinguish between $MCM_{main}$ and $MCM_{sub}$. If it is TRUE, it means that the MCM is an $MCM_{main}$. Otherwise it means that it is an $MCM_{sub}$. Finally, there is *Split Threshold Value*($k$) to determine whether a new node is included in the existing ring or in the new ring. When a new node participates in the membership, the information about the node must be included in an MCM. However, if every node in the membership is connected by one ring, the length of an MCM can be so long that it takes a long time for a node to transmit it. It also takes a long time for messages to traverse the entire network. Thus, it is necessary to define a limit to ensure that the MCMs traverse the network in a reasonable time. To keep the length of an MCM and the time taken by the messages to traverse the network short enough, we define the split threshold value $k$. To determine the split threshold value $k$, every new node measures $RTT_{new,contact}$ to $v_{contact}$ and sends it to $v_{contact}$. The $v_{contact}$ calculates the total RTT ($RTT_{total}$) by Eq. (1).

$$RTT_{total} = \left( \prod_{i=1}^{l} RTT_{i,(i+1)\%l} \right) \quad (1)$$

where $l$ is the number of nodes in the ring and % is a modulus operator. Also, without loss of generality, we assume that $RTT_{i,j} > 1$. Then, the split threshold value $k$ is calculated by Eq. (2).

$$k = \sqrt[l]{RTT_{total}} \quad (2)$$

The value $k$ reflects the number of nodes in a ring and their *RTT*. When $v_{new}$ receives a response for a Join operation from $v_{contact}$, it calculates $RTT_{new,contact}$ and compares it with $k$. If $k$ is greater than $RTT_{new,contact}$, $v_{new}$ sends $v_{contact}$ a request *REQ_CONNECT* and $v_{contact}$ adds $v_{new}$ to its ring. If $k \leq RTT_{new,contact}$, $v_{contact}$ becomes a splitting node and creates a new ring by connecting $v_{new}$ with $out_{main}$ and adjusting the necessary connections.

We define $OutMCM_t$ as the MCM transmitted through $out_{main}$ at time interval $t$ and $InMCM_t$ as the MCM received through $in_{main}$. The $MH(v_i)$ compares $OutMCM_t$ and $InMCM_t$. If they are the same, the ring is not changed so there is nothing to be done. If not, it implies that the membership has been changed and $MH(v_i)$ updates $OutMCM_t$ as $InMCM_t$.

## 4. Construction of a Membership Ring

We assume that there is a service to ask for a contact node, $v_{contact}$. If there is no node in the membership and a new node sends a request, the service designates the new node as a contact node. In such a case, the new node builds a membership graph with itself only.

### 4.1 Join operation

When a new node tries to participate in the membership, it sends *REQ_JOIN* to $v_{contact}$ to initiate a Join operation. Based on the condition of $k$ and $RTT_{new,contact}$, there are two different sequences of steps to be performed.

If $k$ is greater than $RTT_{new,contact}$, those steps depicted by a sequence diagram in Fig. 3 are executed to put $v_{new}$ into the current ring of $v_{contact}$. After the process has been completed,

$v_{new}$ set its Depth($v_{new}$) to Depth($v_{contact}$).

If $k$ is less than or equal to $RTT_{new,contact}$, the steps

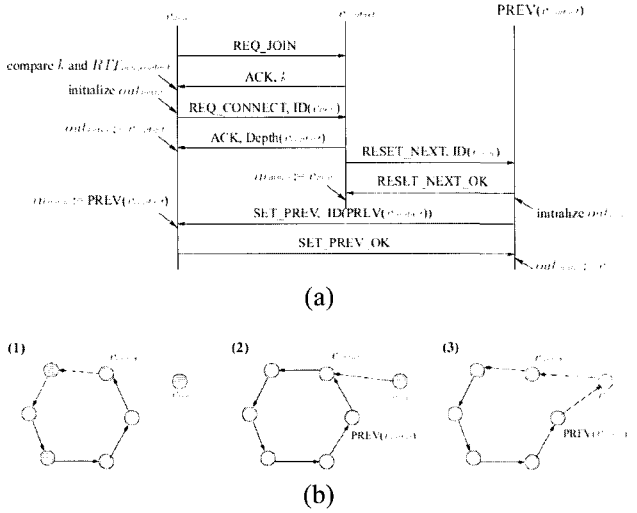



**Fig. 3.** (a) Join operation when $k > RTT_{new,contact}$
(b) Concept of Join operation when $k > RTT_{new,contact}$

depicted in Fig. 4 are executed. A new ring is created which contains $v_{new}$, $v_{contact}$ and PREV($v_{contact}$). In this case, the node $v_{new}$ sets its Depth($v_{new}$) to Depth($v_{contact}$) + 1. After joining the membership, the node $v_{new}$ sends Depth($v_{new}$) to *SplitNode* when it receives the first MCM.

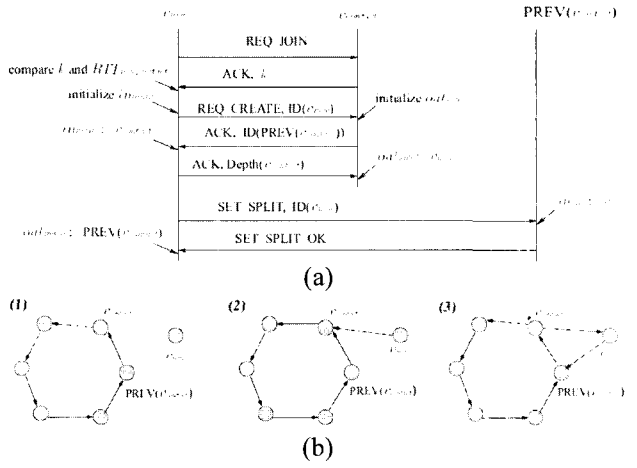



**Fig. 4.** (a) Join operation when $k \leq RTT_{new,contact}$
(b) Concept of Join operation when $k \leq RTT_{new,contact}$

## 4.2 Leave operation

A node $v_{leave}$ initiates a Leave operation by notifying the other nodes in the same ring. There are three cases for a Leave operation based on the condition of the node $v_{leave}$: the splitting node, the combining node, or other nodes with $in_{main}$ and $out_{main}$ only.

**Leave operation: Ordinary Nodes** If $v_{leave}$ is neither a splitting node nor a combining node, only the connections for PREV($v_{leave}$) and NEXT($v_{leave}$) must be adjusted to point each other. The diagram in Fig. 5 illustrates the steps for this case.

**Leave operation: The Splitting Node and the Combining Node** If a splitting node or a combining node leaves the membership, connections for the $R_{child}$ must be adjusted properly to maintain the structure. However, all three nodes connected to $v_{leave}$ may have $out_{sub}$ or $in_{sub}$ already. In this case, any one of them cannot be assigned to the position of $v_{leave}$. To cope with this situation, a Leave operation must begin with the search for the candidate $v_{candidate}$ for $v_{leave}$.

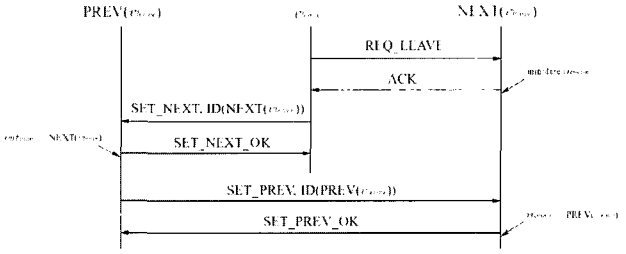



**Fig. 5.** Leave operation for ordinary node

Any node which receives REQ_LEAVE message through the incoming connection forwards it through its outgoing connection and vice versa to propagate the message to find the candidate node. For example, $NEXT_{sub}(v_{leave})$(or NEXT($v_{leave}$)) receives REQ_LEAVE from $v_{leave}$ through its $in_{main}$ connections and forwards it through $out_{main}$. PREV($v_{leave}$) receives the message through $out_{main}$ and forwards it through $in_{main}$. If any node, $v_{candidate}$, without $out_{sub}$ and $in_{sub}$ receives the message, it sends REQ_LEAVE_OK message with its ID to $v_{leave}$. Upon receiving REQ_LEAVE_OK, $v_{leave}$ responds with information about its connections to begin the substitution process. After the node $v_{candidate}$ receives them, it first sends messages to PREV($v_{candidate}$) and NEXT($v_{candidate}$). This means that the $v_{candidate}$ leaves from its local ring. Then, the $v_{candidate}$ sends its information to PREV($v_{leave}$), NEXT($v_{leave}$) and $PREV_{sub}(v_{leave})$ (or $NEXT_{sub}(v_{leave})$) respectively. This process is depicted as a diagram in Fig. 6. When PREV($v_{candidate}$) and NEXT($v_{candidate}$) receive a message from $v_{candidate}$, they adjust their connections if PREV($v_{candidate}$) is not a splitting node and NEXT($v_{candidate}$) is not a combining node. If PREV($v_{candidate}$) is a splitting node and NEXT($v_{candidate}$) is a combining node, $NEXT_{sub}$(PREV($v_{candidate}$)) and $PREV_{sub}$(NEXT($v_{candidate}$)) are set to nil.

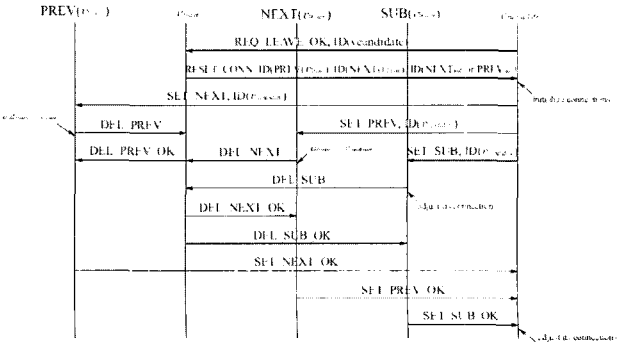



**Fig. 6.** Leave operation for the splitting node and the combining node

**Depth** If a node leaves the membership, it may decrease the depth of the structure. It may occur if $v_{leave}$ or $v_{candidate}$ is the only node in a ring except the splitting node and the combining node. In either case, $v_{leave}$ and $v_{candidate}$ notify *SplitNode* in $R_{root}$ of the change of the structure, which may initiate a hierarchy adjustment process.

### 4.4 Hierarchy Adjustments

The proposed membership management maintains a tree-like structure of rings for MCM transmission. That is, it provides the optimum performance if the whole structure is balanced with the minimum depth variation. However, if the nodes participate in or leave the membership in arbitrary order and place, this property cannot be guaranteed. Thus, a mechanism is necessary to detect the imbalance and adjust the structure for the minimum depth variation. For the possible minimum depth variation, the rings may be broken and rebuilt as a new set of rings. However, it might require many communication messages and computations, which would be inappropriate. So, we only reassign the role of $R_{root}$ to another ring to reduce the depth variation and improve the performance. The process consists of two steps: detection of the imbalance and adjustment of the structure.

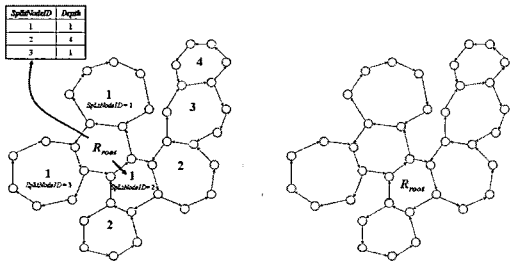


**Fig. 7.** Hierarchy adjustment

First, the process to detect the imbalance of the structure is carried out by Join and Leave operations. When a node participates in or leaves the membership, the depth of the structure may change. When this happens, *SplitNode* in $R_{root}$ will be notified. If the difference of depth between any pair of branches is greater than 2, the adjustment process may be performed. However, if it is performed whenever necessary, it may cause traffic and computation overhead. So, the adjustment process can be initiated by an administrator. The node with the greatest depth, *SplitNode*, sends *SET_ROOT* through $out_{sub}$. Upon receiving *SET_ROOT*, each node decreases its depth and forwards it through $out_{main}$. If it is a splitting node, it sends *ADD_DEPTH_DEC* with its ID through $out_{sub}$. Any node receiving ADD_DEPTH_DEC decreases its depth and adjusts *SplitNode* in MCM, and then forwards it through $out_{main}$ and $out_{sub}$, if it is a splitting node.

An example of hierarchy adjustment is depicted in Fig. 7.

## 5. Performance Analysis

We compared the performance of the HRing with two other existing managing approaches by simulation. The first one, All-to-All (A2A), assumes that every node knows about all the other nodes in the membership. In the second one, Gossip-based membership management, each node has just a partial view of its neighbors. Table 1 shows the performance evaluation metrics.

**Table 1.** Performance Evaluation Metric

| | |
|---|---|
| **Communication Cost** | Network bandwidth availability or consumption during execution of the management. |
| **Management Cost** | Size of storage and frequency of information updates to maintain the membership. |
| **Convergence Time** | The time taken to disseminate the information to all other nodes when changes occur. |

### 5.1 Communication Cost

In A2A, each node multicasts membership information to every other node. Thus, the total amount of network bandwidth consumption is $O(n^2)$.

In the Gossip-based membership management (SCAMP), each gossip message contains only a partial view of the whole membership and each node accumulates a global view incrementally by exchanging messages with randomly chosen neighbors. If there are $n$ nodes and $s$ is a partial view size, each node gossips to $\log(n) + s$ on average [16], which makes the bandwidth consumption $O(n \log n)$.

In the case of the HRing, every node in the membership sends only one message, with the exception of the splitting node, which sends two messages. So, its bandwidth consumption is $O(n)$.

### 5.2 Management Cost

Management cost ($C_{management}$) consists of $C_{storage}$ and $C_{operation}$. $C_{storage}$ is the amount of storage for membership information and $C_{operation}$ is the number of operations required to update the information. Thus, $C_{storage}$ of A2A is proportional to $(n - 1)$ and $C_{operation}$ is $n - 1$ because each node has to maintain the information of all the other nodes. In the case of Gossip-based management, the size of the partial view is $m$ which is predefined and fixed. So, $C_{storage}$ and $C_{operation}$ can be considered as $O(1)$. In the case of the HRing, similar to the Gossip approach, it just maintains the information about its own ring. However, because the HRing does not maintain membership information, $C_{storage}$ and $C_{operation}$ can be considered as $O(1)$.

### 5.3 Convergence Time

We define the convergence time as the time taken to disseminate the change of a network to all other nodes. The convergence time of A2A is $O(1)$ because the information can be disseminated by one multicast. In Gossip-based management, assuming that the partial view sizes are all

roughly of size $(c + 1) \log(n)$ and $c$ is a design parameter, the number of forwarding steps before a subscription is kept is roughly $(c + 1) \log(n)$[16]. So, the convergence time of Gossip is $O(\log n)$. In our management, each node notifies the changes to $R_{root}$ , and it then sends the information to the lower level rings. Let $r$ be the average number of children rings in the structure. It depends on the split threshold value, $k$. Then, $r$ is an integer and $r \geq 1$. The average depth of the structure is $\log_r n \leq \log n$. So, the time taken to disseminate the changes is $O(\log n)$ at worst.

## 6. Experiments and Results

### 6.1 Configuration

We compared the performance of our management with A2A, Gossip (SCAMP)[16] and Hierarchical Tree [18] by simulation. The simulation code is written in Java. The number of nodes for the simulation is up to 100,000. The number of nodes for the hierarchy adjustment is 1,000–5,000.We assume that there is no traffic over the network except the messages.

### 6.2 Experimental Results

First, we simulated the effect of $r$ and the hierarchy adjustment of the proposed method. The $r$ has influenced the number of children rings of each ring in HRing. As each ring has more children, the depth of the tree will be shallower. Fig. 8 shows the relationship between $r$ and the convergence time: the greater $r$ is, the shorter the convergence time.

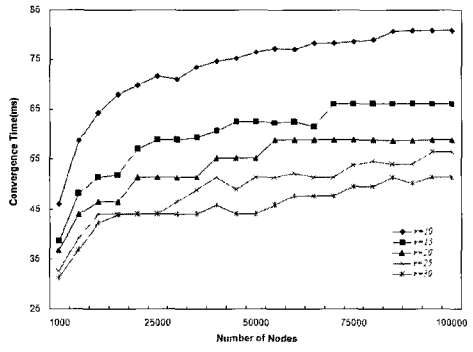



**Fig. 8.** Convergence time as size of local ring increase

Fig. 9 shows the convergence time after the tree hierarchy has been adjusted. From Fig. 9, we can assess the effect of the hierarchy adjustment on performance. The hierarchy adjustment was performed 5 times. The points in the figure indicate the average convergence time between the consecutive hierarchy adjustments. That is, those values at the first adjustment show the average convergence time between the initial ring and the first adjustment. After the first adjustment, the convergence time decreased, which means that the management performance had improved. We also observed that the differences between the second and third adjustment were not so great, which means that the hierarchy had become well balanced after the first adjustment.

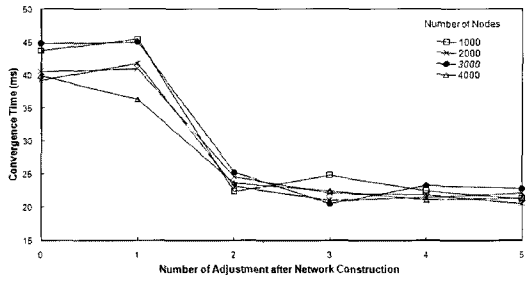



**Fig. 9.** Hierarchy adjustment

Fig. 10 shows the operating overhead, namely $C_{operation}$, on each node during the adjustment. We used Standard Deviation(STDEV) to measure the management cost, $C_{operation}$. This means that HRing was not affected by the number of nodes. In other words, the management cost in HRing may be uniform, regardless of the number of nodes.

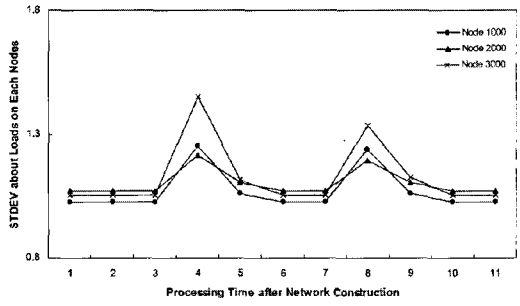



**Fig. 10.** Management cost of HRing

Fig. 10 shows how the hierarchy adjustment was performed twice at time 4 and time 8. As the number of nodes increased, $C_{operation}$ would increase. For this reason, the STDEV value was high during the hierarchy adjustment. In our simulation, the initial HRing (Node 3000) was large and unbalanced. So the value was high at the first adjustment (time = 4) and lower at the second time.

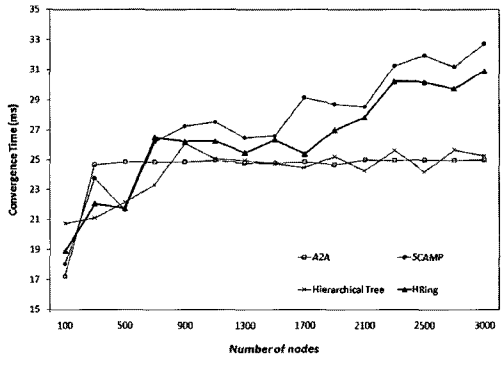



**Fig. 11.** Convergence time among the three approaches

Second, we compared the performance of our management with the other three approaches. Fig. 11

shows that the convergence time of A2A is almost constant because all the nodes in A2A have the whole membership information. However, A2A causes lots of unnecessary traffic. In the case of SCAMP and HRing, they have a shorter convergence time with a small number of nodes because they are more efficient in traffic generation.

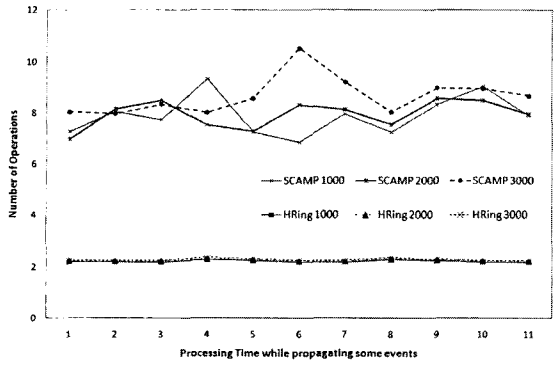



**Fig. 12.** Comparison of the management cost between HRing and SCAMP

Third, we compared the management cost of HRing, and especially $C_{operation}$, with SCAMP. Although the convergence time between HRing and SCAMP is similar, $C_{operation}$ of HRing is lower than SCAMP. Fig. 12 shows that HRing is more efficient in managing the membership than SCAMP, because it executes fewer operations than SCAMP.

The hierarchical tree approach should transfer more messages as the number of nodes increases. Therefore, the communication cost of HRing is better than SCAMP and the hierarchical tree approach. Fig. 13 shows bandwidth consumption as the scale grows more than 500 nodes.

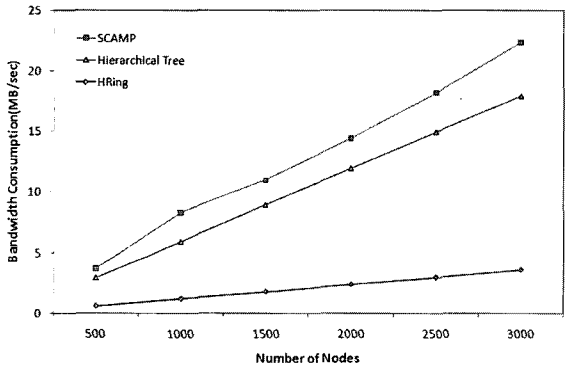



**Fig. 13.** Communication cost of the three approaches

## 7. Conclusion

In this paper, we proposed a hierarchical ring-based membership management called HRing, which can be used to construct P2P-like Grid environments. In this management, each node maintains a limited number of connections to others, thereby reducing the communication cost. Also, each node stores only local membership information, which leads to reduced costs for membership management. For these reasons, the management can be applied flexibly to changes in the Grid environments.

The performance of the proposed method was evaluated in three metrics: communication cost, management cost, and convergence time. The evaluation showed results that are preferable to A2A, gossip-based, and hierarchical tree.

## References

[1] Foster, I., Kesselman, C., eds., *The Grid: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann, San Francisco, CA (1999).

[2] Foster, I., Kesselman, C., Nick, J., Tuecke, S., *The physiology of the grid: An open grid services architecture for distributed systems integration* (2002).

[3] Alliance, T.G., *The web service resource framework* v1.2. http://globus.org/wsrf

[4] Gong, Y., Li, W., Sun, Y., Xu, Z., *A c/s and p2p hybrid resource discovery framework in grid environments*, In: 34th International Conference on Parallel Processing (ICPP'05), Oslo, Norway, IEEE Computer Society (June 2005) 261–268.

[5] Gupta, A., Agrawal, D., Abbadi, A. E., *Distributed resource discovery in large scale computing systems*, In: Proc. of the 2005 Symposium on Applications and the Internet (SAINT'05), Washington, DC, USA, IEEE Computer Society (Februrary 2005) 320–326.

[6] Hauswirth, M., Schmidt, R., *An overlay network for resource discovery in grids*, In: 16th International Workshop on Database and Expert Systems Applications (DEXA'05), Copenhagen, Denmark, IEEE Computer Society (August 2005) 343–348.

[7] Mastroianni, C., Talia, D., Verta, O., *A p2p approach for membership management and resource discovery in grids*, In: International Conference on Information Technology: Coding and Computing (ITCC'05). Volume 2, Las Vegas, NV, USA, IEEE Computer Society (April 2005) 168–174.

[8] Alliance, T.G.: Information services in the globus toolkit 3.0 release. http://wwwunix.globus.org/toolkit/mds

[9] Iamnitchi, A., Foster, I., *A peer-to-peer approach to resource location in grid environments*, In: 11th IEEE International Symposium on High Performance Distributed Computing (HPDC), (August 2002) 419.

[10] Talia, D., Trunfio, P., *Toward a synergy between p2p and grids*, IEEE Internet Computing **07**(4) (2003) 96, 94–95 11. : Gnutella Protocol Development. http://www.the-gdf.org

[12] Kazaa. http://www.kazaa.com

[13] Ratnasamy, S., Francis, P., Handley, M., Karp, R., Schenker, S., *A scalable content addressable network*, In: SIGCOMM '01: Proceedings of the 2001

Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, New York, NY, USA, ACM Press (2001) 161–172.

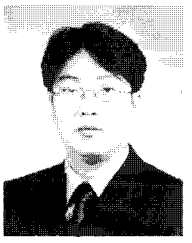
[14] Stoica, I., Morris, R., Karger, D., Kaashoek, M.F., Balakrishnan, H., *Chord: A scalable peer-to-peer lookup service for internet applications*, In: SIGCOMM'01: Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, New York, NY, USA, ACM Press (2001) 149–160.

[15] Zhao, B.Y., Huang, L., Stribling, J., Rhea, S.C., Joseph, A.D., Kubiatowicz, J.D., *Tapestry: A resilient global-scale overlay for service deployment*, IEEE Journal on Selected Areas in Communications, **22**(1) (January 2004) 41–53.

[16] Ganesh, A.J., Kermarrec, A.M., Massouli´e, L., *Peer-to-peer membership management for gossip-based protocols*, IEEE Transactions on Computers **52** (February 2003) 139–149.

[17] Shen, K., Yang, T., Chu, L., Holliday, J.L., Kuschner, D.A., Zhu, H., *Neptune: Scalable replication management programming support for cluster-based network services*, In: 3rd USENIX Symposium on Internet Technologies and Systems (USITS), San Francisco, CA, USA, USENIX (March 2001) 197–208.

[18] Zhou, J., Chu, L., Yang, T., *An efficient topology-adaptive membership protocol for large-scale cluster-based services*, In: 19th IEEE Internal Parallel and Distributed Processing Symposium (IPDPS'05) CD-ROM, Denver, CO, USA (April 2005).



**Tae-Wan Gu**

Gu received a BS degree in Computer Engineering and Mathematics and an MS degree in Computer Engineering from Hallym University in 2000 and 2002, respectively. He is now pursuing a Ph.D. degree at Hallym University. His research interests include Grid Computing, Grid Resource Discovery and Allocation, Information Service in the Wireless Network.

**Seong-Jun Hong**

Hong received a BS degree in Computer Science from Hallym University in 2006. He is now undertaking a master's course as a member of the programming system laboratory at Hallym University. His research interests include Distributed Systems and Network Security.

**Saangyong Uhmn**

Uhmn received BS and MS degrees in Computer Science from Hallym University in 1987 and 1997, respectively. He is now pursuing a Ph.D. degree at Hallym University. His research interests include Programming Languages, Design and Analysis of an Algorithm for Cluster Computing, and Bioinformatics.

**Kwang-Mo Lee**

Lee received BS, MS, and Ph.D. degrees from Seoul National University in 1975, 1984 and 1992 respectively. He was on the Faculty of the Department of Computer Science at Chosun University from 1980 to 1985, and eventually joined Hallym University, where he is now a Professor in the Computer Engineering Department. His current interests include Parallel Programming Languages, Distributed Processing, and Cluster Computing.