

An End Host Multicast Protocol for Peer-to-Peer Networks *

Wanqing Tu and Weijia Jia

Department of Computer Science, City University of Hong Kong,

Hong Kong SAR, P. R. China

tu.wanqing@student.cityu.edu.hk, itjia@cityu.edu.hk

Abstract

This paper presents a novel end host multicast protocol (DSM) on top of the mesh overlays for P2P networks. Unlike the previous schemes, the major feature of DSM is that it does not need a multicast tree for routing the multicast messages. DSM can implement the scalable and efficient multicast communications in a fully distributed way and consists of the following algorithms: 1. cluster formation that forms the group into clusters and each cluster consists of a small number of members; 2. cluster core selection that selects a core for each cluster who has the minimum sum of overlay hops to all other cluster members; 3. multicast routing that includes the balanced RA routing approach which enables a cluster core to construct the connections with other cores dynamically by using the rectangle area and the balanced forward flooding scheme in each cluster which evenly distributes the data traffic and link stress among the links in the multicast system. Simulation results show that DSM is fully distributed, scalable and efficient as compared with some well-known end host multicast systems.

1. Introduction

Peer-to-peer (P2P) computing is the system of mutually exchanging information and services directly between a sender and a receiver. It permits any two peers to communicate with each other in such a way that either ought to be able to initiate the communications. Therefore, typical P2P applications span the content delivery, file-sharing, collaboration and user(s)-to-user(s) multimedia communications. Multicast is a scalable way to support these applications. End hosts in P2P networks construct the overlay networks that create the convenience of implementing multicast in the application layers, which is called the *end host multicast* by many researchers. An end host multicast is such a data structure that provides multicast functionality in the application layer through unicast connections among group members in the underlying layers. Compared to the network layer multicast, the

major difference of end host multicast is that packets are replicated and forwarded by end hosts instead of intermediate multicast routers. This property makes the implementation of end host multicast more practicable because the deployment of end host multicast can be fully independent of real network architectures.

However, packet forwarding and replicating at the end hosts may incur some performance penalty as compared with the network layer multicast. The end host multicast is less scalable in terms of group size than the network layer multicast because the same underlying links (especially the underlying links between the sources and their adjacent intermediate network components, such as the hubs) are occupied by the identical packet copies more than one time. It is called *link stress* by many researchers. Link stress shows that the network resources (e.g., bandwidth) are excessively consumed and therefore the number of members that can join in the multicast group is limited. Many end host multicast protocols [5-14] have been proposed in recent years, and several well-known end host multicast protocols [5,8] have been devoted to addressing this problem. However, in these protocols, the underlying links used to connect the senders or packet forwarders with their nearest intermediate network components are apt to suffer from “bottleneck” as multiple sources coexist. “Bottleneck” decreases the multicast ability to scale to a large group size. A general way to improve the scalability is to decrease the control traffic. It will reserve the network capability to enable more new end hosts to join in the multicast system. S. Ratnasamy *et al.* [10] adopts an improved flooding scheme — CAN-based multicast to distribute the multicast packets. It has been proven in [10] that such flooding scheme is an effective way to enable end host multicast to scale to a very large group especially when multiple sources coexist. It is because the flooding method evenly distributes the link stress and data traffic among the overlay links in the multicast system. However, the flooding scheme neglects the end host heterogeneities in the output capacities and incurs the inefficient communication in terms of multicast delay. In the end host multicast, traffic is forwarded by the end hosts in the group. If the multicast architecture is built up by assuming that all end hosts have the same output capacities, those end hosts with smaller output capacities are still apt to suffer from “bottleneck”. As for the multicast delay, it takes quite a long time to transmit the packets

* The work is partially supported by Research grant council (RGC) Hong Kong, SAR China under grant Nos CityU 1055/00E (9040687) and CityU 1039/02E (9040596).

to the end hosts that are far away from the source by the flooding scheme. Most P2P applications (e.g., cooperative streaming media applications and queries in file-sharing applications) have requirements for the efficient communications. Hence, the end host heterogeneities and the efficiency should be taken into account in designing the end host multicast for P2P networks.

Apart from scalability and efficiency, P2P computing brings new challenges to the end host multicast. First, there is no “central server”. Each peer in P2P networks plays the role of sender and receiver at the same time. Thus, P2P networks are considered to be completely distributed and therefore the end host multicast for such networks should be fully distributed. Second, multiple sources coexist during the communications. The end host multicast should guarantee the “good” communication performances (i.e., the scalability and efficiency) in facing of multiple sources.

Most current end host multicast protocols [5-11] have a subset of the above properties. In this paper, our motivation is to design a novel distributed and scalable end host multicast protocol on top of mesh overlays for P2P networks. We name our protocol as DSM in which the letters are the initials of the words: distributed, scalable and multicast respectively. Unlike the previous schemes, the major feature of DSM is that it does not need a multicast tree for routing and delivering the multicast messages. DSM is a fully distributed protocol and consists of the following algorithms: 1. *cluster formation* that forms the group into clusters and each cluster consists of a part number of group members; 2. *cluster core selection* that selects a core for each cluster who has the *minimum* sum of *overlay hops* to all other cluster members; 3. *multicast routing* that includes the *balanced RA routing* approach which enables a cluster core to construct the connections with other cores dynamically by using the *rectangle area* instead of managing a multicast tree and the *balanced forward flooding* scheme within each cluster which evenly distributes the packets among all underlying links to enable the data traffic and link stress to be independent of the group size. Both the routing schemes are *balanced* in terms of end host output capacities and therefore the potential communication “bottleneck” is released.

The rest of the paper is organized as follows: Section 2 gives the design model. Section 3 details the design of DSM architecture. DSM *multicast routing* is discussed in Section 4. Section 5 evaluates the performances of DSM through the simulation observations. Section 6 concludes the paper.

2. Design Model

Let G be the multicast group with n end hosts such that $G = \{v_0, \dots, v_i, \dots, v_{n-1}\}$ ($i \in [0, n-1]$). Several studies [1-4] proposed schemes to construct the overlay networks and the problem of constructing a topologically-aware overlay was addressed in [12-14]. Our design supposes that the end hosts in the multicast group have been mapped into a m -D ($m \geq 2$) topologically-aware overlay CAN mesh which bears the relationship with the “closeness” metric by utilizing some P2P scheme [12]. A m -D

mesh is generated by partitioning a m -D Cartesian space among all end hosts in G such that every end host “owns” its individual, distinct zone within the overall space. Without loss of generality, we suppose that an end host “owning” a zone is mapped onto the central point of the zone. The location of end host v_i in the mesh is identified by m coordinates: $(V_{i,0}, \dots, V_{i,j}, \dots, V_{i,(m-1)})$ ($j \in [0, m-1]$). Several paths may exist between any two zones, but only the shortest path is employed for setup of the connection between these two zones. We use the number of *overlay hops* to measure the length of the shortest path between two zones. Denote the number of *overlay hops* between the two zones occupied by v_i and $v_{i'}$ ($i, i' \in [0, n-1], i \neq i'$) as $h(v_i, v_{i'})$, then the sum of *overlay hops* from v_i to all other group members $v_{i'}$ is defined as $h(v_i) = \sum_{i'=0, v_{i'} \neq v_i}^{n-1} h(v_i, v_{i'})$. In terms of the number of *overlay hops*, the lengths of the shortest paths between any two adjacent zones are with a uniform unit of 1. For instance, in Figure 1 (b), the numbers of *overlay hops* from 5 to 1, 5 to 2, and 5 to 4 are all 1. It follows that the lengths of the shortest paths between any two zones in the m -D mesh are k ($k \in \mathbb{N}$). Our theorems of *cluster core selection* in the paper are based on using the number of *overlay hops* to measure the shortest path length.

Apart from the shortest path, to construct the *balanced RA routing*, the *rectangle area* (RA) and the maximum number of direct receivers n_{v_i} of each end host v_i play the critical roles. We use a 2-D mesh to describe the RA. For any two nodes v_i in (X_0, Y_0) and $v_{i'}$ in (X_1, Y_1) , let $X_{min} = \min\{X_0, X_1\}$, $X_{max} = \max\{X_0, X_1\}$, $Y_{min} = \min\{Y_0, Y_1\}$ and $Y_{max} = \max\{Y_0, Y_1\}$, then all nodes (x, y) such that $X_{min} \leq x \leq X_{max}$ and $Y_{min} \leq y \leq Y_{max}$ uniquely construct a RA $[v_i, v_{i'}]$. As for n_{v_i} , considering a multicast flow with the rate r , let v_i 's available output capacity be OC_i , then we have $n_{v_i} = \lfloor \frac{OC_i}{r} \rfloor$.

3. DSM End Host Multicast Architecture

To construct DSM architecture, we assume the existence of an *agent set* $AS = \{ag_0, \dots, ag_{k-1}\}$. Suppose AS has an associated DNS domain name. The DNS is resolved to the agent (say $ag_l, l \in [0, k-1]$) that is the closest one to the issued end host by some anycast routing scheme [16] (in this paper, the closest agent is in terms of unicast distance). The function of each agent is to register the multicast group for the end hosts that send requests to it. A member list of the end hosts that have registered with ag_l is maintained by the agent. After the members join in the group, DSM organizes the end hosts into a two-layer architecture as shown in the example of Figure 1. Figure 1 (a) gives the original network in which the end hosts are mapped into the topologically-aware 2-D overlay mesh in Figure 1 (b) by some hashing function [12]. This overlay mesh is the lower layer of DSM architecture called the *member layer* (ML). In ML, end hosts are partitioned into different clusters as shown by the dotted lines in Figure 1 (b). Some selected end hosts join in the upper layer in Figure 1 (c) called the *core layer* (CL). More specifically, we make the detailed description in the following subsec-

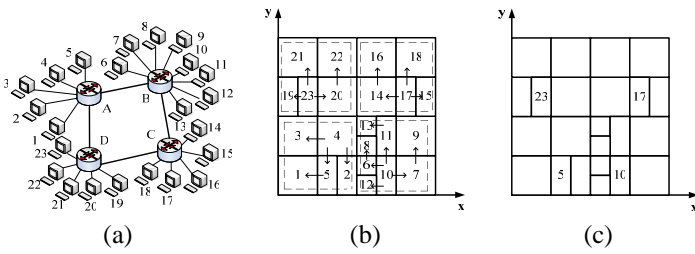


Figure 1. An example of DSM architecture.

tions.

3.1. DSM Architecture Construction

3.1.1. DSM Cluster Formation DSM employs the *cluster formation* to assign the end hosts in ML into several clusters. To select the cluster members, DSM defines the *member selection distance unit* D as the maximum number of *overlay hops* between the *constructor/sub-constructor* and the members selected by it; the *cluster size bound* S as the maximum number of end hosts that a cluster may contain. For the brevity of description, the following definitions are give.

- *Constructor*. It is an end host randomly chosen by an agent to initiate the current cluster construction. Each cluster has one and only one *constructor*.
- *Sub-constructor*. It is an end host assigned by the *constructor* to continue the current cluster construction only when the number of cluster members is less than the *cluster size bound* and the *constructor* cannot find any member with the *overlay hops* not greater than the *member selection distance unit*. In such situation, the cluster must have at least one *sub-constructor*.
- *Unassigned end host*. Define the end host that currently has not been assigned into any cluster as the *unassigned end host*.

We now sketch *DSM cluster formation*. One agent ag_l is initially selected through negotiating among the agents in AS . ag_l selects one *unassigned end host* v_i ($i \in [0, n - 1]$) at random from its member list. v_i becomes the *constructor* of the cluster, achieves the zone information of group members that are in the member list of ag_l , and then starts the cluster construction by selecting the *unassigned end hosts* in G with the numbers of *overlay hops* not greater than D as its cluster members. Algorithm 1 gives the distributed *cluster member selection* procedure.

If v_i finishes its *cluster member selection* due to v_i finds a member v_j with $h(v_i, v_j) > D$, a cluster member selected by v_i (i.e., a *sub-constructor*) will continue the current cluster construction by Algorithm 1. v_i informs ag_l of this *sub-constructor*. ag_l updates the information with other agents subsequently. The formation of current cluster terminates till all of the cluster members cannot find any *unassigned end host* who is within D over-

Algorithm 1 Distributed Cluster Member Selection

Input: The selected agent ag_l , the *constructor/sub-constructor* v_i ($i \in [0, n - 1]$), the current cluster size cur_size in ag_l , S and D ; // cur_size is 0 if v_i is the *constructor*

Output: Cluster member set $CM = \{\}$;

1. ag_l informs v_i of cur_size ;
2. If (v_i is the *constructor*) { ag_l puts v_i into SCM ; v_i sets $cur_size = 1$; }
3. For ($j = 0$ to $(l - 1)$ & $j \neq i$) do { // l members listed in ag_l
4. If ($h(v_i, v_j) \leq D$ & $cur_size < S$) {
5. v_i sends *Invitation*($cluster_id$) to v_j ;
6. If (v_j is *unassigned*) { v_j sends *Response*($cluster_id$) to v_i ; v_i puts v_j into CM and sets $cur_size = cur_size + 1$; }
7. Else { v_j sends *Negative*($cluster_id$) to v_i ; }
8. If (v_i detects $h(v_i, v_j) \leq D$ for all members v_j listed in ag_l & $cur_size < S$) { v_i continues selecting cluster members in the member list of the next closest agent to v_i by the same way to select v_j ; }
9. Else { v_i finishes its member selection and sends cur_size to ag_l . }

lay hops or the number of selected cluster members equals to S . Then, DSM initiates another cluster formation among the *unassigned end hosts*. When there is no *unassigned end host* in G , *DSM cluster formation* completes. We give the distributed *cluster formation algorithm* as below

Algorithm 2 Distributed Cluster Formation

Input: Unassigned end host set $UEH = \{v_0, \dots, v_i, \dots, v_{n-1}\}$;

Output: Cluster set $CS = \{\}$;

1. While ($UEH \neq \phi$) do {
2. An agent ag_l is selected through negotiating among the agents in AS ; ag_l randomly chooses an *unassigned end host* v_i in its member list, sends *Constructor*($cluster_id$) to v_i , removes v_i from UEH and updates the information with other agents;
3. Upon receiving *Constructor*($cluster_id$), v_i selects p cluster members by Algorithm 1, puts them into SCM , and asks ag_l to remove them from UEH ; ag_l updates such change with other agents;
4. If (v_i finishes its member selection & $cur_size < S$) { v_i selects a *sub-constructor* from the p cluster members and informs ag_l of the *sub-constructor*; The *sub-constructor* continues the cluster member selection by Algorithm 1; }
5. If ($cur_size == S$ || any member in the cluster cannot find an outside group member who is within the *overlay hops* of D) { v_i sends *Complete*($cluster_id$) to ag_l and puts the constructed cluster into CS ; ag_l updates the information with other agents. }

DSM adopts a distributed way to construct the clusters. Although each *constructor/sub-constructor* needs to calculate several numbers of *overlay hops* and compares these results with D to decide the member selection, such calculation is a linear calculation, and only the numbers of *overlay hops* to a subset of group members are calculated. If v_i detects that its number of *overlay hops* to a cluster member in ag_l 's member list exceeds D , it will not conduct the *cluster member selection* with those group members listed in an agent that is farther away from v_i than ag_l . The value of D guarantees that only close end hosts can be assigned into the same cluster.

3.1.2. DSM Cluster Core Selection In DSM, each cluster has a cluster core that constructs the CL of DSM architecture. The criteria used to select the cluster core is: *the cluster core has the minimum sum of overlay hops to all other cluster members*. The following theorem gives the *sufficient and necessary* condition for seeking such a core in a 2-D mesh.

Theorem 1 Suppose end host u occupies the zone (X, Y) in the topologically-aware 2-D CAN mesh and let $n_{>X}$, $n_{<X}$ and $n_{=X}$ ($n_{>Y}$, $n_{<Y}$ and $n_{=Y}$) be the numbers of members whose corresponding coordinates are larger than, less than and equal to X (Y) respectively. End host u is a core if and only if the following two inequalities hold simultaneously:

$$|n_{>X} - n_{<X}| \leq n_{=X}, |n_{>Y} - n_{<Y}| \leq n_{=Y} \quad (1)$$

Proof (Sufficient condition): CAN employs a uniform way to partition and merge the mesh zones, and such partition and merging is done by assuming along a certain dimension order. Hence, there is no large difference in the split among all mesh zones. We now prove the *sufficient condition*. Suppose end host u in the zone (X, Y) is a core, then for any other end host $u^{(1)}$ in the mesh, there exists $h(u) \leq h(u^{(1)})$. To achieve the first inequality in (1), we first consider an end host $u^{(1)}$ in (X_1, Y_1) and its sum of *overlay hops* to all other cluster members $h(u^{(1)})$. Let $u^{(1)}$ be in the closest adjacent zone of u at u 's right side (i.e., $u^{(1)}$ has the minimum x coordinate among all the members in the zones at u 's right side). Given any member u_i in (X_i, Y_i) , three cases are considered: (a) if $X < X_i$, the hops from u_i to u are equal to or one unit larger than the hops from u_i to $u^{(1)}$; (b) if $X > X_i$, the hops from u_i to u are one unit less than the hops from u_i to $u^{(1)}$; (c) if $X = X_i$, the hops from u_i to u are one unit less than the hops from u_i to $u^{(1)}$.

Because there exist $(n_{<X} + n_{=X})$ members whose x coordinate values are less than or equal to X , and $n_{>X}$ members whose x coordinate values are larger than X , we have

$$\begin{aligned} 0 \leq h(u^{(1)}) - h(u) &= \sum_{i=0, u_i \neq u^{(1)} \neq u}^{n'-1} [h(u^{(1)}, u_i) - h(u, u_i)] \\ &= n_{<X} + n_{=X} - n_{>X} \Rightarrow n_{>X} - n_{<X} \leq n_{=X} \end{aligned}$$

where n' is the number of cluster members.

Let $u^{(2)}$ be in the closest adjacent zone (X_2, Y_2) of u at u 's left side (i.e., $u^{(2)}$ has the maximum x coordinate among all the members in the zones at u 's left side), $u^{(3)}$ be in (X_3, Y_3) that is the closest adjacent zone above u (i.e., $u^{(3)}$ has the minimum y coordinate among all the members in the zones above u), and $u^{(4)}$ be in (X_4, Y_4) that is the closest adjacent zone below u (i.e., $u^{(4)}$ has the maximum y coordinate among all the members in the zones below u). By comparing $h(u^{(2)})$ with $h(u)$ in the same way as above, we can achieve the first inequality in (1). By comparing $h(u^{(3)})$, $h(u^{(4)})$ with $h(u)$ separately, we can achieve the second inequality in (1) similarly.

(Necessary condition): It is easy to demonstrate that if one of the inequalities in (1) is violated, end host u in (X, Y) cannot be

the core. Assume $n_{>X} - n_{<X} > n_{=X}$, then $n_{>X} > n_{<X} + n_{=X}$. It means that the number of end hosts with the x coordinates greater than X is more than the other two cases. Thus the *overlay hops* from u to these end hosts are larger than some other end hosts, a desired contradiction. Q.E.D.

Now we extend Theorem 1 to a m -D CAN mesh network.

Theorem 2 Let $(U_0, U_1, \dots, U_{(m-1)})$ represents any zone in a m -D CAN mesh network that is occupied by end host u and $n_{>U_j}$, $n_{<U_j}$ and $n_{=U_j}$ be the numbers of members with the j -th coordinates larger than, less than and equal to U_j respectively. Then end host u is the core if and only if the following m inequalities hold simultaneously:

$$|n_{>U_j} - n_{<U_j}| \leq n_{=U_j}, j = 0, 1, \dots, m-1 \quad (2)$$

The proof of Theorem 2 is similar as the one of Theorem 1 and thus is omitted here. We now sketch *DSM cluster core selection*. In *DSM cluster formation*, the *constructor* and *sub-constructors* achieve the zone information of the cluster members selected by them respectively from the agents. The *constructor* collects the members' zone information in the *sub-constructors* and then selects the cluster core according to the theorems. If there is more than one member who meets the theorem conditions, a random one is selected as the cluster core. Others become the backup cluster cores in case the leave/failure of current cluster core. The cluster core informs the closest agent of its state information. The agent set will update the information consistently. Suppose the cluster has n' members. We denote the coordinates of cluster member u_i as $(U_{i,0}, \dots, U_{i,j}, \dots, U_{i,(m-1)})$, $i \in [0, n'-1]$, $j \in [0, m-1]$ and the coordinates of cluster core u^* as $(U_0^*, \dots, U_j^*, \dots, U_{(m-1)}^*)$, where $U_j^* \in [(U_j)_{min}, \dots, (U_j)_{max}]$ and $(U_j)_{min} = \min\{U_{0,j}, U_{1,j}, \dots, U_{(n'-1),j}\}$, $(U_j)_{max} = \max\{U_{0,j}, U_{1,j}, \dots, U_{(n'-1),j}\}$. Algorithm 3 gives the *cluster core selection algorithm*.

In Algorithm 3, steps 1-4 can be executed in time $O(n')$. Steps 5-8 is a simple linear procedure. It is also executed in time $O(n')$. We can improve steps 5-8 by using the binary searching algorithm that yields an $O(\ln(n'))$ complexity. But for brevity of discussion, we keep the linear search algorithm here. According to the algorithm, the cores of the clusters in Figure 1 (b) are 5, 10, 17 and 23 respectively.

3.1.3. Group Maintenance Similar to other protocols [4-9,14], DSM employs the periodic refresh messages to follow the membership alterations. End hosts periodically exchange the refresh messages with their cluster neighbors in ML, and cluster cores also update their state information periodically with their neighbors in CL. Table 1 gives the comparison of the worst control overheads generated by one end host in NARADA, NICE with the cluster size of s , CAN-based multicast and DSM in the m -D regular mesh overlay when there are n members in the multicast group. Generally, $m < s$ and $m \ll n$ for a large group. When the membership alterations (e.g., join/departure/failure) occur, DSM

Algorithm 3 Cluster Core Selection in m -D Mesh Networks

Input: Cluster member set $CM = \{u_0, \dots, u_i, \dots, u_{n'-1}\}$, the *constructor* v and v 's closest agent ag_i ;

Output: Cluster core $u^* = (U_0^*, \dots, U_j^*, \dots, U_{(m-1)}^*)$;

1. ag_i informs v of the *sub-constructors*; v sends $Req_Member(cluster_id)$ to each *sub-constructor* who then responds with the zone information of cluster members selected by it through $Ack_Member(cluster_id)$;
 2. v initializes $\{a_{(U_j)_{min}}, \dots, a_{(U_j)_t}, \dots, a_{(U_j)_{max}}\} = \{0, \dots, 0, \dots, 0\}$; // $a_{(U_j)_t}$ records the number of cluster members whose j -th coordinates equal to $(U_j)_t$, where $t \in N$
 3. For $(i = 0$ to $n' - 1)$ v do {
 4. If $(u_i$'s j -th coordinate $U_{i,j} == (U_j)_t \{a_{(U_j)_t} = a_{(U_j)_t} + 1;\}$
 5. For $(i = 0$ to $n' - 1)$ v do {
 6. For $(j = 0$ to $m - 1)$ v do {
 7. If $(|\sum_{l=(U_j)_{min}}^{(U_j)_t} (a_l) - a_{(U_{i,j})}| - |\sum_{l=(U_{i,j})}^{(U_j)_{max}} (a_l) - a_{(U_{i,j})}|) \leq a_{(U_{i,j})} \{(U_j^*) = (U_{i,j}); j = j + 1;\}$
 8. Else $\{j = m - 1; i = i + 1;\}$
-

	NARADA	NICE	CAN-based multicast	DSM
Control Overhead	$O(n)$	$O(s)$	$O(m)$	$O(m)$

Table 1. Control overhead comparison.

dynamic group management is employed to restore the normal multicast communications. We present the details of DSM *dynamic group management* in [18].

4. DSM Dynamic Multicast Routing

DSM adopts different data routing schemes in ML and CL. For the short delay multicast communications, packets are transmitted *in parallel* in these two layers. Denote \tilde{n}_v as the number of cluster neighbors (i.e., the cluster members in the zones with 1 *overlay hop* to v) of any cluster member v . In ML, if the cluster core c satisfies $\tilde{n}_c < n_c$, it forwards the outside packets to all its cluster neighbors; otherwise, it only forwards the packets to the closest $(n_c - 1)$ neighbors for reserving the capacity to send packets to 1 direct receivers in the upper layer. Those $(\tilde{n}_c - n_c + 1)$ neighbors may receive the packets from their other neighbors. The neighbors received packets from c continue forwarding the packets to their neighbors who haven't received the packets before by the same way above. Such packet forwarding scheme is called the *balanced forward flooding*. The arrow lines in Figure 1 (b) depict the multicast routing paths in each cluster by assuming that each end host v is with $n_v \geq 4$. As for the source cluster, the source s not only sends the packets to its \tilde{n}_s neighbors if $n_s > \tilde{n}_s$ or $(n_s - 1)$ neighbors if $n_s \leq \tilde{n}_s$ but also sends the packets to the cluster core directly to speed up the message transmission to other clusters. For example, in Figure 1 (b), suppose end

host 4 is the source. It forwards the packets to its cluster neighbors (i.e., end hosts 2 and 3) and the cluster core (i.e., end host 5) simultaneously as illustrated by the arrow lines. The procedure goes on until all the cluster members receive the packets. The *balanced forward flooding* scheme distributes the data traffic evenly over all links and the maximum out-degree of each cluster member is a constant. The potential "bottleneck" when multiple sources coexist is released. Moreover, such flooding is efficient in terms of packet forwarding times because the cluster core has the *minimum* sum of *overlay hops* to other cluster members. Therefore, the communication efficiency is improved as analyzed in [14].

To construct the delivery paths in CL, we design the *balanced Rectangle Area (RA) routing algorithm*. We first define the *forwarder* and *successor*. If a cluster core c_i receives packets from another cluster core $c_{i'}$ ($i, i' \in [0, n'' - 1], i' \neq i$), $c_{i'}$ is the *forwarder* of c_i and c_i is the *successor* of $c_{i'}$. To avoid the same packets being transmitted to the same end host more than one time, each cluster core only has one *forwarder*. Recall each cluster core declares its existence with its closest agent. The *balanced RA routing algorithm* is described as follows. The source cluster (i.e., the cluster that the source belongs to) core c_s ($s \in [0, n'' - 1]$) achieves the cluster core list from its closest agent ag_i and selects the cluster core c_f ($f \in [0, n'' - 1], f \neq s$) with the maximum number of *overlay hops* to c_s . If there is more than one such cluster cores, c_s randomly chooses one of them. The *rectangle area* $[c_s, c_f]$ between c_s and c_f is constructed. $[c_s, c_f]$ may cover the mesh zones that several cluster cores locate in. If so, there exists a cluster core $c_{f'}$ ($f' \in [0, n'' - 1], f' \neq s, f' \neq f$) in $[c_s, c_f]$ who has the maximum number of *overlay hops* to c_s . $c_{f'}$ forms $[c_s, c_{f'}]$ which is called the *inner RA* of $[c_s, c_f]$ that can be denoted as $[c_s, c_{f'}]_{\{c_f\}}$. An *inner RA* of $[c_s, c_f]$ is defined as the *RA* that is constructed by the source cluster core and one cluster core covered by $[c_s, c_f]$. An *inner RA* of $[c_s, c_{f'}]$ is also an *inner RA* of $[c_s, c_f]$ if $[c_s, c_{f'}]_{\{c_f\}}$ exists. Let the current number of *successors* of $c_{f'}$ be $\hat{n}_{c_{f'}}$. If $\hat{n}_{c_{f'}} + \tilde{n}_{c_{f'}} < n_{c_{f'}}$, $c_{f'}$ becomes the *forwarder* of c_f ; otherwise, the path $(c_{f'} \rightarrow c_f)$ will not be set up. c_s continues constructing the *inner RA* of $[c_s, c_{f'}]$ in the same way. The member $c_{f''}$ used to construct the first *inner RA* of $[c_s, c_{f'}]$ is the *forwarder* of $c_{f'}$ if $\hat{n}_{c_{f''}} + \tilde{n}_{c_{f''}} < n_{c_{f''}}$. When a *RA* only contains two items: c_s and a cluster core, the delivery path from c_s to this cluster core is constructed if $\hat{n}_{c_s} + \tilde{n}_{c_s} < n_{c_s}$. After that, c_s continues constructing *RAs* with those members who have not been covered by the generated *RAs*. When all the cluster cores have been covered by at least one *RA*, one *forwarder searching procedure* completes. There may several cluster cores called the *unassigned cluster cores* who haven't found their *forwarders* in this procedure. The *balanced RA routing* begins another *forwarder searching procedure* to construct the *balanced RA* paths among them. In the new procedure, c_s is still the source cluster core if $\hat{n}_{c_s} + \tilde{n}_{c_s} < n_{c_s}$; otherwise, ag_i assigns $c_{s'}$ who is the closest *unassigned cluster core* to c_s to act as the source cluster core. $c_{s'}$ receives packets from $c_{s''}$ who is $c_{s'}$'s closest core that sat-

ifies $n_{c_s} + n_{c_{s'}} < n_{c_{s''}}$. The *balanced RA* delivery path construction completes till all the cluster cores find their *forwarders*. Algorithm 4 gives the *balanced RA routing algorithm*.

Algorithm 4 Balanced Rectangle Area Routing

Input: Cluster core set $CC = \{c_0, c_1, \dots, c_{n''}\}$, source cluster core c_s , and c_s 's closest agent agl ; n'' is the number of clusters in G

Output: Delivery path set $P = \{\}$;

1. While ($CC \neq \Phi$) {
 2. If ($n_{c_s} + n_{c_{s'}} \geq n_{c_{s''}}$) { agl selects $c_{s'}$ who is c_s 's closest cluster core and with $n_{c_{s'}} + n_{c_{s''}} < n_{c_{s''}}$; $c_s = c_{s'}$; }
 3. c_s achieves the cluster core list from its closest agent and constructs $[c_s, c_f]$ with the cluster core c_f who has the maximum number of *overlay hops* to c_s ; Cluster cores not covered by $[c_s, c_f]$ form the set $\overline{C}([c_s, c_f])$;
 4. While ($|\overline{C}([c_s, c_f])| > 2$) do { $|\overline{C}([c_s, c_f])|$ records the number of cluster cores in $\overline{C}([c_s, c_f])$
 5. c_s constructs $[c_s, c_{f'}]$ with the cluster core $c_{f'}$ who has the maximum number of *overlay hops* to c_s ; Cluster cores in $[c_s, c_{f'}]$ but not covered by $[c_s, c_{f'}]$ form $\overline{C}([c_s, c_{f'}])$;
 6. If (c_f is *unassigned* & $n_{c_{f'}} + n_{c_{f''}} < n_{c_{f''}}$) {Put ($c_{f'} \rightarrow c_f$) into P ; $\overline{C}([c_s, c_{f'}]) = \overline{C}([c_s, c_{f''}])$; $f = f''$; }
 7. If ($n_{c_s} + n_{c_{s'}} < n_{c_{s''}}$) {Put ($c_s \rightarrow c_f$) into P ; Remove c_f from CC ; Construct the delivery paths among the cluster cores in $\overline{C}([c_s, c_{f'}])$ by the same way to build the paths in $[c_s, c_{f'}]$ if $\overline{C}([c_s, c_{f'}]) \neq \phi$;
 8. If ($\overline{C}([c_s, c_{f'}]) \neq \phi$) { Constructs the delivery paths among the cluster cores in $\overline{C}([c_s, c_{f'}])$ by the same way to build the paths in $[c_s, c_{f'}]$. }
-

The *balanced RA routing* eliminates the protocol cost to manage a multicast tree and achieves the similar routing efficiency as the tree routing. The algorithm enables a cluster core to construct the connections with other cluster cores dynamically through a set of shortest overlay links among the cluster cores and based on the end host output capacities that releases the potential ‘‘bottleneck’’. The short multicast delay and scalable communications are therefore guaranteed. Figure 2 shows an example of constructing the *RA* delivery paths. The number of arrow lines indicates the packet forwarding times from the source cluster core to the current cluster core. In this example, we suppose each cluster core has the output capacity to serve 5 direct receivers and A is the source cluster core. The coordinates of A, B, C, D, E and F are $(0.5, 1.25), (0.5, 1.75), (1.25, 1.5), (1.75, 1.5), (0.5, 0.5)$ and $(1.5, 0.5)$ respectively. Based on step 3, A selects F with the maximum number of *overlay hops* to construct $[A, F]$ that only covers E as shown in Figure 2 (a). The dotted lines in Fig. 2 (a) mean that C and D are not covered by $[A, F]$ and therefore are put into the set $\overline{C}([A, F])$. Then the algorithm goes to steps 5 and 6 where E is selected to be F 's *forwarder* and constructs $[A, E]$. After that, the algorithm goes to step 7 due to only two items in $\overline{C}([A, E])$. The delivery path ($A \rightarrow E$) is constructed. Then, in step 8, the algorithm builds the delivery paths among A, B, C and D as shown in Figure 2 (b) and Figure 2 (c) by the same way to

construct the paths among A, E and F . The completed *RA* delivery paths are illustrated in Figure 2 (c).

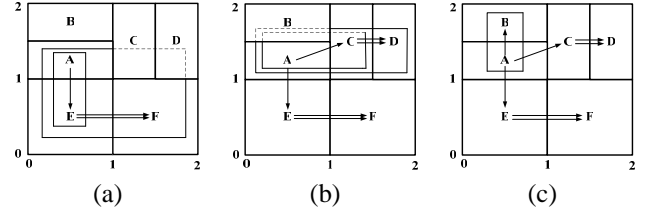


Figure 2. An example of constructing the *RA* delivery paths.

5. Experimental Evaluations

5.1. Experimental Model and Evaluation Criteria

We use *ns-2* [15] to run the simulations on a group of SUN SOLARIS workstations. The simulation backbone network is the combination of two MCI ISP backbones, which is shown in Figure 3. End hosts in the multicast group are connected to the routers in the backbone network directly or through some intermediate network components (e.g., the hubs). The link bandwidth in the backbone network and the local area network are 1000Mbps and 100Mbps respectively, and the propagation delays of links in the backbone network and the local area network are 2ms and 1ms respectively. The simulation traffic is the 1.5Mbps MPEG-1 video streams. We evaluate DSM and three well-known end host multicast protocols: NARADA, NICE and CAN-based multicast along the following criteria.

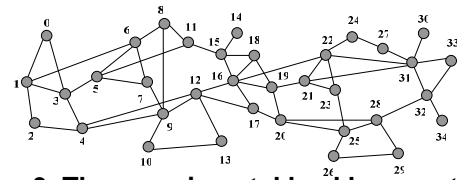


Figure 3. The experimental backbone network.

- Average Link Stress (ALS): Link stress is defined as the number of identical packet copies traversing over the same underlying physical links. Suppose L underlying links are used in the multicast communication and the number of identical packet copies traversing the i -th link is p_i , *ALS* is calculated by:

$$ALS = \frac{\sum_{i=1}^L p_i}{L} \quad (3)$$

- Average Multicast Delay (AMD): Define the multicast delay $d(s, v_i)$ from the source s to the group member v_i as the

end-to-end delay between them. Then, the average multicast delay from s to all the group members is

$$AMD = \frac{\sum_{i=0, v_i \neq s}^{n-1} d(s, v_i)}{n-1}, v_i \in G \quad (4)$$

- Total Number of Links Used (TNLU): It refers to the sum of links that are used during the multicast communication.
- Control Overhead (CO): It refers to the traffic generated by the group members during the membership management.

5.2. Simulation Results and Observations

Three simulations have been done to evaluate different protocols. Because NARADA is less scalable in terms of group size than other three protocols as illustrated by the results in Table 1 and the simulation results in Figure 6, we evaluate the scalability

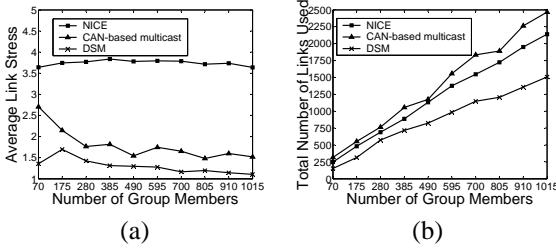


Figure 4. Performance comparison of link properties.

and efficiency of other three protocols in the first two simulations.

The first simulation observes the protocol performances when there is one source and the number of group members varies from 70 to 1015. Figure 4 (a) plots the *average link stress ALS* curves of the three protocols. The diagram shows that NICE has the worst *ALS* performance as compared with other two protocols. It is because that the cluster leaders in NICE forward the packets to all the members of the clusters that they belong to. The links near to the cluster leaders have to carry the identical packets more than several times. DSM has a better *ALS* than CAN-based multicast. It is because that the group members in DSM are partitioned into several clusters that enables the group members in the zones at the cluster edges to have less cluster neighbors. The reason for the decline trends of the three curves is that *ALS* is the ratio of the total link stress to the total used links and the increment of total link stress is less than the increment of total used links in this simulation. Figure 4 (b) gives the results of *total number of links used TNLU*. CAN-based multicast consumes the most physical links in order to complete the multicast communication. Although DSM adopts the similar scheme as CAN-based multicast in each cluster, the cluster formation and the *balanced RA routing* decrease the usage of links greatly.

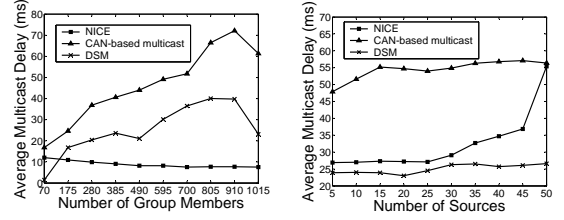


Figure 5. Average multicast delay performance comparison.

	NICE	CAN-based multicast	DSM
ALS	3.7786	1.4982	1.1428
TNLU	1130	1175	1072

Table 2. Comparison of link properties.

Figure 5 (a) shows the *AMD* curves of the three protocols. It can be seen that DSM has shorter *AMDs* than CAN-based multicast. Such trend becomes more obvious when the group size increases. The *AMD* of DSM is half of the one of CAN-base multicast when the number of group members is 910. It is mainly because the *RA* delivery paths in CL of DSM enable the packets to be transmitted to the remote end hosts within a short delay, and the packet flooding initiator in each cluster is the cluster core who has the *minimum* sum of *overlay hops* to other cluster members. The *AMD* decline trends in DSM and CAN-based multicast when the number of group members is larger than 805 is because the increment of the sum of multicast delay is less than the increment of the group members in this simulation. The figure also shows that NICE has the shortest *AMDs* in the three protocols. However, as shown in Figure 5 (b), when multiple sources coexist, NICE has longer *AMDs* than DSM.

The second simulation observes the protocol performances when there are 490 group members and the number of sources increases from 5 to 50. Figure 5 (b) plots the *AMD* curves. It can be seen that the flooding scheme in CAN-based multicast incurs much longer *AMDs* than the other two protocols. Under the situation of multiple sources, DSM achieves shorter *AMDs* than NICE, especially when a large number of sources coexist. Table 2 gives the link property comparison of the three protocols in this simulation. Data in the table show that DSM is more scalable than NICE and CAN-based multicast.

The last simulation compares the control overheads of the four protocols. There are 490 group members and 5 sending sources. The simulation period is 100 seconds and two different phases: a join phase and a leave phase are simulated. In the join phase, 72 new hosts join the group uniformly at random between the simulated time 30 and 48 seconds. Starting at the time of 60 sec-

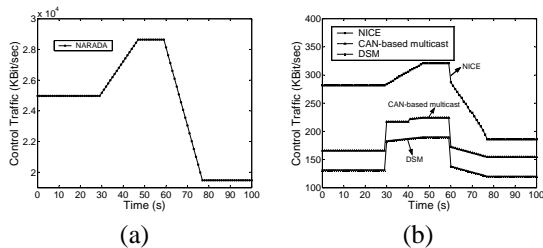


Figure 6. Control overhead comparison.

onds, 180 members leave the group uniformly. The leave procedure lasts 18 seconds. Figure 6 (a) gives the *CO* curve of NARADA. Compared with the results in Figure 6 (b), NARADA generates much greater control traffic than the other three protocols. The large control traffic compromises its scalability. Figure 6 (b) shows that NICE has larger control overheads than the other two protocols. Each cluster member needs to exchange the refresh messages with all other cluster members in NICE, while the group members only need to update their state information with their neighbors in other two protocols. Furthermore, DSM achieves better *CO* performances than CAN-based multicast because it decreases the control traffic used to maintain the membership of the members in different clusters.

6. Conclusion

This paper presented a scalable and efficient data structure — DSM end host multicast protocol for P2P networks by making full use of the properties of mesh overlays. With the *agent set*, all these operations are fully distributed. DSM adopts a two-layer architecture. In the lower layer, group members are assigned into different clusters by the *cluster formation*, and then the *balanced forward flooding* is employed to distribute packets within each cluster. These designs evenly distribute the link stress and data traffic among all the links in the multicast system. Therefore, the ability of DSM to scale to a large group size even when multiple sources coexist is improved. The upper layer is formed by the cluster cores of the lower layer. Each cluster core floods the packets to all its cluster members with the *minimum* sum of *overlay hops*. For the connections of different clusters, the *balanced RA* delivery paths based on the end host output capacities are constructed. Such paths release the potential bottleneck and speed up the packet transmission to the remote end hosts. Our simulation results match our theoretic analysis. In our future research, we are going to study and evaluate DSM in the area of robustness (e.g., failure recovery) by comparing it with some well-known multicast schemes, such as [17].

References

[1] S. Ratnasamy, P. Francis, M. Handley, and R. Karp, “A Scalable Content-Addressable Network”, *Proc. of ACM SIGCOMM 2001*,

pp. 161-172, August 27-31, 2001, San Diego, California, USA.

[2] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan, “Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications”, *Proc. of ACM SIGCOMM 2001*, pp. 149-160, August 27-31, 2001, San Diego, California, USA.

[3] A. Rowstron, and P. Druschel, “Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems”, Available at <http://research.microsoft.com/antr/PAST/>, 2001.

[4] B.Y. Zhao, J. Kubiatowicz, and A. Joseph, “Tapestry: An Infrastructure for Fault-Tolerant Wide-Area Location and Routing”, Available at <http://www.cs.berkeley.edu/ravenben/tapestry/>, 2001.

[5] H. Chu, S. Rao, S. Seshan, and H. Zhang, “A Case for End System Multicast”, *Proc. of ACM SIGMETRICS 2000*, pp. 1-12, June 17-21, 2000, Santa Clara, California, USA.

[6] P. FRANCIS, “Yoid: Extending The Internet Multicast Architecture”, available at <http://www.aciri.org/yoid/docs/index.html>, April, 2000.

[7] J. Jannotti, D. K. Gifford, K. L. Johnson, M. Frans Kaashoek, and J. W. O’Toole Jr., “Overcast: Reliable Multicasting with An Overlay Network”, *Proc. of The 4th Usenix Symposium on Operating Systems Design and Implementation*, October 22-25, 2000, Paradise Point Resort, San Diego, California, USA.

[8] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, “Scalable Application Layer Multicast”, *Proc. of ACM SIGCOMM*, pp. 205-217, August 19-23, 2002, Pittsburgh, Pennsylvania, USA.

[9] M. Castro, P. Druschel, A. M. Kermarrec, and A. Rowstron, “SCRIBE: A Large-Scale and Decentralized Application-level Multicast Infrastructure”, *IEEE Journal on Selected Areas in communications (JSAC)*, vol. 20, no. 8, pp. 1489-1499, October, 2002.

[10] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, “Application-Level Multicast Using Content-Addressable Networks”, *Proc. Of The 3rd International Workshop on Network Group Communication*, pp. 14-29, November 7-9, 2001, London, UK.

[11] B. Zhang, S. Jamin, and L. Zhang, “Host Multicast: A Framework for Delivering Multicast to End Users”, *Proc. of IEEE INFOCOM 2002*, pp. 1366-1375, June 23-27, 2002, New York, USA.

[12] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, “Topologically-Aware Overlay Construction and Server Selection”, *Proc. of IEEE INFOCOM 2002*, pp. 1190-1199, June 23-27, 2002, New York, USA.

[13] Z. Xu, C. Tang, and Z. Zhang, “Building Topology-Aware Overlays Using Global Soft-State”, *Proc. of ICDCS 2003*, pp. 500-508, May 19-22, 2003, Providence, Rhode Island, USA.

[14] W. Tu and W. Jia, “A Scalable and Efficient End Host Multicast for Peer-to-Peer Systems”, *Proc. of IEEE Globecom 2004*, pp. 967-971, November 29-December 3, 2004, Dallas, Texas, USA.

[15] UC Berkeley, LBL, USC/ISI, and Xerox PARC, “Ns Notes and Documentation”, October 20, 1999.

[16] D. Xuan, W. Jia, and W. Zhao, “Routing Protocols for Anycast Message”, *IEEE Transaction on Parallel and Distributed Systems*, pp. 571-588, Vol. 11, No. 6, June 2000.

[17] K. Birman, and R. Cooper, “The ISIS Project: Real Experience with A Fault Tolerant Programming System”, *ACM/SIGOPS European Workshop on Fault-Tolerance Techniques in Operating Systems*, Bologna, Italy, 1990.

[18] W. Tu, “Design of End Host Multicast Protocol on Mesh Overlays”, *Ph.D Dissertation, Chapter 4*, City University of Hong Kong, August 2005.