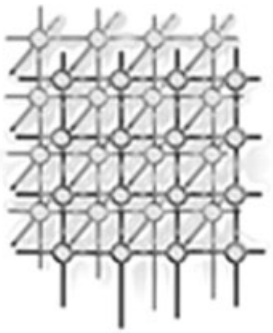
CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 2007; **19**:2013–2029
Published online 19 December 2006 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/cpe.1146



Distributed end-host multicast algorithms for the Knowledge Grid

Wanqing Tu[1,*,†], Jogesh K. Muppala[2] and Hai Zhuge[3]

[1]*Mobile and Internet Systems Laboratory, Department of Computer Science, University College Cork, Cork, Ireland*
[2]*Department of Computer Science, The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong*
[3]*The China Knowledge Grid Research Group, Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, People's Republic of China*

SUMMARY

The Knowledge Grid built on top of the peer-to-peer (P2P) network has been studied to implement scalable, available and sematic-based querying. In order to improve the efficiency and scalability of querying, this paper studies the problem of multicasting queries in the Knowledge Grid. An *m*-dimensional irregular mesh is a popular overlay topology of P2P networks. We present a set of novel distributed algorithms on top of an *m*-dimensional irregular mesh overlay for the short delay and low network resource consumption end-host multicast services. Our end-host multicast fully utilizes the advantages of an *m*-dimensional mesh to construct a two-layer architecture. Compared to previous approaches, the novelty and contribution here are: (1) cluster formation that partitions the group members into clusters in the lower layer where cluster consists of a small number of members; (2) cluster core selection that searches a core with the minimum sum of overlay hops to all other cluster members for each cluster; (3) weighted shortest path tree construction that guarantees the minimum number of shortest paths to be occupied by the multicast traffic; (4) distributed multicast routing that directs the multicast messages to be efficiently distributed along the two-layer multicast architecture *in parallel*, without a global control; the routing scheme enables the packets to be transmitted to the remote end hosts within short delays through some common shortest paths; and (5) multicast path maintenance that restores the normal communication once the membership alteration appears. Simulation results show that our end-host multicast can distributively achieve a shorter delay and lower network resource consumption multicast services as compared with some well-known end-host multicast systems. Copyright © 2006 John Wiley & Sons, Ltd.

Received 6 September 2006; Accepted 21 September 2006

KEY WORDS: end-host multicast; P2P networks; the Knowledge Grid; query

*Correspondence to: Wanqing Tu, Mobile and Internet Systems Laboratory, Department of Computer Science, University College Cork, Cork, Ireland.
†E-mail: wanqing.tu@cs.ucc.ie

Contract/grant sponsor: National Grand Fundamental Research 973 Program of China; contract/grant number: 2003CB317000
Contract/grant sponsor: Embark Postdoctoral Fellowship Grants of Ireland Government; contract/grant number: 501-et-504 4890
Contract/grant sponsor: Research Grants Council of Hong Kong SAR, China; contract/grant number: HKUST6177/04E

1. INTRODUCTION

The Knowledge Grid is a sustainable human–machine interconnection environment that enables people or agents to effectively generate, capture, publish, share, manage and promote knowledge, to process resources through machines and to transform resources from one form to another [1,2]. Peer-to-peer (P2P) networking is the system of mutually exchanging information and services between the sender and the receiver directly. It permits any two peers to communicate with each other in such a way that either peer is able to initiate communication. P2P technology offers the possibility of content capture, collaborative process and sharing in the Knowledge Grid environments. Zhuge [2,3] presented the scalable platform *IMAGINE-P2P* that supports path queries (i.e. queries are forwarded along paths of an index to find the matched keys) by publishing nodes of an index on a structured P2P overlay network.

Issuing queries is one of the most popular operations used to locate information in Knowledge Grid environments. Scalability, availability and sematic-based operations are three major challenges facing the Knowledge Grid [1,2]. *IMAGINE-P2P* [3] is dependent on the embedded P2P network to provide availability and sematic-based queries. In this paper, we propose the use of an end-host multicast (EMcast) protocol to distribute queries and content to achieve improved scalability.

End hosts in the P2P network of *IMAGINE-P2P* construct an overlay network that creates the convenience of implementing EMcast in the Knowledge Grid. An EMcast protocol is a data structure that provides multicast functionality in the application layer through unicast connections among group members in the underlying layers. The popular m-dimensional mesh networks are overlay networks constructed in structured P2P systems, such as content-addressable networks (CANs) [4]. Multicast solutions [5,6] based on regular mesh networks have been studied in the literature. However, in practical P2P systems, end hosts always construct irregular overlay mesh networks. NARADA [7] and CAN-based multicast protocol [8] are two well-known EMcast protocols for irregular mesh networks. However, both protocols have difficulties in providing scalable and efficient multicast performance, as we discuss later in the paper.

In this paper, our motivation is to provide distributed, short delay and low network resource consumption multicast communications on top of m-dimensional irregular overlay mesh networks. Several studies [4,9–11] have proposed schemes to construct overlay networks and the problem of constructing a topologically-aware overlay is addressed in [12–14]. Our design assumes that the end hosts in the multicast group have been mapped onto a topologically-aware overlay mesh, which shares a similarity with the 'closeness' metric by utilizing some P2P scheme [12]. A two-layer multicast architecture is constructed based on the properties of a topologically-aware mesh and through use of the agent set. Group members are partitioned into several clusters by *cluster formation* in the lower layer. The upper layer is composed of cluster cores that are selected by *cluster core selection* and have the minimum sum of overlay hops to other cluster members. The group members are connected through the trees constructed by the *weighted shortest path tree construction algorithm*. More specifically, our EMcast is fully distributed and consists of the following algorithms. (1) *Cluster formation* forms the group into clusters and each cluster consists of a part number of group members. Only the members with the 'closeness' metric may be assigned into the same cluster. The algorithm greatly limits the chances of message transmission among the members with long latencies and therefore the multicast delay and resource consumption caused by such message transmission are decreased. (2) *Cluster core selection* searches a core for the cluster with the minimum sum of overlay hops

DOI: 10.1002/cpe

to all other cluster members. (3) *Weighted shortest path tree construction* guarantees the minimum number of shortest paths to be occupied by the multicast traffic. (4) *Distributed multicast routing* directs the multicast messages to be efficiently distributed along the two-layer multicast architecture in parallel, without a global control. The packets can be transmitted to the remote end hosts within a short delay through some common shortest paths. (5) *Multicast path maintenance* restores the normal communication once the membership alteration appears.

The paper is organized as follows. Section 2 introduces the previous studies on multicast protocols in mesh networks. Section 3 details the architecture design of our EMcast that includes *cluster formation*, *cluster core selection* and *weighted shortest path tree construction*. Section 4 presents the *distributed multicast routing algorithm*. Section 5 introduces our operations to restore the membership alteration. Section 6 evaluates the performances of our EMcast through simulation observations. Section 7 concludes the paper.

2. RELATED WORKS

Currently, multicast routing protocols for m-dimensional mesh networks may be classified into two categories: protocols for m-dimensional regular mesh networks, and protocols on top of m-dimensional irregular mesh networks.

Double-channel XY multicast wormhole routing (DCXY) [5] uses a tree-based extension of an XY routing algorithm (i.e. routing the packets along the X and Y dimensions of mesh topology in turn) to set up the multicast paths within a group of nodes in a two-dimensional regular mesh. Dual-path multicast routing (DPM) [6] was also developed for two-dimensional regular mesh networks. It assigns a label to each node in the mesh and partitions the group into two subgroups, g_{h} and g_{l}, such that they are composed of the members with labels greater (g_{h}) or less (g_{l}) than the source label. The routing paths are constructed by connecting the nodes covered by g_{h} in ascending order of label values and the nodes covered by g_{l} in descending order of label values.

For the multicast routing in the irregular m-dimensional mesh networks, NARADA [7] distributively organizes the group members into an irregular overlay mesh topology, and then runs a protocol similar to the conventional DVRMP [15] to accomplish multicast routing among these members. To guarantee robustness, each group member in NARADA periodically sends refresh packets to each of the other group members, which compromises the scalability of NARADA. Hence, NARADA is only suitable for small-scale P2P systems. To improve the scalability, CAN-based multicast protocols [8] extend the well-known P2P substrate CAN overlaid on the Internet to implement the EMcast. It adopts the improved flooding method to multicast the packets to all of the group members. More specifically, each member forwards the packets to its mesh neighbors who have not received the packets previously. The flooding scheme distributes the data traffic evenly over all overlay links. Hence, CAN-based multicast protocols are able to scale to a very large group size even when it is employed by the multi-source multicast applications. However, the flooding scheme incurs a long delay to deliver the packets to the group members who are far away from the sender and also consumes a large amount of network resources. Usually, it is required that the queries should be distributed within short delays, and the multicast content in the Knowledge Grid are high-rate real-time streaming media that have stringent requirements for short delays and sufficient network resources. This provided the motivation to design the scalable and efficient EMcast on top of the m-dimensional mesh networks.

DOI: 10.1002/cpe

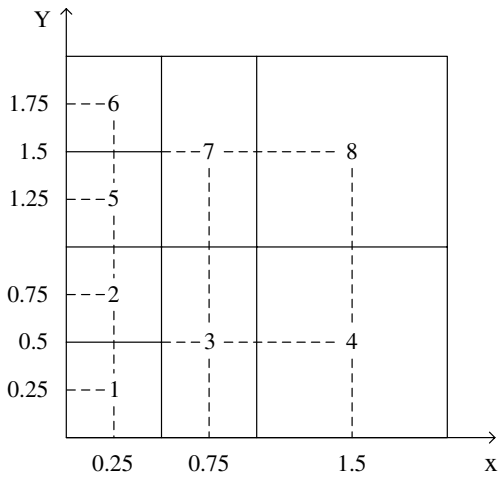


Figure 1. An example of a two-dimensional irregular mesh.

3. ALGORITHMS FOR EMCAST ARCHITECTURE

As mentioned in Section 1, our design assumes that the end hosts are assigned into the zones of a topologically aware m-dimensional overlay mesh network. Without loss of generality, we assume that an end host u 'owing' a zone is mapped onto the central point of the zone representing the zone, and is identified by m coordinates: $(U_0, U_1, \ldots, U_m)$. Several paths may exist between any two zones in the mesh network, but only the shortest paths are considered (based on the path weights introduced in Section 3.3) to be employed for setup of the connection between these two zones that are occupied by two group members. For an irregular mesh, the space sizes of the individual zones are different. Consequently, the central points that the end hosts are mapped onto are arbitrary points in the mesh network, as shown in Figure 1. For example, the coordinates of 1, 3, 5 and 7 are (0.25, 0.25), (0.75, 0.5), (0.25, 1.25) and (0.75, 1.5), respectively. In this paper, we use the number of overlay hops to measure the length of the shortest path between two zones as the m-dimensional mesh is regular in terms of overlay hops (see Figure 1). The lengths of the shortest paths between the two end hosts in two adjacent zones (e.g. end hosts 1 and 2, end hosts 1 and 3) are of uniform unit length of 1.

Our design also supposes the existence of an agent set with l agents $AS = \{ag_1, \ldots, ag_l\}$. Each agent set has an associated DNS domain name, and this resolves to the agent (say ag_i, $i \in [1, l]$) that is the closest agent to the issued end hosts. The function of each agent in AS 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_i is maintained by the agent. With the agent set, a set of distributed algorithms is proposed to construct a two-layer cluster EMcast architecture.

DOI: 10.1002/cpe

3.1. Cluster formation

We suppose that the current group members in $G = \{v_0, \ldots, v_i, \ldots, v_n\}$ are mapped into a topologically aware m-dimensional CAN mesh. Our EMcast employs the *cluster formation* to assign group members into several different clusters in the lower layer. To cluster the group members, the *member selection distance unit SH* is defined as the maximum number of overlay hops between the *constructor* (or *sub-constructor*) and the members selected by it; the *cluster size bound CS* is defined as the maximum number of end hosts that a cluster may contain. The following notions are defined for the description of *cluster formation algorithm*.

- *Constructor*. This is the end host who is the first to be randomly chosen to construct the cluster by an agent that is selected through negotiating among the agents in *AS*. Each cluster has only one *constructor*.
- *Sub-constructor*. This refers to the cluster member (excluding the *constructor*) that selects other cluster members during the cluster formation procedure after the *constructor* finishes its member selection. If the *constructor* finds the number of cluster members that reaches the cluster size bound, the cluster must have no *sub-constructor*; otherwise, the cluster must have at least one *sub-constructor*.
- *Unassigned end host*. We define the end host that has not as yet been assigned into any cluster as the *unassigned end host*.

To construct a cluster, an agent ag_i is selected through negotiation among the agents in *AS*. The agent randomly searches an *unassigned end host* u in its member list. This host becomes the *constructor* of the cluster, and receives the zone information of group members who are in the ag_i member list from ag_i. Then, u calculates the overlay hops denoted as $f(u, v)$ to the *unassigned* members v. End host v becomes the cluster member of u if $f(u, v)$ is not greater than *SH*; otherwise, v is not selected into the current cluster by u. To continue the member selection, the following three cases are considered.

- If the number of members in the current cluster is less than *CS* and $f(u, v)$ is not greater than *SH* for all of the members in the member list of ag_i, u continues its cluster member selection with the members listed in the next nearest agent to u.
- If the number of members in the current cluster is less than *CS* and u can find a member v in ag_i member list with $f(u, v) > SH$, u finishes its cluster member selection and assigns a cluster member selected by it (i.e. a *sub-constructor*) to continue the current cluster *construction* in the same way as u does.
- If the number of members in the current clusters is equal to or larger than *CS*, the current cluster construction completes. Then, another cluster construction among those *unassigned end hosts* is initiated by using the same method as above.

Our EMcast adopts a distributed method to construct the clusters. Although each *constructor* or *sub-constructor* needs to calculate several numbers of overlay hops and compares these results with *SH* to decide the member selection, the calculation is linear 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 listed in ag_1 exceeds *SH*, it will not select those group members who are listed in an agent that is farther away from v_i than the current agent to be its cluster members. Normally, in practical networks, the value of *SH* should guarantee that only close end hosts can be assigned into the same cluster.

DOI: 10.1002/cpe

For example, in our simulations, *SH* guarantees that only end hosts in the same local domain may be assigned into the same cluster. We set *SH* as the minimum value among the maximum overlay hop numbers of all local domains. On the other hand, the definition of *CS* should avoid the large difference in the cluster sizes. Here, similarly to NICE [16], we set *CS* as a random integer between k and $3k - 1$ where k is a constant determined by the application and group size. The maximum difference of two clusters is therefore $2k - 1$. In the simulations of NICE, k is set as 3. In addition, in our simulations, the anticipative performances with the number of group members exceeding 1000 are achieved when the maximum value of k is set as 5. Also, our programs show that the probability of choosing k and $3k - 1$ as the cluster size is low. It shows the smaller difference in the cluster sizes, and therefore balances the packet transmission in different clusters. Suppose that cluster A includes a large number of end hosts, and cluster B only contains a small number of end hosts. When the packets have been multicasted to all end hosts in cluster B, normally, there may be many end hosts in cluster A that have not yet received the multicast packets. The imbalance cluster transmission incurs a longer delay in cluster A. It is obvious that the multicast efficiency will be improved if some members in cluster A can be assigned into cluster B.

3.2. Cluster core selection

Clustering the group members limits the control traffic among the members in different clusters. Each cluster will have a cluster core that constructs the upper layer of the EMcast architecture. The cluster core is responsible for forwarding the packets from the outside group members to the inside cluster members and distributing the inside cluster packets to other clusters. Hence, the selection of cluster core determines the efficiency of our EMcast. The criteria used to select the cluster core is *select the cluster member as the cluster core who has the minimum sum of overlay hops to all other cluster members*. In this section, we first propose the theorems of searching such a core in a CAN mesh, then the *cluster core selection algorithm* is presented.

We define $f(u)$ as the sum of *overlay hops* from a cluster member u to all other cluster members u_i, namely, $f(u) = \sum_{i=1, u_i \neq u}^{n'} f(u, u_i)$, where n_0 is the cluster size and $f(u, u_i)$ is the number of *overlay hops* between u and u_i. The following theorem gives the *sufficient* and *necessary* condition for seeking a core in a two-dimensional irregular mesh.

Theorem 1. *Suppose end host u occupies the zone* (X, Y) *in the topologically aware two-dimensional 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. The end host u is a core if and only if the following two inequalities hold simultaneously:*

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

Proof (Sufficient condition). Suppose that the end host u in the zone (X, Y) is a core, then for any other end host $u^{(1)}$ in the mesh, there exists $f(u) \leq f(u^{(1)})$. 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 $f(u^{(1)})$. Let $u^{(1)}$ be in the nearest adjacent zone of u on the right-hand side of the a (i.e. $u^{(1)}$ has the minimum x coordinate among all of the members in the zones at the right-hand side of u). In a CAN mesh network, CAN utilizes a uniform way to partition and merge the mesh zones, and such partitioning and merging is done by assuming a certain dimension order. Hence, there is no large difference in the split among all

DOI: 10.1002/cpe

mesh zones. In this case, 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

$$0 \le f(u^{(1)}) - f(u) = \sum_{i=1, u_i \neq u^{(1)} \neq u}^{n'-1} [f(u^{(1)}, u_i) - f(u, u_i)]$$
$$= n_{<X} + n_{=X} - n_{>X} \Rightarrow n_{>X} - n_{<X} \le n_{=X}$$

where n' is the number of cluster members.

Let $u^{(2)}$ be in the nearest adjacent zone (X_2, Y_2) of u at its left side (i.e. $u^{(2)}$ has the maximum x coordinate among all of the members in the zones at the left side of u), $u^{(3)}$ be in (X_3, Y_3) that is the nearest adjacent zone above u (i.e. $u^{(3)}$ has the minimum y coordinate among all of the members in the zones above u), and $u^{(4)}$ be in (X_4, Y_4) that is the nearest adjacent zone below u (i.e. $u^{(4)}$ has the maximum y coordinate among all of the members in the zones below u). By comparing $f(u^{(2)})$ with $f(u)$ in the same way as above, we can achieve the first inequality in (1). By comparing $f(u^{(3)})$, $f(u^{(4)})$ with $f(u)$ 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 x coordinates greater than X is greater 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. □

We now extend Theorem 1 to an m-dimensional CAN mesh network.

Theorem 2. *Let $(U_0, U_1, \ldots, U_{(m-1)})$ represents any zone in a topologically aware m-dimensional irregular CAN mesh network that is occupied by an end host u and $n_{>U_j}$, $n_{<U_j}$ and $n_{=U_j}$ are the numbers of members with jth 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:*

$$|n_{>U_j} - n_{<U_j}| \le n_{=U_j}, \quad j = 0, 1, 2, \ldots, m-1 \tag{2}$$

The proof of Theorem 2 is omitted because it is similar to the proof of Theorem 1. We now sketch the *cluster core selection*. In the *cluster formation*, the *constructor* and *sub-constructors* receive the zone information of cluster members selected by them from the agents. The *constructor* collects the cluster 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 member is selected as the cluster core. Others become the backup cluster cores in case of leave/failure of the current cluster core. The cluster core informs the closest agent of its state information. The agent set will update the information consistently. We denote the coordinates of member u_i as $(U_{i,0}, U_{i,1}, \ldots, U_{i,(m-1)})$, $i \in [0, n'-1]$ and the coordinates of cluster core u^* as $(U_0^*, \ldots, U_j^*, \ldots, U_{(m-1)}^*)$, where $U_j^* \in [(U_j)_{\min}, \ldots, (U_j)_{\max}]$ and $(U_j)_{\min} = \min\{U_{0,j}, U_{1,j}, \ldots, U_{(n'-1),j}\}$, $(U_j)_{\max} = \max\{U_{0,j}, U_{1,j}, \ldots, U_{(n'-1),j}\}$, $j \in [0, m-1]$. The *cluster core selection algorithm* is given below.

DOI: 10.1002/cpe

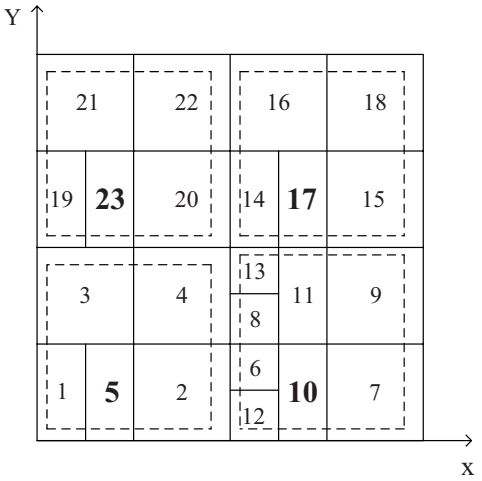


Figure 2. An example of cluster core selection.

Algorithm 1. (Cluster core selection in *m*-dimensional mesh networks)
Input: Cluster member set $CM = \{u_0 = \{U_{0,0}, \ldots, U_{0,j}, \ldots, U_{0,(m-1)}\}, \ldots, u_i = \{U_{i,0}, \ldots, U_{i,j}, \ldots, U_{i,(m-1)}\}, \ldots, u_{n'-1} = \{U_{(n'-1),0}, \ldots, U_{(n'-1),j}, \ldots, U_{(n'-1),(m-1)}\}\}$, the *constructor* v and the closest agent to v, ag;
Output: Cluster core $u^* = (U_0^*, \ldots, U_j^*, \ldots, U_{(m-1)}^*)$;

1. ag informs v of the *sub-constructors*, v sends *Req_Member(cluster id)* to each *sub-constructor*, and *Sub-constructors* respond with the zone information of cluster members selected by them through *Ack_Member(cluster id)*;
2. v initializes $\{a_{(U_j)_{\min}}, \ldots, a_{(U_j)_t}, \ldots, a_{(U_j)_{\max}}\} = \{0, \ldots, 0, \ldots, 0\}$; $//a_{(U_j)_t}$, and records the number of cluster members whose jth coordinates equal to $(U_j)_t$, where $t \in N$;
3. For ($i = 0$ to $n' - 1$) v do {
4. If ($U_{i,j} == (U_j)_t$) $\{a_{(U_j)_t} = a_{(U_j)_t} + 1;\}$} // $U_{i,j}$ is the jth coordinate of u_i
5. For ($i = 0$ to $n' - 1$) v do {
6. For ($j = 0$ to $m - 1$) v do {
7. If ($|\sum_{l=U_{j\min}}^{(U_{i,j})}[a_l - a_{(U_{i,j})}] - \sum_{l=(U_{i,j})}^{(U_j)_{\max}}[a_l - a_{(U_{i,j})}]| \le a_{(U_{i,j})}$) $\{(U_j^*) = (U_{i,j});\ j = j + 1;\}$
8. Else $\{j = m - 1;\ i = i + 1;\}\}$

We will now analyze the complexity of Algorithm 1. Steps 1–4 can be executed in time $O(n')$. Steps 5–8 are 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. However, for brevity of discussion, we keep the linear search algorithm here. According to the algorithm, the cores of the clusters in Figure 2 are 5, 10, 17 and 23.

DOI: 10.1002/cpe

Table I. An example of node weight.

	End host						
	1	2	4	5	6	7	8
Node weights	0	2	1	1	0	2	0

3.3. Weighted shortest path tree

In this section, we introduce the construct of connections (i.e. delivery paths) of cluster members in the lower layer and cluster cores in the upper layer and design a novel weighted shortest path tree construction algorithm. We first give the following definitions (using a two-dimensional mesh as the model).

- *Shortest path area nodes* (SPANs). For any two nodes (X_0, Y_0) and (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 $X_{\min}$, $X_{\max}$, $Y_{\min}$ and $Y_{\max}$ uniquely define a rectangle area that is denoted as $[X_0, Y_0] \times [X_1, Y_1]$. It can be seen that every node (X, Y) in $[X_0, Y_0] \times [X_1, Y_1]$ is on one of the shortest paths from (X_0, Y_0) to (X_1, Y_1), which is called the SPAN from (X_0, Y_0) to (X_1, Y_1).
- *SPAN nodes of a cluster member*. When the tree is built in the cluster with size n', we call all nodes in the rectangle area from the core (X^*, Y^*) to a cluster member u_i $(i \in [0, n' - 1])$ the SPAN nodes of u_i. Again, we use Figure 1 as an example. As the core is in the zone (0.75, 0.5), all end hosts located in the rectangle area $[0.75, 0.5] \times [0.25, 1.75]$ (i.e. the end hosts 2, 5 and 7) are the SPAN nodes of the end host 6.
- *Node weight*. A member may be the SPAN of several cluster members. If a member is the SPAN of w cluster members, this member is assigned the weight of w. The weight w means w cluster members may pass through the member to the cluster core by their respective shortest paths. Table I gives the weights of all members in Figure 1.
- *Path weight*. The path weight is defined as the summation of the weights of all on-path members. For example, the weight of the path $\langle 3 \to 2 \to 5 \to 6 \rangle$ is 3.

Let the cluster core be $u^* = (U_0^*, U_1^*, \ldots, U_{m-1}^*)$ and the cluster member set be

$$CM = \{u_0 = (U_{0,0}, U_{0,1}, \ldots, U_{0,(m-1)}), \ldots, u_i = (U_{i,0}, U_{i,1}, \ldots, U_{i,(m-1)}), \ldots, u_{n'-1} = (U_{n'-1,0}, U_{n'-1,1}, \ldots, U_{n'-1,m-1})\}, \quad i \in [0, n' - 1]$$

The algorithm sorts the cluster members in a non-decreasing order of numbers of overlay hops from u^* to the cluster members, thus $f(u^*, u_i) \leq f(u^*, u_j)$ if $i \leq j$. The main idea of the weighted shortest path tree construction algorithm can be outlined as follows. The *constructor* assigns a weight of 1 for each member that occupies the node in the rectangle area $[U_0^*, U_1^*, \ldots, U_{m-1}^*] \times [U_{i,0}, U_{i,1}, \ldots, U_{i,m-1}]$ as described previously. After assigning the weight of each member, the *constructor* computes the weight of each shortest overlay path. The *constructor* proposes selecting the shortest path between u^* and u_i, with the maximum weight as a tree trunk. For the message forwarding

DOI: 10.1002/cpe

along the weighted path tree, the *constructor* informs each member on the selected shortest path of the identities of its on-path neighbors during the tree construction. For example, in Table I, the end host 8 can be connected to the core 3 by the end host 7 or *vice versa*. When the path through the end host 7 is chosen, the path weight is 2. If the path through the end host 4 is chosen, the path weight is 1. According to above description, the path $\langle 3 \rightarrow 7 \rightarrow 8 \rangle$ that has the maximum weight among all shortest paths from the end host 8 to the core 3 is selected as the tree trunk. By the same token, other tree trunks are generated. The weighted shortest path tree construction algorithm is given in Algorithm 2.

Algorithm 2. (Weighted shortest path tree construction)
Input: Cluster member set $CM = \{u_0 = (U_{0,0}, U_{0,1}, \ldots, U_{0,m-1}), \ldots, u_i = (U_{i,0}, U_{i,1}, \ldots, U_{i,m-1}), \ldots, u_{n'-1} = (U_{n'-1,0}, U_{n'-1,1}, \ldots, U_{n'-1,m-1})\}$ $(i \in [0, n'-1])$, the optimal core $u^* = (U_0^*, U_1^*, \ldots, U_{m-1}^*)$;
Output: The shortest path tree T in CM;

1. $T = \{\}$;
2. for any cluster member $u_i = (U_{i,0}, U_{i,1}, \ldots, U_{i,m-1})$ with $(U_j)_{\min} \leq (U_{i,j}) \leq (U_j)_{\max}$, the *constructor* initializes its weight $W_{u_i} = 0$;
3. for $(i' = 0$ to $n' - 1)$ the *constructor* do {
   If (u_i is a SPAN node of $u_{i'} = (U_{i',0}, U_{i',1}, \ldots, U_{i',m-1})$) $\{W_{u_i} = W_{u_i} + 1; \}\}$
4. for $(i = 0$ to $n' - 1)$ the *constructor* do {Seek the path P with the maximum weight among the shortest paths between u^* and u_i, and add P into T;}

We now consider the tree construction in the upper layer. Recall that each cluster core informs its closest agent in *AS* of its existence after the core is selected. The agents then exchange such information with each other to guarantee the consistency. To construct the weighted shortest path tree, the source cluster core (i.e. the core of the cluster in which the source locates) makes a request to its closest agent for information on the cluster cores in the group. With this knowledge of cluster cores, the source cluster core uses the weighted shortest path tree construction algorithm to set up the multicast paths among the cluster cores similarly to the *constructor* of each cluster.

4. DISTRIBUTED MULTICAST ROUTING

Algorithm 3. (Distributed multicast routing for group G)

1. A source s sends its multicast message M to its cluster core c directly and its on-tree neighbors who then send M to their on-tree neighbors that have not received M;
2. On receipt of M from s, c sends M to the neighboring cluster cores along the *weighted shortest path tree* in the upper layer and also its on-tree neighbors who have not yet received the packets in the lower layer;
3. At the same time, all other cluster cores, on receiving M, forward M to their neighboring cluster cores, except the message incoming cluster core and distribute M to the neighbors on the trees in their clusters.

With the architecture design, our EMcast creates a two-layer data structure. We now introduce how to multicast packets in this architecture. For the short delay communications, the distributed multicast

DOI: 10.1002/cpe

routing algorithm directs the packets to be transmitted *in parallel* in these two layers without a global control. The data multicast paths in these two layers are constructed by different group members using the weighted shortest path tree construction algorithm. We first consider the data routing in the lower layer. Recall that the *constructor* is fully aware of the end hosts and the shortest tree architecture in the cluster. When the tree is constructed, the *constructor* is responsible for forwarding the on-tree neighbor information to each cluster member. Then, the on-tree neighbors periodically exchange the up-to-date state information to maintain the cluster trees. The multicast data are distributed to each cluster member along the trees within each cluster, except for the source cluster. In the source cluster, the sending source initiates the data routing in the cluster. It not only forwards the packets to its on-tree neighbors but also sends the packets to its cluster core directly, guaranteeing that the packets can be received by the outside group members quickly. The cluster members who have received the packets continue the packet transmission along the trees until all cluster members receive the packets.

For the packet distribution among different clusters, as introduced in the previous section, the source cluster core constructs the *weighted shortest path tree* in the upper layer. To make each cluster core aware of its on-tree neighbors, the source cluster core informs all other cluster cores of their on-tree neighbors. Then, the multicast data can be distributed along the tree from one cluster to another. To maintain the tree, each cluster core exchanges the refresh messages with its on-tree neighbors periodically to follow the up-to-date state information of the cluster cores. The distributed multicast routing algorithm is given in Algorithm 3.

5. MULTICAST PATH MAINTENANCE

We now consider the operations of restoring the dynamic group member alterations. Since the EMcast is designed in the context of an m-dimensional CAN mesh, the same schemes as the CANs dealing with members' join/leave/failure are employed to restore the normal m-dimensional CAN mesh topology.

When a new member joins in the group, the CAN splits an available zone into two equal sub-zones that are assigned to the new member and the original group member, respectively. The new member may join in the multicast architecture by the weighted shortest path tree construction algorithm. The *constructor* then recalculates the weights of nodes only covered by the two rectangle areas that are constructed by the cluster core with the new member and the original group member. After obtaining the updated node weights, the *constructor* revises the multicast tree according to the weighted shortest path tree construction algorithm.

When a group member wants to leave the group, the CAN lets the member adjacent to the alteration member merge the zone of the alteration member. Hereafter, the *constructor* employs the weighted shortest path tree algorithm to update the part of multicast tree where the nodes are covered by the two rectangle areas that are constructed by the cluster core with the alteration member and the merging member, respectively.

The failure of a group member is detected by all of the group member's neighbors because of the absence of periodical refresh message from the group member. When such a failure is detected, our multicast protocol adopts the same method to restore the normal communication as was used when the member left the group.

DOI: 10.1002/cpe

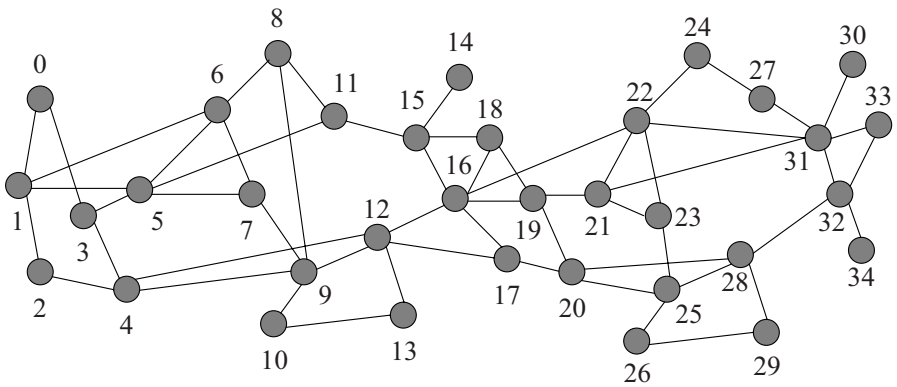


Figure 3. The experimental backbone network.

6. EXPERIMENTAL EVALUATION

In this section, we use the computer simulations to compare our EMcast protocol Two-layer Shortest Tree (TST) with two well-known EMcast protocols: NICE [16] and CAN-based multicast. In addition, the experimental observations and evaluations are presented.

6.1. Simulation model

We use *ns*-2 [17] to run the simulations on a group of SUN SOLARIS workstations. The simulation backbone network is the combination of two MCI ISP backbones as illustrated 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). In our simulation, the link bandwidth in the backbone network and the local area network is 1000 and 100 Mbps, respectively, and the propagation delays of links in the backbone network and the local area network are set as 2 and 1 ms, respectively. The simulation traffic is 1.5 Mbps MPEG-1 video streams. The parameters *SH* and *CS* are determined by the programs. We set *SH* as the minimum value of the maximum numbers of overlay hops in all local domains and *CS* as a random integer between k and $3k - 1$, where k is a constant and increases from 3 to 5 with the increment of group size.

6.2. Evaluation criteria

The following criteria are used to evaluate the performances of different EMcast protocols.

- Average multicast delay (*AMD*). We first define the multicast delay $d(s, u_i)$ from the source s to the group member u_i as the end-to-end delay between them. Then, the sum of multicast delay

DOI: 10.1002/cpe

from the source to all of the group members is

$$d(s) = \sum_{i=1, u_i \neq s}^{n} d(s, u_i),\ u_i \in G \tag{3}$$

where s is the source, n is the number of group members and G refers to the multicast group. In the experimental evaluations, we utilize the average multicast delay to evaluate the efficiency of different multicast protocols. AMD is expressed as

$$\text{AMD} = \frac{d(s)}{n-1} \tag{4}$$

- Number of links used (NLU). This refers to the total number of links that are used during the multicast communications.

6.3. Simulation results and observations

We evaluate the performances of different EMcast protocols using the metrics of AMD and NLU by two simulations. The first simulation is to observe the performances of the CAN-based multicast, NICE and TST when there is one source selected randomly by the programs and the number of group members varies from 70 to 1015.

Figure 4 illustrates the NLU results of this simulation. From this figure, we can see that the CAN-based multicast occupies a large number of physical links in order to complete the multicast communication. TST employs the weighted shortest path tree algorithm that focuses the packet transmission on the common shortest paths and therefore consumes the minimum NLU of the three protocols.

Figure 5 shows the AMD curves of the CAN-based multicast, NICE and TST. We can see in this diagram that the AMD of TST is less than that of CAN-based multicast. This trend becomes more obvious when the group size is increased. This is mainly the result of the CAN-based multicast protocols adopting the flooding scheme to route packets within the group, which requires a long latency to send the packets to the end hosts who are far away from the source. TST constructs a two-layer architecture that enables the packets to be transmitted in the two layers simultaneously, and the delays from the source to the remote end hosts are greatly decreased. The curves also show that NICE achieves the shortest AMDs. In NICE, packets are forwarded to all cluster members directly by the cluster leaders (i.e. cores). Thus, compared with TST, the packet forwarding times are decreased greatly. Furthermore, the cluster leaders in NICE are the graph-theoretic centers of their clusters who have the minimum number of maximum delays to distribute the packets within the clusters.

In the second simulation, the performances of different protocols are compared when there are 490 group members and the number of sources increases from 5 to 50. Figure 6 plots the AMD curves as multiple sources coexist. The figure illustrates that the flooding scheme in CAN-based multicast protocols incurs much larger AMD than in the other two protocols. Furthermore, in this case of multiple sources, TST can achieve better AMD performance than NICE, especially when a large number of sources coexist. This is because that the network traffic load becomes heavy when the number of coexisting sources increases. The cluster leaders in NICE need to forward multiple flows to all cluster members simultaneously, which makes them suffer from bottleneck. The tree delivery paths in TST

DOI: 10.1002/cpe

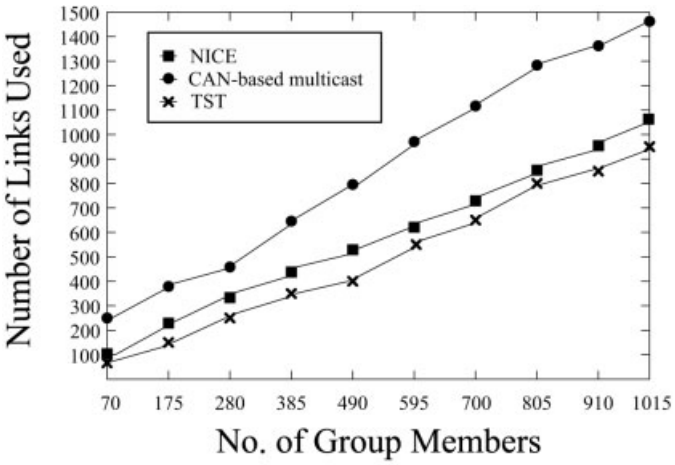


Figure 4. Simulation results for G comparison of the *total* NLU.

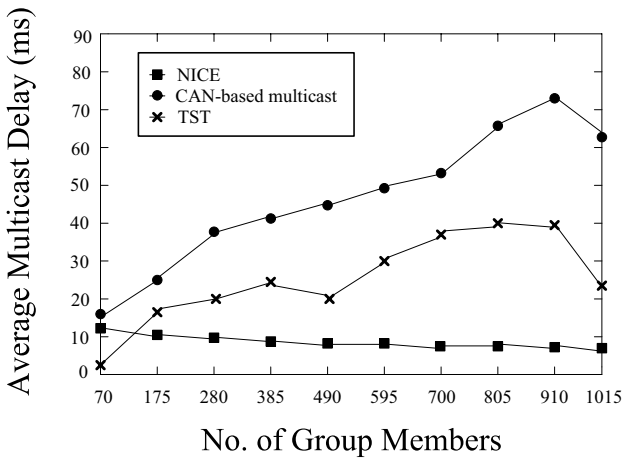


Figure 5. Simulation results for comparison of the AMD with only one sending source.

DOI: 10.1002/cpe

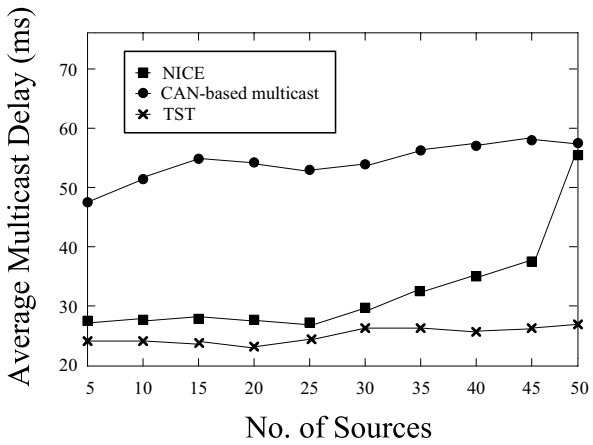


Figure 6. The comparison of the AMD with multiple sources.

Table II. The comparison of link properties in the three protocols.

	Protocol		
	CAN-based multicast	NICE	TST
Total number of links used	1175	1130	973

release the burden of cluster cores, avoiding the potential bottleneck and contributing to the shorter multicast delays when multiple sources coexist.

Table II gives a comparison of the NLU by the three protocols in this simulation. Data in the table show that TST occupies less physical links than NICE and CAN-based multicast protocols.

7. CONCLUSION

In this paper, we have fully utilized the properties of topologically aware mesh networks to design the cluster formation, cluster core selection, weighted shortest path tree construction and multicast routing in order to build up a scalable and efficient EMcast data structure. With the agent set, all of these operations are distributed. The proposed EMcast adopts a two-layer architecture to implement multicast functionality. In the lower layer, group members are assigned into different clusters by cluster formation, which decreases the group maintenance overheads. The upper layer is composed of the cluster cores of the lower layer. Cluster core selection generates a core for each cluster that may distribute the packets to all cluster members with the minimum sum of overlay hops. The connections

DOI: 10.1002/cpe

among cluster members in the lower layer and cluster cores in the upper layer are via the weighted shortest path trees.

We then used computer simulations to evaluate our EMcast protocol. The simulation results match our theoretic analysis. We believe that it is possible to achieve the distributed, short delay and low resource consumption communications on top of the m-dimensional irregular mesh network embedded in the Knowledge Grid. Our future work will extend our EMcast protocol in the Knowledge Grid, such as integrating EMcast into the sematic overlay for the key string distribution and content collection.

ACKNOWLEDGEMENTS

The authors are grateful to Dr Weijia Jia as part of the work described in this paper was done when the first author pursed her PhD degree as a member of Dr Jia's research group at the Department of Computer Science, City University of Hong Kong under the support of the National Grand Fundamental Research 973 Program of China with Grant no. 2003CB317000.

This work is also supported by the Embark Postdoctoral Fellowship Grants of Ireland Government with the Funding Source No. 501-et-504 4890 and partially supported by the Research Grants Council of Hong Kong SAR, China (Project No. HKUST6177/04E).

REFERENCES

1. Zhuge H, Sun X, Liu J, Yao E, Chen X. A scalable P2P platform for the Knowledge Grid. *IEEE Transactions on Knowledge and Data Engineering* 2005; **17**(12):1721–1736.
2. Zhuge H. *The Knowledge Grid*. World Scientific: Singapore, 2004.
3. Zhuge H. The Knowledge Grid and its methodology. *Proceedings of the 1st International Conference on Semantics, Knowledge and Grid (SKG05)*, Beijing, People's Republic of China, 2005. IEEE Computer Society Press: Los Alamitos, CA, 2001.
4. Ratnasamy S, Francis P, Handley M, Karp RD. A scalable content-addressable network. *Proceedings SIGCOMM*, San Diego, CA, 2001. ACM Press: New York, 2001.
5. Lin X, McKinley PK, Ni LM. Deadlock-free multicast wormhole routing in 2-D mesh multicomputers. *IEEE Transactions on Parallel and Distributed Systems* 1994; **5**:793–804.
6. Lin X, McKinley PK, Esfahanian AH. Adaptive multicast wormhole routing in 2-D mesh multicomputers. *Proceedings of the 5th Parallel Architecture and Languages Europe (PARLE93)*, Munich, 1993. Springer: Berlin, 1993.
7. Chu H, Rao SG, Zhang H. A case for end system multicast. *Proceedings of the 2000 ACM SIGMETRICS Conference*, Santa Clara, CA, 2000. ACM Press: New York, 2000.
8. Ratnasamy S, Handley M, Karp R, Shenker S. Application-level multicast using content-addressable networks. *Proceedings of the 2001 Special Interest Group on Data Communication (ACM SIGCOMM 2001)*, UC San Diego, CA, 2001. ACM Press: New York, 2001.
9. Stoica I, Morris R, Karger D, Kaashoek M, Balakrishnan H. Chord: A scalable peer-to-peer lookup service for internet applications. *Proceedings of the 2001 Special Interest Group on Data Communication (ACM SIGCOMM 2001)*, UC San Diego, CA, 2001. ACM Press: New York, 2001.
10. Rowstron A, Druschel P. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems, 2001. http://research.microsoft.com/∼antr/PAST/ [25 October 2006].
11. Zhao BY, Kubiatowicz J, Joseph A. Tapestry: An infrastructure for fault-tolerant wide-area location and routing, 2001. http://www.cs.ucsb.edu/∼ravenben/publications/pdf/tapestry_jsac.pdf [25 October 2006].
12. Ratnasamy S, Handley M, Karp R, Shenker S. Topologically-aware overlay construction and server selection. *Proceedings of the IEEE INFOCOM 2002*, New York, 2002. IEEE Computer Society Press: Los Alamitos, CA, 2002.
13. Xu Z, Tang C, Zhang Z. Building topology-aware overlays using global soft-state. *Proceedings of the 23rd International Conference on Distributed Computing Systems (ICDCS 2003)*, Providence, RI, 2003. IEEE Computer Society Press: Los Alamitos, CA, 2003.
14. Tu W, Jia W. A scalable and efficient end host multicast for peer-to-peer systems. *Proceedings of the IEEE Global Telecommunications Conference*, Dallas, TX, 2004. IEEE Computer Society Press: Los Alamitos, CA, 2004.

DOI: 10.1002/cpe

15. Baker F. Distance vector multicast routing protocol—DVMRP. *Request for Comments 1812*, IEEE Communication Society, 1995.
16. Banerjee S, Bhattacharjee B, Kommareddy C. Scalable application layer multicast. *Proceedings of the 2002 ACM Special Interest Group on Data Communication (ACM SIGCOMM 2002)*, Pittsburgh, PA, 2002. ACM Press: New York, 2002.
17. UC Berkeley, LBL, USC/ISI, and Xerox PARC. Ns notes and documentation. http://nsnam.isi.edu/nsnam/index.php/User_Information [20 October 2006].

DOI: 10.1002/cpe