Multi-layer Autonomous Community Overlay Network for Enhancing Communication Delay

Khaled Ragab, 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., horikoshi@mori., kuriyama@mori., mori@}cs.titech.ac.jp

Abstract

Autonomous Decentralized Community Communication System (ADCCS) is a decentralized architecture that forms a community of individual and autonomous end-users (community members) having the same interests and demands in somewhere, at specified time. It enables them to mutually cooperate and share information without loading up any single node excessively. ADCCS does not seize the advantage of heterogeneity of the community nodes-nodes latencies. Thus, this paper proposes ADCCS-Multilayer. It is an autonomous decentralized multi-layer community overlay network that enhances the communication delay among the community members. This paper presents an efficient step-step construction technique to reduce both the communication delays among members and the required time to join/leave. Experimental results show that ADCCS-Multilayer enhances the community communication delay.

1. Introduction

Current Internet information services (*e.g. Web-based publish/subscribe*) provide services for anyone, anywhere and anytime. The service providers (SP) provide information regardless of the end-users' demands and situations (e.g. location and time). Users in any situation receive the same contents. We believe that the time has come for an Internet infrastructure that provides large, rapidly evolving, customized, highly accessed information spaces for specific end-users, in specific time and location.

The web-based publish/subscribe has two traditional models: *Pull-model* that requires users accesses to the SP frequently to retrieve new information and *Push-model* that is required by SP to deliver contents that change frequently. Each of these models has disadvantages. With the pull model, users might access information and find that it's unchanged since their last access or that it contains a large, redundant subset of earlier content retrievals. This model is also vulnerable to flash crowds – rapid, sharp surges in requests that overwhelm the server and dramatically increase response time. For its part, the push-model often uses a one-to-many model in which the SP delivers contents directly to each user. Clearly, this approach has scalability limitations.

To address the drawbacks of the pull and push models, based on the success of the *Autonomous Decentralized System* (ADS) [1, 2, 3], we have designed an *Autonomous Decentralized Community Communication System* (ADCCS) [4] as a framework for large-scale information systems such as content delivery systems. ADCCS forms a community of individual *members* having the same interests and demands in somewhere, at specified time. It allows the 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]. 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 ADCCS can achieve good performance for large number of members under the assumption that the communication cost between each node is one unit of time [4]. While in reality the latencies from nodes-nodes are different. The question then is: can ADCCS support large number of members with different communication cost. The main concern and contribution of this paper is to answer this question. It describes and evaluates an approach for constructing and maintaining an autonomous *Community Overlay Network* (CON). The performance of the communication could be improved if the CON is constructed with node-node latency awareness. In this paper, we propose the CON's *construction technique* to reduce both the communication delay of a message that broadcasted to all community nodes with considering the latency among them and the required time for membership management.

The remainder of this paper is organized as follow. Section 2 briefly clarifies the ADCCS concept and the system architecture. Section 3 presents the proposed construction technique. Section 4 presents the community communication techniques. The evaluation and the simulation results of this communication protocol over the ADCCS-Multilayer are described in section 5. We review related work on application level multicast protocols in section 6. The last section draws conclusion.

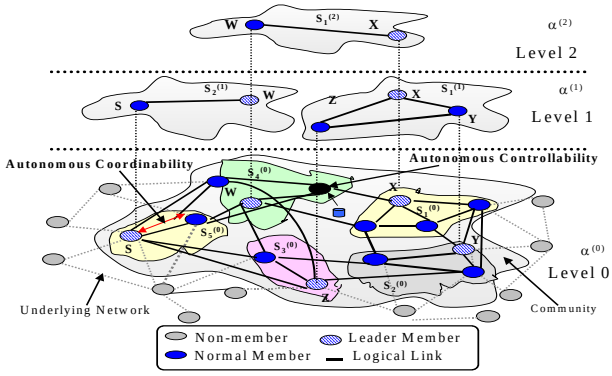

Fig. 1 ADCCS-Multilayer: architecture.

2. ADCCS

2.1 Concept

The ADCCS technology [4] has been proposed on the basis of the *Autonomous Decentralized System* (ADS) concept [2] [3]. The basis of the ADCCS is to provide the information to specific users in specific place at specific time (for example when a particular Mall holds a sale at specific time [6]). In contrast, current information systems provide the information to anyone, anywhere and anytime. The *Autonomous Community* is a communication environment for 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 ADCCS, each member plays a dual role as an information sender and receiver. Furthermore, each message from a participant is meaningful to all other 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 CON is a self-organized logical topology. It is a set of nodes with considering the existence of loops. It organizes the community nodes into multi-level of sub-communities [7]. The i-th sub-community at level j is denoted by $S_i^{(j)}$. *Leader* is a special member of $S_i^{(j)}$ that is responsible for membership management of the $S_i^{(j)}$. Each level j+1 contains leaders of all $S_i^{(j)}$ at level j. Figure 1 shows a CON that contains five sub-communities in level 0, two sub-communities in level 1 and one sub-community in level 2.

In figure 1, 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 not only on the node latency with its neighbors, but also on its user's preferences. Each node keeps track of its neighbors in a table that contains their addresses. Each node knows its neighbors and shares this knowledge with other nodes for forming a loosely connected mass of nodes.

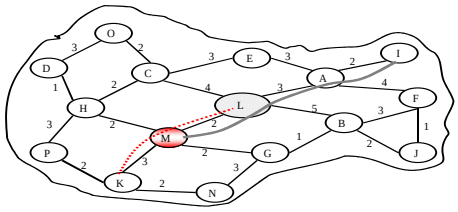

Fig. 2 An example of sub-community structure

3. Autonomous Decentralized Community Construction Technique

To take advantage of the heterogeneity of the community nodes-nodes latencies, this section describes the proposed construction technique. It organizes the community network into multi-levels of sub-communities.

3.1 Sub-Community

3.1.1 Definition and structure

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 latency to $L_i^{(j)}$. It keeps a list of the Leader' s neighbors. Each node belongs to the $S_i^{(j)}$ has communication delay to the leader and the mediator that is bounded by a selected value $\alpha_i^{(j)}$. Thus, $S_i^{(j)}$ at level j can be defined as a set of nodes x_0 that satisfies the *Latency Awareness Condition* (*LAC*) as follows:

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

Where δ(X, Y) denotes the current round-trip end-to-end delay from X to Y. The dotted line in figure 2 represents the $Path_1$= x_0, x_1,…,x_m=$L_i^{(j)}$ from node x_0=k to L through the mediator node M. The gray line in figure 2 represents the $Path_2$= x_0, x_1,…,x_m=$M_i^{(j)}$ from node x_0=I to M through node A and leader node L. Figure 2 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. Each leader $L_i^{(j)}$ autonomously determines $\alpha_i^{(j)}$ and adapts it to cope with the changing of the nodes communication delays. We conclude that the sub-community is a set of nodes with considering *LAC*, existing of loops and bounding node's connectivity by π.

3.1.2 Sub-community Step-Step Construction

To construct the $S_i^{(j)}$ step-step, we have developed a *join_sub(X, $S_i^{(j)}$)* routine. It inserts the new node X to the $S_i^{(j)}$. The *join_sub* scenario is as follows. The leader $L_i^{(j)}$ of the $S_i^{(j)}$ forwards join-request to its neighbors and then waits their replies. As soon as a node received the join–request, it processes the instance of the procedure *Node_joinCheck* that is given in figure 3, 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 that have the smallest latency to X and then X connects to at most π nodes from them.

Procedure 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}});
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 ($\tau < \alpha_i^{(j)}$) Send($L_i^{(j)}$,"Ok to join", τ); } }

Fig. 3 Node_joinCheck operation at each node

3.2 Proposed Multi-layer structure

The *ADCCS-Multilayer* structure organizes CON into multi-level of sub-communities [7] as follows:

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 and mediators 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/mediators at level j automatically become members of the sub-community of leaders/mediators 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/ mediator of the sub-community and belongs to all lower levels.
5. For any i and j, $\alpha_i^{(j)} < \alpha_i^{(j+1)}$.

Where β_j is the number of sub-communities at level j and K is the number of levels. This scheme is used to map the community nodes into levels as shown in figures 1 and 5.

3.2.1 Step-Step Construction

This section illustrates the step-step construction of the ADCCS-Multilayer. Figure 4-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 then sends join request to node A. As soon as, node A receives join request it checks the round-trip latency to the joining node B. If the joining node satisfies the *LAC* (δ (A, B)< $\alpha_1^{(0)}$), then A connects B via 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. Figure 4-ii shows the join process of node C when the *LAC* satisfied. Otherwise, Figure 4-iii shows that the joining node C, joins the community and becomes a leader of its own created $S_2^{(0)}$. 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 or not. 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 ADCCS-Multilayer is constructed.

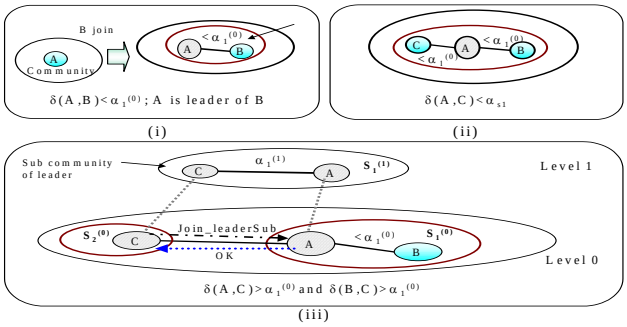

Fig. 4 Step-step construction multi-layer CON structure.

3.3 Members Join and Leave

As new members join and existing members leave the community, the basic operations to create and maintain the ADCCS-Multilayer structure is required. This section presents autonomous decentralized algorithms for community membership management. This approach is proposed to arrange the set of members into a multi-layer control topology with considering the latency awareness condition.

3.3.1 Join Process: Bottom-up and Top-down

Similar to Narada [8], ADCCS-Multilayer is able to get at least one community node *A* by an out-of-band bootstrap. The joining node *X* sends a join request to one node *A* as shown in figure 5. The join request is redirected along the multi-layer CON structure *bottom-up* and *top-down* to find the appropriate sub-community as follows:

1. Node *X* sends join request to node *A* that belongs to $S_i^{(j)}$ at level j=0.
2. Node *A* checks
   a) If [δ(X,A) + δ(X, $L_i^{(j)}$)] < $\alpha_i^{(j)}$ → Create_link(X,A)
   b) Otherwise, forwards join request to $L_i^{(j)}$
3. $L_i^{(j)}$ checks
   a) If δ(X, $L_i^{(j)}$) < $\alpha_i^{(0)}$ → Call join_sub(X, $S_i^{(j)}$); 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, $L_u^{(j)}$) < $\alpha_u^{(0)}$ → 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) < $\alpha_i^{(0)}$ → z_m sends a reply message to $L_u^{(j)}$ that contains "ok to join" and δ(X, z_m).

b) Otherwise, no node from z_m send any reply message to $L_u^{(j)}$.

6. $L_u^{(j)}$ waits for a period of time γ to gather replies from all z_m nodes belong to the $S_u^{(j)}$. Then, two cases exist as follows.
   a) $L_u^{(j)}$ receives some replies and selects 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.
7. The join request is forward from bottom to up and then top to down until the *LAC* is satisfied. Otherwise, there is no sub-community satisfies the *LAC* then a new sub-community containing the joining node is created.

The join process terminates at level 0 when the joining node either finds a sub-community (e.g. $S_4^{(0)}$ in figure 5) that satisfies the *LAC* 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.

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 scalable for large number of nodes.

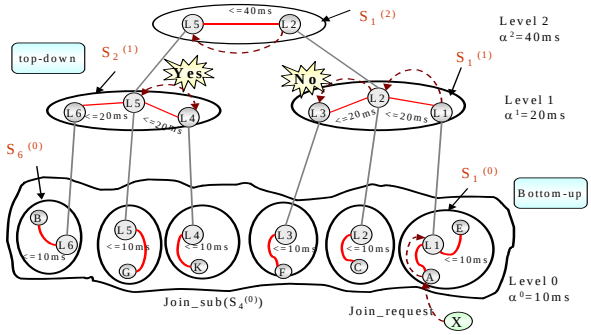

Fig. 5 Example: Join process in control tree.

3.3.2 Leave/failure Process

When node *X* wants to leave the community, it notifies its neighbors in the $S_i^{(0)}$. $L_i^{(0)}$ 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 figure 2.
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 that has 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 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. 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.

4. Community Communication

4.1 Autonomous Decentralized Community Communication Technique

The conventional communication has been built on the one-to-one and one-to-many group's communication protocols. Currently, there is no design for the application-level multicast protocol that scales to thousands of members (e.g. *Overcast* [10], Scattercast [9], Narada [8] and ALMI [11]). Conventional communication techniques use the destination address (e.g. unicast address, multicast address) to send the data. They are not applicable in very changing environment likes ADCCS (i.e. end-users are frequently join and leave). Thus, the following communication technique has broached [5], [4].

4.1.1 Service-oriented and Multilateral Community Communication

The service-oriented community communication technique separates the logical community services' identifier from the physical node address [4]. Thus, 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 6 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).

CC	CH	Data/Request

Fig. 6 Message format.

In the multilateral community communication [4], all members communicate productively for the satisfaction for all members contrary to the peer-peer (P2P) communication techniques, as follow.

The community 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, these 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 technique 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 technique 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 [12] or multicast service nodes [9]. We present further details of 1→N over a CON that is constructed as k-regular overlay network (ADCCS) in [4], [5], where N=k-1.

4.2 Community Communication Technique on Multi-layer Structure

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

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

Thus, the ADCCS-Multilayer approach considers the heterogeneity of node-node latencies. It results in a CON clustering of nodes into homogenous sub-communities thereby reducing the communication delay.

5. Performance

5.1 Metrics

To evaluate the community communication technique over ADCCS-Multilayer and compare it with ADCCS [4], [5] and the conventional communication techniques; we use the following metrics that show the effectiveness of the proposed multi-layer construction techniques.

- *Latency* measures the communication delay from a community node to all others community nodes.
- *Relative Mean Delay Penalty (RMDP)* measures the increased delay that applications perceive while using ADCCS-Multilayer and ADCCS.
- *Stress* measures the number of identical copies of a message carried by a physical link. Obviously, we'd like to keep the link stress on all links low as possible.

5.2 Simulation

We set the simulation's parameters as follows. The Georgia Tech [13] random graph generator using the transit-stub model is used to construct a network topology with 100 routers linked by core links. 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/leave the CON with random distribution as had shown in figure 7-a. It shows the variations of the community network size with the simulation time. The CON spends 4-array connectivity for each end-node and it is organized into two levels (i.e. set k = 2). Each sub-community leader L_i sets $\alpha^{(0)}$ = 4ms, approximately less than or equal Minimum(τ_{ij}), where L_i is connected to router i and τ_{ij} is the communication cost from router i to router j. As a result it is expected that the communication cost on ADCCS-Multilayer be close to the IP-Multicast.

5.2.1 Communication Results

We have conducted a simulation to compare ADCCS [4], ADCCS-Multilayer with unicast. 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. We ran this simulation for 20 minutes, for the sake of the simplicity; figure 7-b 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) required to send a message from a node to all nodes participated at each instance of time during the experiment. ADCCS-Multilayer has shown about 39% improvement of the MCC compared with ADCCS and 93% compared with unicast. We argue that ADCCS-Multilayer shows 39% imprecision compared with ADCCS to the proposed latency-awareness multi-layer structure of sub-communities. In addition, figure 7-c plots the variation of RMDP for sequential unicast, ADCCS and ADCCS-Multilayer. The vertical axis represents a given value of RMDP associated with the community network size in log-scale presentation. ADCCS-Multilayer shows about 94% improvement of the RMDP to unicast and about 39% imprecision to ADCCS of the RMDP. From these results we conclude that the ADCCS-Multilayer enhance the community communication compared to the ADCCS.

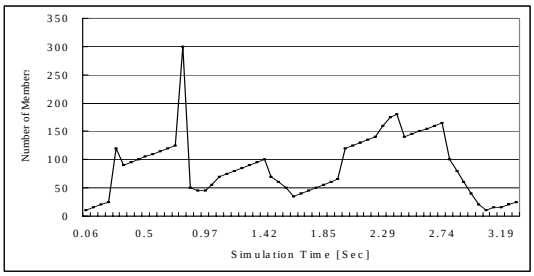

Fig. 7-a Variation of Community network size per time.

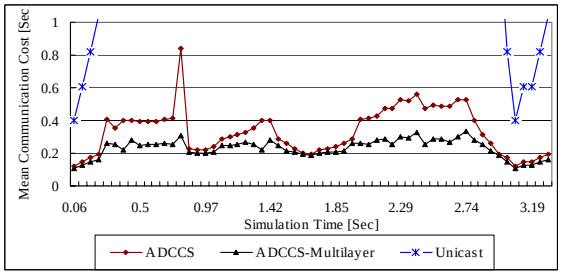

Fig. 7-b MCC: Comparison

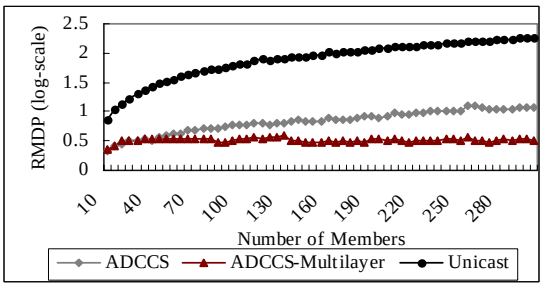

Fig. 7-c RMDP: Comparison

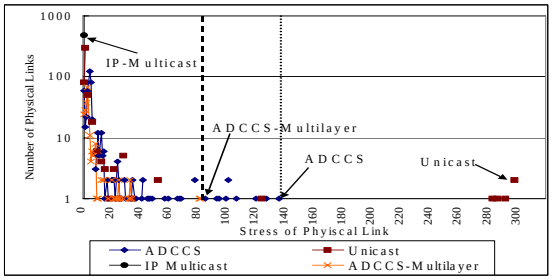

Fig. 7-d Physical link stress

5.2.2 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 ADCCS-Multilayer, ADCCS, IP Multicast and naïve unicast as shown in figure 7-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 ADCCS-Multilayer, ADCCS and naïve unicast, most links have a small stress-this to be expected. However, the significant lies in the tails of the plots. Under naïve unicast, one link has stress 299. This because that links near the source have high stress. However, ADCCS-Multilayer and ADCCS distribute the stress more evenly across the physical links. ADCCS-Multilayer has about 73% improvement over naïve unicast. ADCCS has about 55% improvement over naïve unicast. We argue that ADCCS-Multilayer has high improvement ratio than ADCCS to the multi-layer structure that is aware with the latency among the community nodes.

6. Conclusion

This paper presents the step-step construction and maintenance technique of the latency awareness ADCCS-Multilayer structure. This structure reduces both the communication delay among community nodes and the join overhead. Furthermore, it presents the simulation results that show the effectiveness of the proposed technologies.

References

[1] K. Mori, Et. al., "Proposition of Autonomous Decentralization Concept," Journal of IEE Japan, Vol. 104, no. 12, 1984, pp. 303-310 (Japanese).

[2] K. Mori, "Autonomous Decentralized Systems: Concept, Data Field Architecture and Future Trends," Proc. IEEE ISADS, Japan, 1993.

[3] K. Mori, et al., "Autonomous Decentralized Software Structure and its Application," Proc. IEEE FJCC'86, Nov. 1986.

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

[5] K. Ragab, Et al., "ACIS: A large-scale Autonomous Decentralized Community Communication Infrastructure," IEICE Transaction on Info. Sys. Vol. E87-D, No. 4, April 2004.

[6] T. Ono Et. al., "Service-oriented Communication Technology for Achieving Assurance," 22nd ICDCS (ADSN workshop) IEEE CS Press, 2002, pp. 69-74.

[7] K. Ragab Et al., "A Novel Hierarchical Community Architecture with End-to-End Delay Awareness for Communication Delay Enhancement", Proc. IEEE/IPSJ SAINT, Jan., 2004, Tokyo, Japan.

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

[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] J. Jannotti, Et al., "Overcast: Reliable Multicasting with as n Overlay Network," In Proc. 4th Symp. OSDI, Oct. 2000.

[11] D. Pendarakis, Et al., "ALMI: an application level multicast infrastructure", In Proc. of 3rd Symp. USITS, March 2001.

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

[13] E. W. Zegura, Et al., "How to model an internetwork" in Proc. of IEEE Infocom, 1996, San Francisco.