A Novel Hierarchical Community Architecture with End-to-End Delay Awareness for Communication Delay Enhancement

Khaled Ragab, Naohiro Kaji, Khoirul Anwar, Yuji Horikoshi,

Hisayuki Kuriyama and Kinji Mori

Tokyo Institute of Technology
2-12-1 Ookayama, Meguro, Tokyo 152-8552, Japan
Tel: +81-3-5734-2664, Fax: +81-3-5734-2510,
Email: {ragab@mori., nkaji@mori, anwar@mori. horikoshi@mori.,

kuriyama@mori. mori@}cs.titech.ac.jp

Abstract

The extreme dynamism and the rapidly changing user's requirements in current information systems promote imperative needs for the Autonomous Community Information System (ACIS) proposition. ACIS is a decentralized architecture that forms a community of individual end-users (*community members*) having the same interests and demands in somewhere, at specified time. It allows the community members to mutually cooperate and share information without loading up any single node excessively. In this paper, an efficient *autonomous decentralized community construction technique* is proposed to reduce: the communication delays among members with take into consideration the latency among them and the required time to join/leave. This technology organizes the community members into a hierarchy of sub-communities. This paper illustrates the step-step construction technique and the membership management operations for the proposed hierarchical community structure. In addition, it studies the community communication among members to quantify and study the tradeoff between the communication delay and the membership control (join/leave) overhead.

1. Introduction

Internet services provide large, rapidly evolving, highly accessed information spaces for anyone, anywhere and anytime. They are constructed from the service providers (SP)' point of view. SPs provide information regardless of the end-users' demands and situations (e.g. location and time). There is no discernment between differences in place and time; end-users in any situation receive the same contents. The web-based publish/subscribe has two traditional models: *Pull-model* that requires users accesses to the SP periodically to retrieve new information and *Push-model* that requires SP to deliver contents that change frequently. The pull-model has the following disadvantages: First, the information users receive may be not have changed since their last access and even if it has changed, will often contain a large redundant subset of the earlier retrieval contents. Second, when a rapid and sharp surge in the volume of requests arriving at a server often results in the server being overwhelmed and response times shooting up. Flash crowds are typically triggered by events of great interest, whether planned ones such as sport events (e.g. FIFA 1998 world cup event [1]) or unplanned ones such as, terrorists attack in September, 11, 2001 overwhelmed major news sites such as MSNBC and CNN [2]. The push-model fails to take the advantage of the collaborative power of the Internet, currently, 90% of Internet resources are invisible and untapped [3]. It often uses a one-to-many model where the SP is expected to deliver contents directly to each of the users. Clearly, this approach has scalability limitations. We believe that the time has come for an Internet infrastructure for efficient real time and cooperative content delivery.

The *Autonomous Community Information System* (ACIS) [4] has been proposed as a framework for large-scale information systems such as content delivery systems. ACIS is a decentralized architecture that forms a community of individual end-users (*community members*) having the same interests and demands in somewhere, at specified time. It allows the community members to mutually cooperate and share information without loading up any single node excessively. For a productive cooperation and flexible communication among members an *autonomous decentralized multilateral communication technique* has been proposed [5], [6]. It is a hybrid pull/push approach, when at least one of the community members has downloaded an interesting content for the community from the server (e.g. news server), she/he shares it with all the community members by forwarding it to all of them. Overall our results suggest that ACIS can achieve good performance for large number of members under the assumption that the communication cost between each node is one unit of time. The question then is: can ACIS support

large number of members with different communication cost. The main concern and contribution of this paper is to answer that question by constructing and maintaining the community overlay network. The performance of the community communication could be improved if the application level connectivity between community nodes is congruent with the underlying IP-level topology [7]. We identified that the use of underlying topology awareness may raise the bandwidth bottleneck problem. Thus, the communication delay will be increased. ACIS considers latency between nodes as an important criterion that need to be optimized. Thus, we have turned to construct the community network with node-node latency awareness. The problem of constructing an optimal community network overlay is known to be NP-hard [8], [9]. In this paper, an efficient *autonomous decentralized community construction technique* is proposed to reduce both the communication delay of a message that broadcasted to all community nodes with take into consideration the latency among them and the required time for membership management. This technique organizes the community nodes into a hierarchy of sub-communities as will be described in section 3. This paper studies the community communication among members to quantify and study the tradeoff between the communication delay and membership control (join/leave) overhead.
The remainder of this paper is organized as follow. Section 2 briefly clarifies the autonomous community information system concept, the system architecture and communication technique. Section 3 presents our proposed construction technique. Section 4 presents the community communication protocol and evaluation over the hierarchical structure. We review related work on application level multicast protocols in section 5. The last section draws conclusions and future work.

2. Community System, Architecture and Communication Technology

2.1 Concept

We have identified that the constructive cooperation among end-users assure the well-customized information service's provision and utilization. Blending the spirit of cooperation in the social communities, and the Autonomous Decentralized System (ADS) concept [10] [11], we have proposed the concept of *Autonomous Community Information System* (ACIS), [4]. The basis of the ACIS concept is to provide the information to specific users in specific place at specific time. On the contrary, current information systems provide the information to anyone, anywhere and anytime. Thus, we have defined *Autonomous Community* as a place of a coherent group of autonomous members having individual objectives, common interests and demands at specified time and somewhere/anywhere. The community members are autonomous, cooperative and active actors and they mutually cooperate to enhance the objectives for all of them timely. In ACIS, each community member acts both as an information sender and a receiver. Furthermore, each message from a participant is meaningful to all the other community members and at the same time every member is typically interested in data from all other senders in the community. Contrary to the peer-peer systems, the communication among the community members is conducted on multilateral basis, as will be shown in section 2.3. Community members cooperate not only for the satisfaction of one of them but also for all of them. Thus, the average satisfaction rate of M members is approximately one.

ACIS is a promising concept for information services operating at the edge of the network. It realizes the large-scale information system that successfully able to carry out, and enhance community members' objectives (e.g. timely information sharing) in a very dynamic environment. It guarantees the constructive cooperation and fairness among the community members with a very high degree of autonomy among them. We have developed a system architecture, called *Autonomous Decentralized Community System* (ADCS), that fosters the concept of the autonomous community information system.

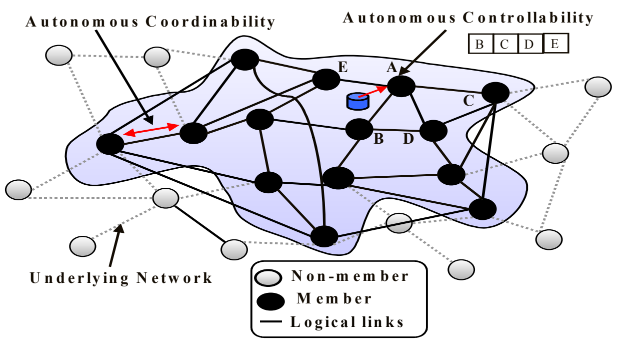

Fig. 1 Autonomous decentralized community system architecture.

2.2 Architecture

The autonomous decentralized community network is a self-organized logical topology. It is a set of nodes with considering the bilateral-hierarchy, the symmetric connectivity and the existence of loops. Community nodes are networked on a bilateral hierarchy basis. The bilateral logical contact between two community nodes will occur on the basis, the users of those nodes have same interests and demands, at specified time in somewhere. It is likely that in bilateral contacts, community members are get to know each other and share information. Each node keeps track of its immediate neighbors in a table contains their addresses. Each node knows its neighbor's nodes and shares this knowledge with other nodes for forming a loosely connected mass of nodes. For example, Figure 1 shows that each community node knows only four members. The bold lines represent the logical bilateral-link among the community nodes. Each node judges autonomously to join/leave the community network by creating/destroying its logical links with its neighbor's members based on its user's preferences. Section 3 will present the proposed hierarchical structure to manage the community network.

2.3 Autonomous Decentralized Community Communication Technology

The conventional communication, typically through Web browsers, has been built on the one-to-one communication protocol. In one-to-one, data travels between two users, e.g., e-mail, e-talk. This protocol gobbles up the network bandwidth and makes the real time services unresponsive. In the conventional one-to-many group's communication the message travels primarily from a server to multiple users, e.g., web download and software distribution. For very large groups (thousands of members) or very dynamic multicast groups (frequent joins and leaves), having a single group controller (e.g. *Bayeux* [12]) might not scale well.

Conventional communication techniques use the destination address (e.g. unicast address, multicast address) to send the data. In very changing environment likes ACIS, the state of the community nodes and the stability of connections are so unpredictable (i.e. end-users are frequently joined and left). Obviously, these conventional communication techniques are not applicable. Thus the autonomous decentralized community communication technique has broached [5], to assure a productive cooperation, a flexible and timely communication among members. The main ideas behind our proposed communication technique are: content-code communication (community service-based) for flexibility and multilateral communication for timely and productive cooperation among members.

2.3.1 Service-oriented and Multilateral Community Communication

The first main idea behind the autonomous decentralized community communication technique is the separation of the logical community services' identifier from the physical node address [14]. In this communication technique, the sender does not specify the destination address but only sends the content/request with its interest *Content Code* (CC) to its neighbor's nodes. CC is assigned on a type of the community service basis and enables a service to act as a logical node appropriate for the community service. Figure 2 shows the community communication message format. CC is uniquely defined with respect to the common interest of the community members (e.g. politic, news, etc.). The information content is further specified by its *Characterized Code* (CH). The CH is the hash of the message content. It is uniquely specified with respect to the message content (e.g. data or request). We can compute it with the *collision resistance hash* function (e.g. SHA-1 [13]) that ensures a uniform distribution of CH.

CC	CH	Data/Request

Fig. 2 Community communication message format.

The second main idea behind the autonomous decentralized community communication technique is *multilateral* communication for timely and productive cooperation [6]. The multilateral communication likely occurs between the community members that are already networked on a bilateral basis. All members communicate productively for the satisfaction for all the community members, as follow.

The proposed communication technique performs the communication among the community members that has called "*1→N*". A brief scenario of the 1→N community communication is described as follows. The community node asynchronously sends a message to each one from N neighbor's nodes. Then, those N nodes forward the same message to another N nodes in the next layer and so on, until all the community nodes received the message. The autonomy of the 1→N communication can be seen as follow. Each community node recognizes autonomously a member from non-member and judges autonomously to forward community messages to only N community neighbor's nodes. In order to avoid the congestion that may be happening if some of the community nodes simultaneously send identical messages, each node keeps a short memory of the recently routed messages and judges autonomously to forward only one copy of the received messages to the other neighbor's nodes. Moreover, each node autonomously takes a decision to keep or delete the short memory of the received message based on the frequency of receiving such message.

The 1→N communication technology does not rely on any central controller. Each community node has its own local information and communicates only with specified number (N) of the neighbor's nodes. There is no global information such as IP multicast group address [15] or multicast service nodes [9], [16].

2.3.2 Community Communication Protocols

The autonomous decentralized community communication technology has two communication protocols: *hybrid pull/push based* and *request /reply-all based*.

- *Hybrid pull/push based protocol*. When one of the community members has new information (e.g. downloaded from news server), she/he publishes it to all the community members using "1→N". Thus, it offers an effective solution to the flash crowd and represents a scalable solution for large-scale information dissemination systems.

- *Request/reply-all based protocol.* When a community member wants to locate information, she/he emits a request message. Then the others community members cooperate to locate the requested information. When any community node receives the requested message, it processes the request. If no results are found at that node, the node will forward the request to its neighbor's nodes by using "1→N". Otherwise, the node will produce results, such as pointers to the information or the whole content based on the size of the information. Then that node will send a reply message not only to the node, which requested the information but also to all the community members. The *reply to all* protocol affords the other community members to send the same request. Consequently, all the community members enrich their experiences and/or get to know new services without requesting, in which individually they cannot get to know. . In addition, it decreases the traffic per node by avoiding multiple requests for the same content.

In 1→N multilateral community communication all members cooperate for the satisfaction of all the community members, [6], contrary to the peer-peer (P2P) communication techniques. In P2P, peers cooperate for the satisfaction of only one, which request the information (unilateral benefits).

3. Autonomous Decentralized Community Construction Technique

3.1 Proposed Hierarchical structure

In this section, we propose a hierarchical structure that is used to manage N nodes currently in the community. Community nodes are organized in a multi-levels hierarchy of sub-communities. $S_i^{(j)}$ denotes the sub-community number i at the level j. Each sub-community $S_i^{(j)}$ has two special members: *Leader* and *Mediator*. The leader $L_i^{(j)}$ is responsible for membership management of the $S_i^{(j)}$. The mediator $M_i^{(j)}$ is responsible for transmitting contents from/to the $S_i^{(j)}$. The mediator $M_i^{(j)}$ is one of the neighbors of $L_i^{(j)}$ that has the smallest communication cost to $L_i^{(j)}$. It keeps a list of the Leader' s neighbors. In the sub-community $S_i^{(j)}$ all nodes have communication delay to the leader and the mediator that is bounded by a selected value $\alpha_i^{(j)}$. Thus, the sub-community $S_i^{(j)}$ at level j can be defined as a set of nodes x_0 that satisfy the *latency awareness condition* as follows:

$$S_i^{(j)} = \begin{Bmatrix} x_0;\ \sum\limits_{path_1} \delta(x_k, x_{k+1}) \le \alpha_i^{(j)} \quad if\ x_{m-1} = M_i^{(j)} \\ x_0;\ \sum\limits_{path_2} \delta(x_k, x_{k+1}) \le \alpha_i^{(j)}\ \ if\ x_{m-1} = L_i^{l(j)} \end{Bmatrix};$$

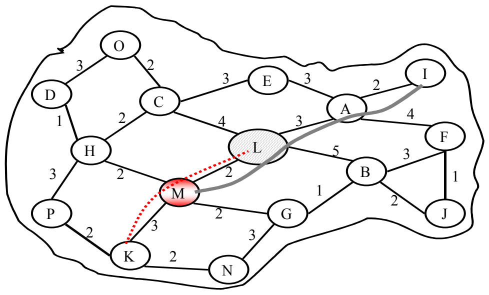

Fig. 3 An example of sub-community structure

Where $\delta(X, Y)$ denotes the currently end-to-end delay from X to Y measured by round-trip message. $Path_1 = x_0, x_1, \ldots, x_m = L_i^{(j)}$ as shown in figure 3 by the doted line from node k to L through the mediator node M. $Path_2 = x_0, x_1, \ldots, x_m = M_i^{(j)}$ as shown in figure 3 by the gray line from node I to M through nodes A and the leader node L. Figure 3 shows that the communication delay from any node to the leader and the mediator in the sub-community is less than or equal $\alpha_i^{(j)}$ =10ms (i.e. All nodes satisfy the *latency awareness condition*). Each sub-community leader $L_i^{(j)}$ determines $\alpha_i^{(j)}$ autonomously and adapts it to cope with the changing of the nodes communication delay.

Community nodes are organized in multi-level hierarchy of sub-communities that recursively defined as follows (where β_j is the number of sub-communities at level j and K is the number of levels):

1. Level 0 contains all nodes currently in the community. It is partitioned into β_0 sub-communities.
2. Level j+1 contains all leaders of the sub-communities at level j. It is partitioned into β_{j+1} sub-communities. Obviously, $\beta_j > \beta_{j+1}$; j=0,1,…k-1.
3. The leaders at level j automatically become members of a sub-community of leaders at level j+1, if they satisfy the latency awareness condition at level j+1. For j≥0, the number of nodes at level j+1 is β_j. Level-k has a few sub-communities (e.g. one or two). Each one contains a few members.
4. If a node belongs to level j then it must occur in one sub-community in each of the levels 0,1,… j-1. Furthermore, any node at a level j>0 must be a leader of the sub-community it belongs to at every lower level.
5. For any i and j, $\alpha_i^{(j)} < \alpha_i^{(j+1)}$.

This scheme is used to map the community nodes into levels as can be seen in figure 4. This figure shows that the hierarchical community structure consists of three levels. Level 0 contains 13 nodes and organized into six sub-communities. The sub-communities of leaders at level 0 form level 1 and they are organized into two sub-communities. Finally, level 2 contains only one sub-community contains two nodes.

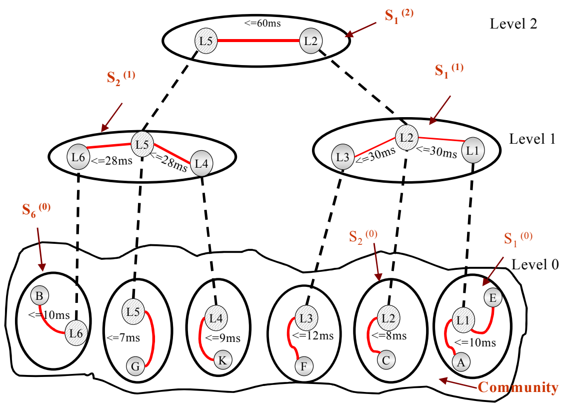

Fig. 4 Example: Multi-level hierarchy of sub-communities.

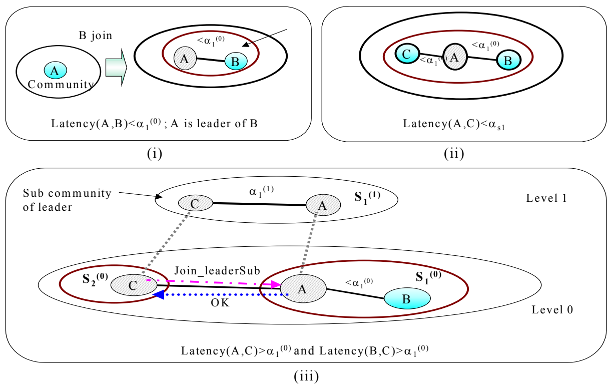

Fig. 5 Step-step construction hierarchical community structure.

3.2 Step-Step Construction

This section illustrates the construction of the community as hierarchical structure. Figure 5-i shows that node A initiates the community of an interest, creates a sub-community $S_1^{(0)}$ and becomes a leader of $S_1^{(0)}$. Then, node B has the same interest and wants to join the community. Node B sends a join request to node A. As soon as node A received join request it checks the round-trip latency to the joining node B. If the joining node satisfies the latency awareness condition (δ (A, B)$<\alpha_1^{(0)}$), then connects B to A by a logical link. Similarly, the joining node C sends a join request to a node in the community (e.g. B). This node forwards the join request to the leader of the sub-community it belongs (e.g. leader is node A). The leader checks the latency awareness condition with the joining node. Figure 5-ii shows the join process of node C where the latency awareness condition is satisfied. Otherwise, Figure 5-iii shows that the joining node C, joins the community and becomes a leader of its own created sub-community $S_2^{(0)}$. Then node C sends a *join_leadersub* ($S_1^{(1)}$) request to the neighbor leader (e.g. node A). Then this neighbor leader checks if it satisfies the latency awareness condition at the upper level. If δ (A,C)$<\alpha_1^{(1)}$) then C joins the sub-community of leaders $S_1^{(1)}$ at the upper level otherwise C creates a new sub-community of leaders $S_2^{(1)}$ at the upper level. Recursively new joining nodes join the community and consequently the community hierarchical structure is constructed.

3.3 Members Join and Leave

This section presents autonomous decentralized algorithms for community membership management. This approach is proposed to arrange the set of members into a hierarchical control topology with take into consideration the latency awareness condition. As new members join and existing members leave the community, the basic operations to create and maintain the hierarchy is required.

3.3.1 Join Process: Bottom-up and Top-down

When a node wishes to join the community, ADCS assumes that the node is able to get at least one community node A by an out-of-band bootstrap mechanism similar to Narada [17] and CAN [18]. In this paper we do not address the issue of the bootstrap mechanism. The joining node (X) sends a join request to one community node A as shown in figure 6. The join request is redirected along the hierarchical community structure *bottom-up* and *top-down* in order to find the appropriate sub-community as follows:

1. Node X sends join request to a node, A belongs to $S_i^{(j)}$ at level j=0.
2. Node A checks
   a) If [δ(X,A) + δ(X, $L_i^{(j)}$)] $<\alpha_i^{(j)}$ $\rightarrow$Create_link(X,A)
   b) Otherwise, forwards join request to $L_i^{(j)}$
3. $L_i^{(j)}$ checks
   a) If δ(X, $L_i^{(j)}$) $<\alpha_i^{(0)}$ $\rightarrow$ Call join_sub($S_i^{(j)}$); exit;
   b) Otherwise, $L_i^{(j)}$ forwards join request to the leader $L_u^{(j+1)}$ of the sub-community $S_u^{(j+1)}$ at the upper level where, $L_i^{(j)} \in S_u^{(j+1)}$.
4. Set j:= j+1 and then $L_u^{(j)}$ checks
   a) If δ(X, $L_u^{(j)}$) $<\alpha_u^{(0)}$ $\rightarrow$ forwards join request down.
   b) Otherwise, forwards join request to all members z_m belongs to $S_u^{(j)}$ except z_m=$L_u^{(j)}$.
5. Each node z_m checks
   a) If δ(X, z_m) $<\alpha_i^{(0)}$ $\rightarrow$ z_m sends a reply message to $L_u^{(j)}$ that contains "ok to join" and δ(X, z_m).
   b) Otherwise, z_m does not send a reply.
6. $L_u^{(j)}$ waits for a period of time γ to gather replies from all z_m nodes belong to the sub-community $S_u^{(j)}$. There exist two cases as follows:
   a) $L_u^{(j)}$ receives some replies and selects the one that has the smallest latency to X. Then, it forwards the join request down (i.e. set j:= j-1) to the selected leader that calls join_sub routine
   b) Otherwise, $L_u^{(j)}$ does not receive any reply within the time-out period γ. Thus, it forwards the join request to the upper level (i.e. set j:= j+1) and then repeats steps 4 - 6.

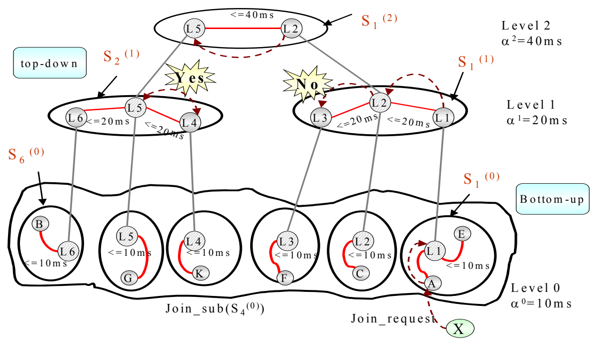

Fig. 6 Example: Join process in control tree.

7. The join request will forward from bottom to up and then top-down until satisfies the latency condition awareness. Otherwise there is no sub-community satisfies the latency awareness conditions then creates a new sub-community contains the joining member.

The join process terminates at level 0 when the joining node either finds a sub-community (e.g. $S_4^{(0)}$ in figure 6) that satisfies the latency awareness condition or not. Therefore, the join overhead is $O(\beta_0)$ in terms of the number of nodes that must check the latency with the joining node.

3.3.2 Leave/failure Process

When a node X wishes to leave the community, it notifies its neighbors in the sub-community $S_i^{(0)}$. $L_i^{(0)}$ and $M_i^{(0)}$ are the leader and the mediator of $S_i^{(0)}$ respectively. The leave algorithm will be described as follows:

1. If $[(X \neq L_i^{(0)})$ and $(X \neq M_i^{(0)})] \rightarrow$ Neighbors of X remove their links to X. For example, nodes k and G remove their links to the leaved node N in figure 3.
2. If $[(X= L_i^{(0)})$ and $L_i^{(0)} \in$ levels $l_0,\ldots,l_h] \rightarrow$ The mediator $M_i^{(j)}$ becomes a leader (i.e. $L_i^{(j)}= M_i^{(j)}$ for j=0..h) and selects a new mediator from its neighbors that has the smallest latency to it. This process is repeated on h-levels $l_0,\ldots,l_h$.
3. If $[(X= M_i^{(0)})$ and $M_i^{(0)} \in$ levels $l_0,\ldots,l_h] \rightarrow$ Each leader $L_i^{(j)}$ selects a new mediator from its neighbors that has the smallest latency to $L_i^{(j)}$, where j=0,..,h.

It is also required to consider the difficult case of node failure. In such a case, failure should be detected locally as follows. The neighboring nodes periodically exchange keep-alive message with the node X. If node X is unresponsive for a period T, it is presumed failed. All neighbors of the failed node update their neighbor's sets. This technique scales well: exchanging messages among small number of nodes does fault detection, and recovery from faults is local; only a small number of nodes are involved. In case of the sub-community leader fails, the mediator is still working and takes the leader responsibilities, connects to the leader's neighbors and selects another mediator to take its responsibilities. Therefore, the failure of the leader does not affect the community service continuity of other nodes. Similarly, in case of the mediator fails, the leader is still working and can appoint new mediator quickly. In addition, it is possible that some nodes failure can cause the community network to become partitioned. In such case, nodes must first detect the existence of a partition and then repair it by adding another links to reconnect the community network.

4. Community Communication on Hierarchy: Protocol and Evaluation

4.1 Community Communication Technique on Hierarchical Structure

For an efficient community communication, we create a hierarchically connected control topology. The content delivery path is implicitly defined in the way the hierarchy is structured and no additional route computations are required. The mediators in this hierarchy play important roles in this communication technique. A node sends a message to its neighbors in its sub-community $S_i^{(0)}$ by using $1\rightarrow N$ communication. Once the mediator $M_i^{(0)}$ receives such message, it forwards the message to all mediators belong to the sub-community at the upper level. Each mediator forwards such message to all members in its sub-community. Each mediator $M_i^{(j)}$ executes an instance of the following procedure.

```
Procedure Hcommunity_comm.($M_i^{(j)}$, rf)
{ // $M_i^{(j)}$ forwards the message that received from rf.
  if ($M_i^{(j)} \in$ levels $l_0,\ldots,l_m$ in sub-communities $S^{(0)}, \ldots S^{(m)}$)
    for (p = 0,…,m; $m \le K$)
    if ($rf \notin S^{(p)}$)
          ForwardMessageTo ($S^{(p)}$ - { $M^{(p)}$})
    end if
    end for
  end if }
```

Consequently all nodes in the community will receive such message. Assume all $\alpha_i^{(j)}=\alpha_{i+1}^{(j)}$ at each level j, where i=1,.., β_j-1 and j=0,..,k. Thus, the transmission time to forward a message from one community node to all nodes is bounded by

$$\alpha^{(k)} + \sum_{j=0}^{k-1} 2\alpha^{(j)} \qquad (1)$$

For example, figure 7 shows the message transmission initiated from node E. Node E sends it to all members in the sub-community $S_1^{(0)}$, once the mediator $M_1^{(0)}$ received such message, it forwards the message to all members in the $S_1^{(1)}$. The mediator $M_1^{(1)}$ forwards the message up to the level 2 and so on. The required sequence to forward the message to all members in the community through the hierarchical structure is shown in figure 7 by dotted arrows with index of order. In this figure, the transmission time is bounded by 100ms. Thus, the hierarchical sub-community approach considers the heterogeneity of node-node latencies.

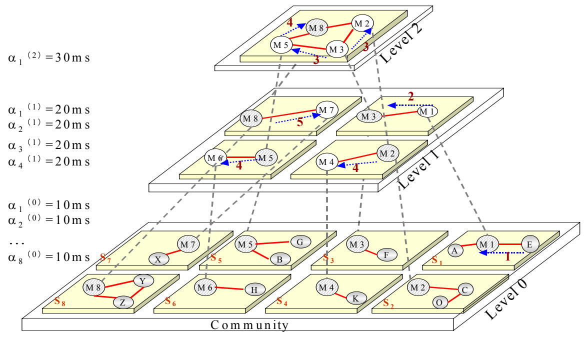

Fig. 7 Community communication through Hierarchical structure

It results in a community network clustering of community nodes into homogenous sub-communities thereby reducing the communication delay.

4.2 Performance Evaluation

This section presents a preliminary performance evaluation and a discussion of the tradeoff between join overhead and the communication delay. In addition, it presents our future plan for the simulation.

For the sake of simplicity, we assume that each sub-community at level j has an equal $\alpha^{(j)}=\alpha_i^{(j)}$ for i=1…β_j and $\alpha^{(j+1)}= C\,\alpha^{(j)}$; C>1. We also assume that the maximum communication latency between any two nodes in the community network is τ. Thus, $\tau \le \beta_k\,\alpha^{(k)} \rightarrow \tau \le \beta_k\,C\alpha^{(k-1)} \rightarrow \tau \le \beta_k\,C^2\alpha^{(k-2)} \rightarrow \tau \le \beta_k\,C^k\alpha^{(0)}$. Then, the number of levels, k can be determined as follows:

$$k \ge \log_C\left(\tau/\beta_k\alpha^{(0)}\right) \qquad (2)$$

For example, assume τ=120ms, β_k=2, $\alpha^{(0)}$ =10 and C=2, then the number of levels, k≈3.

The join overhead is O(β_0) in terms of the number of nodes to contact. It satisfies the following relation.

$$\beta_0 \propto 1/\alpha^{(0)} \qquad (3)$$

where $\tau \le \beta_0\,\alpha^{(0)}$. Then, as $\alpha^{(0)}$ increases, the join overhead decreases. Equation (1) shows the upper bound of the community communication delay on the proposed hierarchical structure. It can be written as follows:

$$\alpha^{(0)}(C^K + 2C^{k-1} + \ldots + 2) \qquad (4)$$

Where $\alpha^{(j+1)}= C\,\alpha^{(j)}$; C is constant and C>1.
The previous equation number 4 shows that if $\alpha^{(0)}$ increases then the number of levels, k decreases and consequently the communication delay among community nodes increases. Thus, the increasing of $\alpha^{(0)}$ leads to increase the communication delay and decrease the join overhead. This relation presents a tradeoff between the join overhead and the community communication delay. Now we are developing a simulation in order to show the effectiveness of our proposition and study this tradeoff. We are using the GT-ITM generator [21] to create 1,000 routers transit-sub graph as our underlying network topology. The routers will not run the code to construct and maintain the community network. In contrast, this code will be run on 100,000 end-nodes that will be randomly designated to routers with uniform probability. Our assumption likes [17], the end-nodes are connected by LAN link to its designated routers. Now we are developing ACIS over end-nodes for the sake of how to join and leave the ACIS network with achieving the latency awareness condition.

5. Related Work

ACIS, like Overcast [19], Narada [17] and ALMI [20], implement multicast, uses a self-organizing overlay network and assume only unicast support from the underlying network layer. Narada and ALMI target collaborative applications with a small number of group members. However, ACIS is a framework for collaborative applications with a large number of group members. ALMI is centralized overlay construction protocol that uses the tree-first approach. In this approach, a shared content delivery tree is constructed. It relies on a recursive algorithm to enhance the tree. Clearly, it constitutes a single point of failure for all control operations related to the group. Narada is distributed overlay construction protocol that uses the mesh-first approach. In this approach, every member should keep a full list of all other members. Therefore, both ALMI and Narada approaches do not scale well to the large group sizes. In contrast, ACIS takes a decentralized approach: no node knows the total system as shown in section3. In addition, ACIS creates a control hierarchical topology with considering the latency awareness condition. The content delivery path is implicitly defined on this hierarchical topology. Thus, the ACIS is scalable for large number of members.

Scattercast [9] and OMNI [16] are designed for global content distribution. They argue for infrastructure support, where proxies are deployed in the Internet to support large number of users. For large-scale data distributions, such as live web casts, a single source exists. In contrast in the ACIS, the nodes are considered to be equal peers and are organized in the community network. The community concept is a "*real*" end-system multicast approach. The end-systems (autonomous members) work cooperatively to deliver the data on the whole community members. ACIS is dedicated for multi-sender applications with large number of participants. It does not depend on the multicast support by the routers (e.g. IP multicast) and does not depend on the multicast service nodes MSNs (e.g. Scattercast and OMNI). A rapid and sharp surge in the volume of requests arriving at MSN often leads to a flash crowd. Clearly, MSN constitutes a single point of failure for information provisions to the group. Scattercast, like Narada takes a mesh-based approach to the tree creation problem. Therefore, Scattercast does not scale well to the large group sizes. In the other side, the ACIS scales well to the large number of members because each member is required to know a small number of other members (neighbors). The proposed community information system (ACIS) is a framework for both information sharing and large-scale data distribution applications. A comparison of different application level multicast systems with the community system is tabulated in table 1.

Table 1: Application level multicast systems

	Control approach	Overlay structure	Group size	Senders
ALMI	Centralized Tree-first	Peers	Small	Multi
NARADA	Distributed Mesh-first	Peers	Small	Multi
Scattercast	Distributed Mesh-first	MSNs	Small	Single
OMNI	Distributed Tree-first	MSNs	Small	Single
ACIS	Decentralized Loosely control Implicit-approach	Autonomous Members	Large	Multi

Some other recent projects like CAN [18] have also addressed the scalability issue in creating the overlay network. CAN defines a virtual d-dimensional Cartesian coordinate space, and each node owns a part of this space. Both ACIS and CAN nodes maintain constant state for other members and as a result exchange a constant number of periodic messages. However, this overhead in Bayeux is logarithmic.

6. Conclusion

This paper considers latency between community nodes as an important criterion that need to be optimized. For that reason an autonomous decentralized community construction technique is proposed. It organizes the community as a number of sub-communities. Each sub-community has a leader and a mediator. The latency from any node to the leader and the mediator is bounded by specific value α. To reduce both the communication delay among community nodes and the join overhead, this paper has presented a novel hierarchical structure of sub-communities. Furthermore, this paper has studied how to construct and maintain sub-communities.

We are currently extending this work in several directions. First, the frequent joining and leaving makes imperative needs for an adaptable hierarchical structure. To do that each sub-community leader must adapt these changes by adapting the value α. As a result, the sub-community has to be divided into ones with small α, or be merged with another ones to form a sub-community with large α. Second, due to the congestion in the network the latency from node to node may change dynamically. It is required that each node detects the communication delay to receive contents in the community. Thus, each node judges autonomously to leave the sub-community and rejoin another one to enhance its communication delay. Finally, we are developing a simulation to show the effectiveness of our proposed techniques.

7. References

[1] Martin Arlitt, Tai Jin, "Workload Characterization of the 1998 World Cup Web Site," Hewlett Packard Co. 1999.

[2] J. Jung, B. Kirshanmurthy, and M. Rabinovich, "Flash Crowds and Denial of Service Attacks: Characterization and Implications for CDNs and Web Sites", The 11th Int. World Wide Web Conference, WWW2002, Hawaii, USA, May, 2002.

[3] R. Dornfest, "Dark Matter, Sheep and the Cluster: Resolving Metaphor Collision in P2P," The O'Reilly Peer-to-Peer and Web Services Conference Washington, D.C., November 5-8, 2001

[4] K. Ragab, T. Ono, N. Kaji, K. Mori, " Autonomous Decentralized Community Concept and Architecture for a Complex Adaptive Information System," Proc. IEEE FTDCS, Puerto Rico, May 2003.

[5] K. Ragab, T. Ono, N. Kaji, K. Mori, "Community Communication Technology for Achieving Timeliness in Autonomous Decentralized Community Systems," Proc. IEEE IWADS, Beijing, China, pp 56-60, Nov., 2002.

[6] K. Ragab N. Kaji, K. Moriyama, K. Mori, "Scalable Multilateral Communication Technique for Large-Scale Information Systems," Proc. IEEE COMPSAC 2003, Nov., 2003, Dallas, USA.

[7] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. "*Topologically-aware overlay construction and server selection,"* INFOCOM, New York, New York, June 2002.

[8] M.R. Garrey and D.S. Johnson, "Computers and Intractability: A Guide to the Theory of NP-completeness," W. H. Freeman and Company, San Francisco, CA, 1979.

[9] Y. Chawathe. Et al., "Scattercast: An Architecture for Internet Broadcast Distribution as an Infrastructure Service," Ph.D Thesis, University of California, Berkeley, Dec. 2000.

[10] K. Mori, "Autonomous Decentralized Systems: Concept, Data Field Architecture and Future Trends," Proc. of the first Int. Sym. On ADS, (ISADS'93), IEEE, Kawasaki, Japan, pp. 28-34, 1993.

[11] K. Mori, H. Ihara, et al., "Autonomous Decentralized Software Structure and its Application," Proc. IEEE FJCC'86, pp.1056-1063. November 1986.

[12] S. Q. Zhuang, B. Y. Zhao, A. D. Hoseph, R. H. Katz and J. D. Kubiatowicz, "Bayeux: An Architecture for Scalable and Fault tolerant Wide-area Data Dissemination," In Proc. Of the 11th Int. Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV), June 2001.

[13] FIPS 180-1 Secure hash standard. Technical Report Publication 180-1, Federal Information Processing Standard (FIPS), National Inst. Of Standards and Technology, US Dept of Commerce, Washington D.C., April 1995.

[14] K. Ragab N. Kaji, K. Mori, "Service-Oriented Autonomous Decentralized Community Communication Technique for a Complex Adaptive Information System," Proc. IEEE/WIC WI 2003, Oct. 2003, Halifax, Canada.

[15] S. Deering and D. Cheriton, "Multicast routing in datagram internetworks and extended LANs," ACM Trans. on Computer Systems, 8(2):85-110, May 1990.

[16] S.Banerjee et al.,"Construction of an Efficient Overlay Multicast Infrastructure for Realtime Applications," Proc. of IEEE INFOCOM 2003.

[17] Y. H. Chu, S. G. Rao, and H. Zhang, "A case for end system multicast," in Proc. Of ACM Sigmetrics, June 2000, pp. 1-12.

[18] S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Shenker, "A Scalable Content-Addressable Network," Proc. Of SIGCOMM'01, California, USA.

[19] J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek, and J. W. O'toole. "Overcast: Reliable Multicasting with as n Overlay Network," In Proc. Of the Fourth Symposium on Operating System Design and Implementation (OSDI), pages 197-212, Oct. 2000.

[20] D. Pendarakis, S. Shi, D. Verma and M. Waldvogel, "ALMI: an application level multicast infrastructure", In Proc. of 3rd Usnex Symp. on Internet Technologies and Systems (USITS), March 2001.

[21] E. W. Zegura, K. Calvert, and S. Bhattacharjee, "How to model an internetwork" in Proc. of IEEE Infocom, 1996, San Francisco.