PAPER   Special Issue on Next Generation Networks Software

ACIS-Hierarchy: Enhancing Community Communication Delay for Large-Scale Information Systems

Khaled Ragab[*], *Nonmember and Kinji Mori*[*], *Regular Member*

SUMMARY To address the extreme dynamism and the rapidly changing user's requirements in current information systems, an *Autonomous Community Information System* (ACIS) has been developed. ACIS is a decentralized architecture that forms a community of individual end-users (*community members*) having the same interests and demands at specified time and location. It admits them to mutually cooperate and share information without loading up any single node excessively. ACIS does not embrace the advantage of heterogeneity of the community nodes-nodes latencies. In this paper, *ACIS-Hierarchy,* decentralized hierarchical community architecture is proposed. It considers latency between community nodes as an important criterion that need to be optimized. It takes advantage of heterogeneity of the community nodes-nodes latencies to improve the communication delay among the community. In this paper, an efficient *autonomous decentralized community construction technology* is proposed to reduce: the communication delays among members taking into consideration the latency among them and the required time to join/leave. This paper illustrates the step-step construction technology and the membership management operations for ACIS-Hierarchy. Experimental results show that ACIS-Hierarchy improves the community communication delay.

Key words: *Autonomous Community Information System, End-End delay awareness.*

1. Introduction

The demands for Internet services that provide large, rapidly evolving, highly accessed information spaces for specific end-users, at specific time and location are growing at an incredible rate. Current Internet information services (*e.g. Web-based publish/subscribe*) provide services 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).

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 not have been changed since their last access and even if it has been 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 [1] or unplanned ones such as, terrorists attack in September 11, 2001 [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]. The push-model 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 time has come for an Internet infrastructure for efficient real time and cooperative content delivery.

The *Autonomous Community Information System* (ACIS) [4], [5] has been proposed as a framework for large-scale content delivery systems (e.g. real-time stock quotes news and news delivery). The stock quotes news application is very sensitive to the communication delay. 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 a *multilateral communication technology* has been proposed [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 [5]. While in reality the nodes-nodes have different latencies. 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. It describes and evaluates an approach for 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]. Moreover, ACIS-Hierarchy deliberates the end-end node latency as an important criterion that must be optimized. Thus, we have turned to construct the community network with node-node

Manuscript received 14, November 2003.
Manuscript revised 23, January 2004.
[*] The author is with the Department of Computer Science, Tokyo Institute of Technology, Tokyo 152-8552, Japan.

latency awareness. The problem of constructing an optimal community overlay network is known to be NP-hard [8], [9]. In this paper, an efficient *autonomous decentralized community construction technology* is proposed to reduce both the communication delay of a message that broadcasted to all community nodes taking into consideration the latency among them and the required time for membership management. This technology organizes the community nodes into a hierarchy of sub-communities as will be described in section 3. This paper studies the community communication delay among members. The remainder of this paper is organized as follow. Section 2 briefly clarifies the ACIS concept, the system architecture and the communication technology. Section 3 presents the proposed construction technology. Section 4 presents the evaluation and the simulation results of the community communication protocol over the ACIS-Hierarchy. 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

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 at specific time and place. 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. Community members cooperate not only for the satisfaction of one of them but also for all of them.

2.2 Architecture

The community network is a self-organized logical topology. It is a set of nodes with considering the symmetric connectivity and the existence of loops. Each node keeps track of its 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, Fig. 1 shows that each community node knows only four members. The bold lines represent the logical 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 nodes based on its user's preferences. ACIS overlay network has been constructed as 2d-regular graph that contains d *Hamilton cycles*, without considering the node-node latency. The detailed descriptions of the ACIS construction processes are out of the scope of this paper and are presented in [5]. Section 3 will present the proposed ACIS-hierarchy structure to manage the community network with node-node latency awareness.

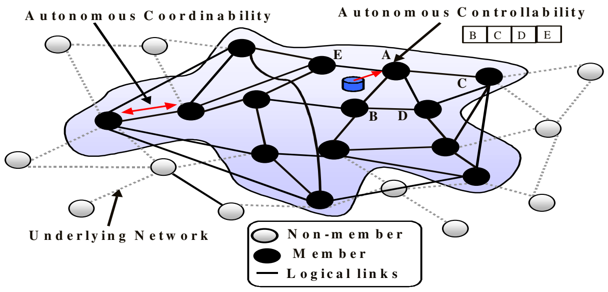

Fig. 1 ACIS: architecture.

2.3 Autonomous Decentralized Community Communication Technology

The conventional communication, such as one-to-one and one-to-many group's communication, for very large groups (thousands of members) or very dynamic multicast groups (frequent joins and leaves), having a single group controller might not scale well. Currently, there is no design for the application-level multicast protocol that scales to thousands of members (e.g. *Overcast* [19], Scattercast [9], Narada [17], *Bayeux* [12] and ALMI [20]). Conventional communication technologies use the destination address (e.g. Unicast address, multicast address) to send the data. They are not applicable in very changing environment likes ACIS (i.e. end-users are frequently join and leave). Thus, the autonomous decentralized community communication technique has broached [4], [5].

2.3.1 Service-oriented and Multilateral Community Communication

The first main idea behind the proposed 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. Fig. 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 uniquely specified by its *Characterized Code* (CH) and can be computed by the *collision resistance hash* function (e.g. SHA-1 [13]).

CC	**CH**	**Data/Request**

Fig. 2 Community communication: Message format.

The second main idea behind the community communication technique is *multilateral* communication for timely and productive cooperation [6]. In the multilateral communication, all members communicate productively for the satisfaction for all community members contrary to the peer-peer (P2P) communication techniques, as follow.

This communication technique performs the communication among the community members that has called "*1→N*". A brief scenario of 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 nodes in the community communication tree received the message. The 1→N has two protocols [4]: *hybrid pull/push based* and *request /reply-all based*. The first offers an effective solution to the flash crowd and represents a scalable solution for large-scale information dissemination systems. The second presents a scalable service discovery technique. The 1→N 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]. We present further details of 1→N over a randomly constructed k-regular overlay network (ACIS) in [4], [5], where N=k-1.

3. Autonomous Decentralized Community Construction Technology

To take advantage of heterogeneity of the community nodes-nodes communication delays, this section describes the proposed autonomous decentralized community construction technology. It organizes the community network into a multi-levels hierarchy of sub-communities. Next sub-section illustrates the sub-community definition, structure and the step-step construction technology.

3.1 Sub-Community

3.1.1 Definition and structure

$S_i^{(j)}$ denotes the i-th sub-community at level j. Each $S_i^{(j)}$ 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 $S_i^{(j)}$ all nodes have communication delay to the leader and the mediator that is bounded by a selected value $\alpha_i^{(j)}$. Thus, the $S_i^{(j)}$ at level j can be defined as a set of nodes x_0 that satisfies the *Latency Awareness Condition* (*LAC*) as follows:

$$S_i^{(j)} = \left\{ \begin{array}{l} x_0 ; \sum_{\substack{path_1 \\ k=0}}^{m-1} \delta(x_k, x_{k+1}) \le \alpha_i^{(j)} \ \ if \ \left(x_{m-1} = M_i^{(j)} \wedge x_m = L_i^{(j)} \right) \\ or \\ x_0 ; \sum_{\substack{path_2 \\ k=0}}^{m-1} \delta(x_k, x_{k+1}) \le \alpha_i^{(j)} \ \ if \ \left(x_{m-1} = L_i^{(j)} \wedge x_m = M_i^{(j)} \right) \end{array} \right\};$$

Where $\delta(X, Y)$ denotes the current end-end delay from X to Y that is measured by the delay of the round-trip message. The dotted line that is shown in Fig. 3 represents the $Path_1 = \{x_0, x_1, \ldots, x_{m-1}=M_i^{(j)}, x_m=L_i^{(j)}\}$ from node x_0=k to L through the mediator node M. The gray line in Fig. 3 represents the $Path_2 = \{x_0, x_1, \ldots, x_{m-1}=L_i^{(j)}, x_m=M_i^{(j)}\}$ from node x_0=I to M through node A and leader node L. Fig. 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 *LAC*). 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. We conclude that the sub-community is a set of nodes with considering LAC, existence of loops and node's connectivity is bounded by π.

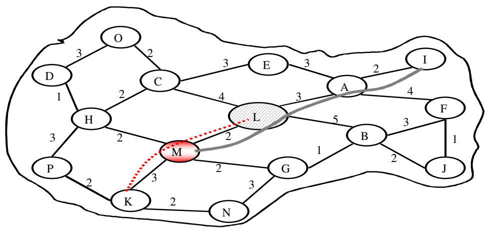

Fig. 3 An example of sub-community structure

3.1.2 Sub-community Step-Step Construction

To construct a sub-community step-step, we have developed a *join_sub(X, $S_i^{(j)}$)* routine. It inserts the new node X to the sub-community $S_i^{(j)}$. The *join_sub* scenario is as follows. The leader $L_i^{(j)}$ of the $S_i^{(j)}$ forwards join-request to its neighbor's nodes and then waits their replies. Each node receives the join–request processes the instance of the function *Node_joinCheck* that is given in Fig. 4, to decide autonomously the new node X can connect to itself or not. As soon as, the leader of the $S_i^{(j)}$ received some replies, it selects the nodes having the smallest latency to X and then X connects to at most π nodes from them.

```
Node_joinCheck(X, rf)
  { // X: new joining node and rf: received from node.
    Id=my_node_id;
    If (connectivity(Id) >π) {
      Forward(join-request(X), {neighbors{Id}- {rf}});
      Return 0;}
    Else {// Path={x_0=Id, x_1, …,x_m=L_i^(j)}
        $\tau = \sum_{x_i \in Path} \delta(x_i, x_{i+1}) + \delta(Id, X);$
      if (τ < $\alpha_i^{(j)}$) {
            Send ($L_i^{(j)}$,"Ok to join", τ);Return 1;}
      Else Return 0;
    } }
```

Fig. 4 Node_joinCheck function at each node

Fig. 5-i shows that node A is the leader of the sub-community $S_i^{(j)}$ with $\alpha_i^{(j)}$ =15ms. When new node B wants to join $S_i^{(j)}$, it calls the join_sub(B, $S_i^{(j)}$). Fig. 5-ii shows that B has joined to $S_i^{(j)}$ because δ(A, B)< $\alpha_i^{(j)}$. The new node C joined the sub-community $S_i^{(j)}$ as shown in Fig. 5-iii with considering that δ(C, B)+ δ(B, A) <$\alpha_i^{(j)}$ and δ(C, A) <$\alpha_i^{(j)}$. In addition, C becomes the mediator of $S_i^{(j)}$ instead of B because δ(C, A)< δ(B, A). Similarly, nodes D and E satisfy the *LAC* of $S_i^{(j)}$ and then they join it.

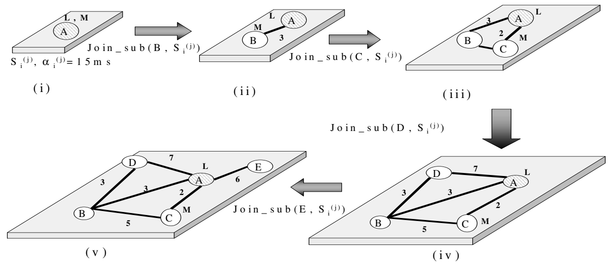

Fig. 5 Step-step sub-community construction

3.2 Proposed Hierarchical structure

This section presents the proposed *ACIS-hierarchy* structure that is developed to organize N community nodes into multi-level hierarchy of sub-communities [22]. It is 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 that are currently partaking 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 the sub-community of leaders at level j+1, if they satisfy the *LAC* at level j+1. For j≥0, the number of nodes at level j+1 is β_j. Level-k consists of a few sub-communities (e.g. one or two). Each sub-community contains a few members.
4. If a node belongs to level j then it must be in one sub-community in each of the levels 0, 1, … j-1. Furthermore, any node at level j>0 must be a leader of the sub-community and belongs to all lower levels.
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 shown in Fig. 6.

3.2.1 Step-Step Construction

This section illustrates the construction of the community as hierarchical structure. Fig. 7-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* wants to join the community and sends 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 (δ (A, B)< $\alpha_1^{(0)}$), then *A* connects *B* by a logical link. Similarly, the joining node *C* sends a join request to a node in the community (e.g. *B*). Node *B* forwards the join request to the leader of the sub-community it belongs (e.g. leader is node *A*). The leader checks if the joining node satisfies the *LAC* or not. Fig. 7-ii shows the join process of node *C* when the *LAC* is satisfied. Otherwise, Fig. 7-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, *A* checks if *C* satisfies the *LAC* 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 nodes join the community and consequently the community hierarchical structure is constructed.

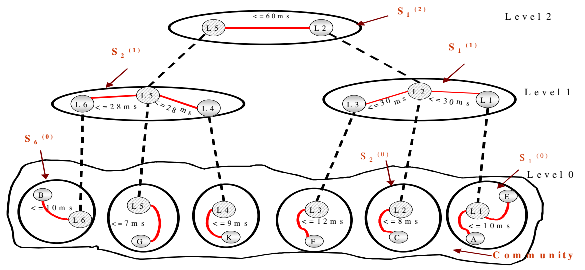

Fig. 6 Example: Multi-level hierarchy of sub-communities.

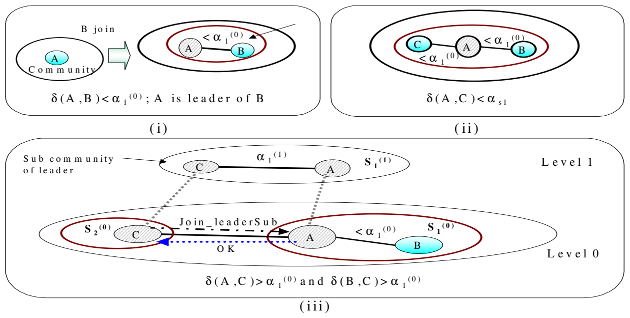
B join
Community
δ(A,B)<$\alpha_1^{(0)}$; A is leader of B
δ(A,C)<α_{s1}
(i)
(ii)
Sub community of leader
$\alpha_1^{(1)}$
$S_1^{(1)}$
Level 1
$S_2^{(0)}$
Join_leaderSub
OK
$S_1^{(0)}$
Level 0
δ(A,C)>$\alpha_1^{(0)}$ and δ(B,C)>$\alpha_1^{(0)}$
(iii)

Fig. 7 Step-step construction hierarchical community structure.

3.3 Join and Maintenance Processes

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

3.3.1 Join Process: Bottom-up and Top-down

When a node wishes to join the community, ACIS 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 Fig. 8. The join request is redirected along the hierarchical community structure *bottom-up* and *top-down* to find the appropriate sub-community as follows. Node X joins the community temporarily by contacting a first contact node, FN that belongs to $S_i^{(j)}$ at level j=0. Thus, *X* can receive community information during the joining process. Then, *X* calls the following join recursive function to find an appropriate sub-community.

```
Join( X, FN, $S_i^{(j)}$ ) {
  // Global variables: First_Contact = 1, CHECK_OUT=0.
// CHECK_OUT: # sub-communities have been checked out
  If ( (!First_Contact) && (CHECK_OUT==$\beta_0$){
     //No sub-community satisfied the LAC.
        Create_Sub(X, $S_{\beta0+1}^{(0)}$ ); // $L_{\beta0+1}^{(0)}$ =X
        Return ( $S_{\beta0+1}^{(0)}$);}
  If( (First_Contact) &&($\delta$(X,FN) + $\delta$( FN, $L_i^{(j)}$) ) < $\alpha_i^{(j)}$ )) {
        Create_link(X,FN); Return ($S_i^{(0)}$);}
```

```
If ( (j==0) && ($\delta$(X, $L_i^{(j)}$) < $\alpha_i^{(0)}$ ){
        **Join_Sub**(X, $S_i^{(0)}$ ); Return ($S_i^{(0)}$);}
First_Contact =0; // Set First_Contact=0 for next recursive call.
If ( (j==0) && ($\delta$(X, $L_i^{(j)}$) > $\alpha_i^{(0)}$ ) ){ // u: Up
     j = j+1; CHECK_OUT++;
     Return **(Join(X, $L_u^{(j)}$ , $S_u^{(j)}$ ));**}
If ( (j>0) && $\delta$(X, $L_i^{(j)}$) < $\alpha_i^{(0)}$ ) { // d: down
     j = j-1; CHECK_OUT++;
     Return **(Join(X, $L_d^{(j)}$, $S_d^{(j)}$ ));** }
If ( (j>0) && ($\delta$(X, $L_i^{(j)}$) > $\alpha_i^{(0)}$ ) ){
  // $L_i^{(j)}$ forwards a request to check LAC to its neighbors
  // Each neighbor forwards that request to its neighbors
 //  until $\forall z_m \in S_i^{(j)}$ received that request.
     For (k=0; k< $L_i^{(j)}$.nu_neighbors; k++)
        $L_i^{(j)}$.neighbor[k].**Node_Join_Check**(X, $L_i^{(j)}$);
     $L_i^{(j)}$.Wait($\gamma$); //$\gamma$ is timeout $L_i^{(j)}$ wait for replies
     CHECK_OUT =CHECK_OUT + $S_i^{(j)}$.Sub_Size -1;
     If ( $L_i^{(j)}$.received) {
   //Selected: SubID with minimum latency from the repliers
         Selected =**SubID_MinLatency** (Repliers);
        j = j-1; Return **(Join(X, $L_{Selected}^{(j)}$ , $S_{Selected}^{(j)}$ ));**}
     Else{  // i.e. No reply within $\gamma$ : $L_i^{(j)}$.received=0.
        j = j+1; Return **(Join(X, $L_u^{(j)}$ , $S_u^{(j)}$ ));**}          }}
```

The join recursive function terminates at level 0 when the joining node either finds a sub-community (e.g. $S_4^{(0)}$ in Fig. 8) that satisfies the *LAC* or not (i.e. *CHECK_OUT=β_0)*. Therefore, the join overhead is O(β_0) in terms of the number of nodes that must check the latency with the joining node. This construction technique reflects that each node takes the decision autonomously based on its local information and there is no specific server that is responsible for membership management. Thus, the proposed construction technique is scaleable for large number of nodes.

3.3.2 Leave/failure Process

When 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 is 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 Fig. 3.
2. If [(X= $L_i^{(0)}$) and $L_i^{(0)} \in$ levels $l_0,\ldots,l_h$] $\rightarrow$ Each mediator $M_i^{(j)}$ becomes leader (i.e. $L_i^{(j)}$= $M_i^{(j)}$), where j=0,..., h. Then, each leader selects a new node having the smallest latency to its neighbors and assigns it as a new mediator. 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 case, failure should be detected locally as follows. The neighboring nodes periodically exchange keep-alive message with node *X*. If *X* is unresponsive for a period T, it is presumed failed. All neighbors of the failed node update their neighbor's sets. This technology 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. If the leader of the sub-community fails and the mediator is still working then the mediator 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, when mediator fails, the leader is still working and can appoint new mediator quickly.

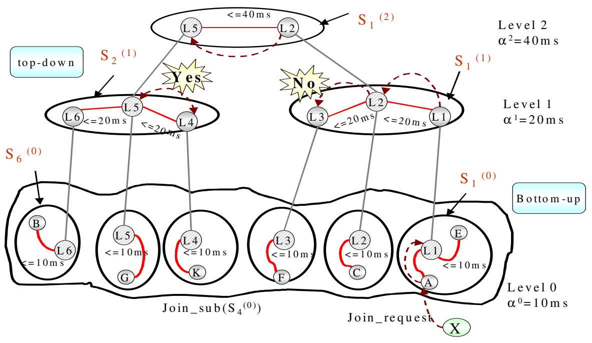

Fig. 8 Example: Join process in control tree.

4. ACIS-Hierarchy: Communication Protocol, Evaluation and Simulation

4.1 Community Communication Technology 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 technology. 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 ($S^{(p)}$ - { $M^{(p)}$}); }*
```

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 a community node to all nodes is bounded by

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

For example, Fig. 9 shows the message transmission initiated from node E. In this figure, the transmission time is bounded by 90ms. Thus, the hierarchical sub-community approach considers the heterogeneity of node-node latencies. It results in a community network clustering of community nodes into homogenous sub-communities thereby reducing the communication delay.

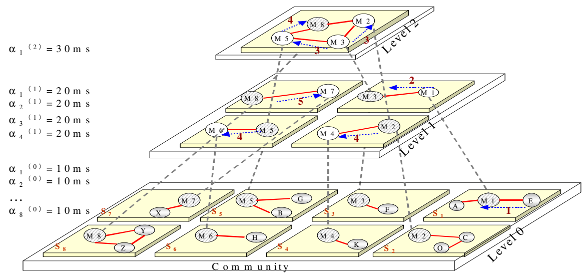

Fig. 9 Community communication through Hierarchical structure

4.2 Performance Evaluation

This section presents performance evaluation and discussion of the tradeoff between join overhead and the communication delay [22]. 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, $\tau \leq \beta_k \alpha^{(k)} \rightarrow \tau \leq \beta_k C\alpha^{(k-1)} \rightarrow \tau \leq \beta_k C^2\alpha^{(k-2)} \rightarrow \tau \leq \beta_k C^k\alpha^{(0)}$. Then, the number of levels, k can be determined as follows:

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

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:.

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

where $\tau \leq \beta_0\ \alpha^{(0)}$. In addition, the average number of control messages that are required to add new node to the ACIS-Hierarchy is proportional to β_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.

Equation (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 construction overhead. This relation presents a tradeoff between the construction overhead and the community communication delay as will be shown in section 4.3.3.

4.2.1 Performance Metrics

To evaluate the communication technique over ACIS-Hierarchy and compare it with ACIS and the conventional communication techniques; we used the following metrics.

- ***Latency.*** It measures the communication delay from a community node to all others community nodes.
- ***Relative Mean Delay Penalty (RMDP).*** It defines the ratio of the mean delay between two community nodes along the ACIS-Hierarchy, ACIS and Unicast to the IP multicast delay between them. It is used to measure of the increase delay that applications perceive while using ACIS-Hierarchy and ACIS.
- ***Stress.*** It measures the number of identical copies of a message carried by a physical link.

4.3 Simulation

To evaluate the performance, we have developed a simulation over a random generated network with different communication cost between nodes. This simulation demonstrates that ACIS-hierarchy architecture can perform quite well in realistic Internet settings. In this section, we study the performance issues with large community size using simulation experiments.

4.3.1 Setup

The simulations ran on two underlying network models, transit-stub model and Waxman model, with 100 routers linked by core links. The Georigia Tech [21] random graph generator is used to create both network models. Random link delay of 4-12ms was assigned to each core link. The community end-nodes were randomly assigned to routers in the core with uniform probability. Each community end-node was directly attached by a LAN link to its assigned router. The delay of each LAN link was set to be 1ms. End-nodes join the community network with joining rate 100 nodes/Sec with equal distribution. Members leave the community network with leaving rate 10 nodes/Sec with random distribution. We have conducted a simulation to compare ACIS-hierarchy with Unicast and ACIS [5]. Unicast operates over underlying network (i.e. no overlay network). ACIS operates over 4-regular graph overly network that spends 4-array connectivity for each node. ACIS-Hierarchy is organized with $\alpha^{(0)} = 5$ and $\alpha^{(j+1)} = 2*\alpha^{(j)}$. The number of sub-communities and levels changes with α and the number of end-nodes. For example, 300 end-nodes are organized into 4 levels and 100 sub-communities at level zero when $\alpha^{(0)} = 5$.

4.3.2 Communication Results

In each run of the simulation, one community member is picked as source at random and then the required communication cost to send a message to all nodes is evaluated. In this simulation, only static latency including no process delay is evaluated. We ran this simulation for 20 minutes as a result the community network size becomes 108,000 members. For simplicity, Fig. 10-a shows only the simulation results of the first 3.5 Seconds from the simulation running time. It plots the variations of the *Mean Communication Cost* (MCC) that is required to send a message from a node to all nodes participated at each instance of time during the experiment. It evaluates and compares ACIS-hierarchy with ACIS and Unicast over two underlying network models. ACIS-hierarchy has shown about 39% improvement of the MCC compared with ACIS and 93% compared with Unicast. We argue that ACIS-hierarchy shows 39% imprecision compared with ACIS to the proposed latency awareness hierarchical structure. The MCC of the Unicast, ACIS and ACIS-hierarchy over Waxman topology is small than both over transit-stub topology. That is to be expected because of the delay of the hierarchical of the transit-stub model. Moreover, the zoom part in fig. 10-a shows that the ACIS-hierarchy is not effective for small number of end-nodes compared with ACIS. However, ACIS-hierarchy is effective for large number of end-nodes compared with ACIS. In addition, Fig. 10-b plots the variation of RMDP for sequential Unicast, ACIS and ACIS-

Hierarchy over transit-stub model. The vertical axis represents a given value of RMDP associated with the community network size in log-scale presentation. ACIS-Hierarchy shows about 94% improvement of the RMDP to Unicast and about 47% imprecision to ACIS. From these results we conclude that the ACIS-hierarchy enhance the community communication compared to the ACIS. Thus, the timeliness is achieved.

4.3.3 Construction Overhead

The construction overhead is measured as the average number of control messages per physical links that are required to join a new node to the ACIS-hierarchy. Fig. 10-c shows the variation of $\alpha^{(0)}$ with the normalized MCC and the normalized construction overhead. It verifies that the trade-off between the community communication delay and the construction overhead exists. The average number of control messages per physical links to add a new node to 300 end-nodes is 1.525 and 0.034, where $\alpha^{(0)}$ is 5 and 100 respectively. The results indicate that the proposed construction technique is scalable and does not show any practical problem.

4.3.4 Link Stress

We have conducted our experiment with a community size 300 members. One of the members picked as source at random and we evaluate the stress of each physical link. We study the variation of physical link stress under ACIS-Hierarchy, ACIS, IP Multicast and naïve Unicast as shown in Fig. 10-d. The horizontal axis represents stress and the vertical axis represents the number of physical links with a given stress. The stress is at most 1 for IP Multicast. Under ACIS-Hierarchy, ACIS and naïve Unicast, most links have a small stress-this to be expected. However, the significant lies in the tails of the plots. Under naive Unicast, one link has stress 299. This because that links near the source have high stress. However, ACIS-Hierarchy and ACIS distribute the stress more evenly across the physical links. ACIS-Hierarchy has about 94% improvement over naive Unicast. ACIS has about 55% improvement over naïve Unicast. The zoom part in Fig. 10-d shows that the ACIS-Hierarchy maximum stress per physical link is 17. Thus, the ACIS-Hierarchy stress is close to IP-Multicast. Owing to the design of the latency-awareness ACIS-Hierarchy structure, the ACIS-Hierarchy has about 87% improvement over ACIS.

5. Related Work

ACIS-Hierarchy, 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-Hierarchy and ACIS is 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.

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-Hierarchy, 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 contrast, ACIS-Hierarchy takes a decentralized approach: no node knows the total system as shown in section 3. In addition, ACIS-Hierarchy 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-Hierarchy is scalable for large number of members. It is framework for both information sharing and large-scale data distribution applications.

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. To reduce both the communication delay among community nodes and the join overhead, this paper has presented a novel hierarchical structure, *ACIS-Hierarchy,* of sub-communities. Furthermore, this paper has studied how to construct and maintain sub-communities. Finally, this paper presents the simulation results that show the effectiveness of the proposed technologies. We are currently extending this work into several directions such as adapting the changes of the end-to-end nodes latencies.

References

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

[2] J.Jung, "Flash Crowds and Denial of Service Attacks: Characterization and Implications for CDNs and Web Sites", WWW2002.

[3] R. Dornfest, "Dark Matter, Sheep and the Cluster: Resolving Metaphor Collision in P2P," The O'Reilly P2P and Web Services Conference 2001.

[4] K. Ragab, N. Kaji and K. Mori, "Scalable Multilateral Autonomous Decentralized Community Communication Technique for Large-Scale Information Systems", IEICE Trans. on Comm., Vol. E87-B, No. 3, March 2004.

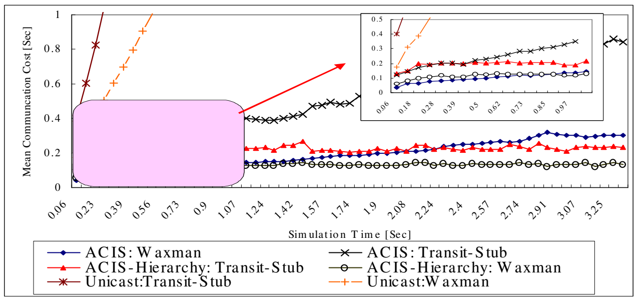

Fig. 10-a MCC: Comparison

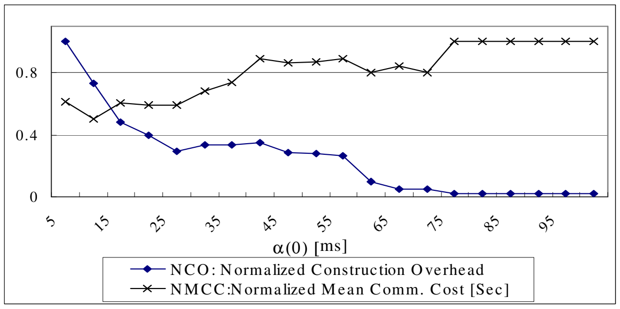

Fig. 10-c Trade-off

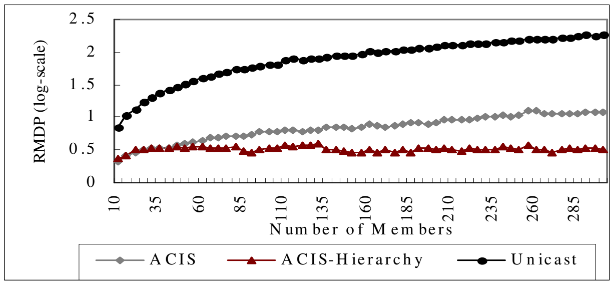

Fig. 10-b RMDP: Comparison

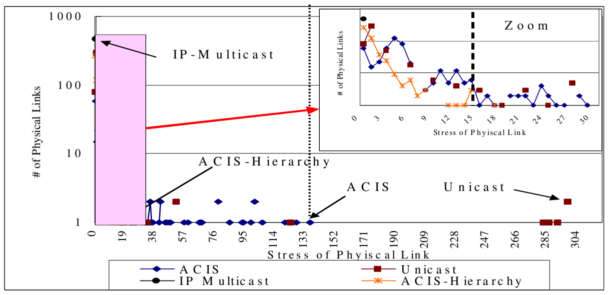

Fig. 10-d Physical link stress

[5] K. Ragab N. Kaji, K. Mori, “ACIS: A large-scale Autonomous Decentralized Community Communication Infrastructure, IEICE Trans. Info. Sys. Vol. E87-D, No. 4, April.

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

[7] S. Ratnasamy “Topologically-aware overlay construction and server selection,” INFOCOM, New York, June 2002.

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

[9] Y. Chawathe, “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. IEEE ISADS, Japan, 1993.

[11] K. Mori, et al., “Autonomous Decentralized Software Structure and its Application,” Proc. IEEE FJCC'86, November 1986.

[12] S. Q. Zhuang, Et al., “Bayeux: An Architecture for Scalable and Fault tolerant Wide-area Data Dissemination,” Proc. NOSSDAV, June 2001.

[13] FIPS 180-1 Secure hash standard, Washington D.C., April 1995.

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

[15] S. Deering, “Multicast routing in datagram internetworks and extended LANs,” ACM Trans. on Comp. Sys., 8(2):85-110, May 1990.

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

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

[18] S. Ratnasamy, Et al.,, “A Scalable Content-Addressable Network,” Proc. Of SIGCOMM’01, California, USA.

[19] J. Jannotti, “Overcast: Reliable Multicasting with as n Overlay Network,” In Proc. 4th Symp. OSDI, Oct. 2000.

[20] D. Pendarakis, “ALMI: an application level multicast infrastructure”, In Proc. of 3rd Symp. USITS, March 2001.

[21] E. W. Zegura, “How to model an Internetwork” IEEE-Proc. INFOCOM, 1996, San Francisco.

[22] K. Ragab N.Kaji, K. Anwar, Y. Hirokoshi, H. Kuriyama and K. Mori, “A Novel Hierarchical Community Architecture with End-to-End Delay Awareness for Communication Delay Enhancement”, Proc. IEEE/IPSJ SAINT’04, Jan., 2004, Tokyo, Japan.