

End Host Multicast for Peer-to-Peer Systems*

Wanqing Tu and Weijia Jia

Department of Computer Engineering and Information Technology,
City University of Hongkong, Kowloon, Hongkong, P.R. China
tu.wanqing@student.cityu.edu.hk, itjia@cityu.edu.hk

Abstract. Multicast is an effective means for conducting the cooperative P2P communications. This paper studies an algorithm to construct a scalable and efficient end host multicast tree – *Shared Cluster Tree* (SCT). Compared with the previous work, by utilizing the underlying network properties, the novelty and contributions of our multicast algorithms are: (1) Scalability: by fully utilizing the underlying network properties to layer and cluster the group members, SCT decreases the transfer of the data traffic in the *backbone domain* links and therefore reserves more *backbone domain* resources to more *local domains* which include many more end hosts; (2) Efficiency: by searching the cluster cores based on the *cluster core selection* method, packets are multicasted to the cluster members by the cluster cores in the minimum delay. Our simulation results indicate that SCT is scalable and efficient for the end host multicast in P2P systems. . .

1 Introduction

Peer-to-peer (P2P) computing is the system of mutually exchanging information and services directly between the sender and the receiver. Grids are systems that integrate of instruments, displays, computational and information resources, managed by diverse organizations in potentially geographically widespread locations. In P2P systems, any two peers are of comparable capabilities and permitted to communicate with each other in such a way either ought to be able to initiate the communications. Thus, P2P computing is a powerful tool to organize grid and cooperative computing. A number of grid and cooperative applications (e.g. audio-video conferences, network games and distance learning) require multicast transmission to enable efficient many-to-many communications. Because multicast is an efficient means for the distribution of information that should be transmitted to a group of members. As end hosts in most P2P systems (e.g. Content-Addressable Network (CAN)) construct an overlay network, such overlay structure is in the sense that each of its edges corresponds to a unicast path between two end hosts in the underlying Internet. In this paper, our motivation is to develop a scalable and efficient multicast in the overlay network for P2P networking. Multicast in the overlay network is often called as the end

* The work is supported by Research grant council (RGC) Hong Kong, SAR China under CityU strategic grant nos. 7001587.

host multicast by many researchers. Compared with unicast, although the end host multicast communication has more scalability and efficiency, the following analysis shows that the end host multicast is less scalable and efficient than the network layer multicast.

One of the dominant differences between the end host multicast and the network layer multicast is that the multicast packets are replicated and forwarded by the end hosts instead of the intermediate routers. End host replicating and forwarding incurs the identical packets traversing the same underlying links more than one times. Excessive identical packets occupy the network resources. It influences the scalability of the end host multicast.

As for efficiency, our studies show that the end hosts instead of intermediate routers forwarding can cause not only worse scalability but also longer transmission delay. This is mainly because of the excessive delay that packets transmit from the application layer to the network layer as shown in Figure 1. We briefly analyze the delay difference as follows: Let the delay from the application layer to the network layer be $d_{a \rightarrow n}$, the delay from the network layer to the physical layer be $d_{n \rightarrow p}$, and the delay that packets transmit in the physical links be $d_{p \rightarrow p}$. The total end-to-end delay in (a) is $d_{nt} = 2d_{a \rightarrow n} + 4d_{n \rightarrow p} + 2d_{p \rightarrow p}$, however, the total end-to-end delay in (b) is $d_{et} = 4d_{a \rightarrow n} + 4d_{n \rightarrow p} + 2d_{p \rightarrow p}$. The more the number of end hosts that forward the packets, the longer end-to-end transmission delay is required for the multicast. Moreover, the end hosts have lower capacities to deal with and forward packets than the intermediate routers, which incurs longer processing delay. Hence, the end host multicast is costly in terms of delay. Although the end host multicast cannot overtake the network layer multicast in terms of the performances, the end host multicast's independence of the underlying network architecture makes it feasible and popular. It is thus worthwhile to improve the scalability and efficiency of end host multicast. Since the underlying network is the fundamental of the overlay network, we fully utilize the underlying network characteristics to develop the end host multicast, which is called as *Shared Cluster Tree* (SCT). "*Shared tree*" refers to the multicast data path, and a "*cluster*" refers to a "mini-group". Unlike the

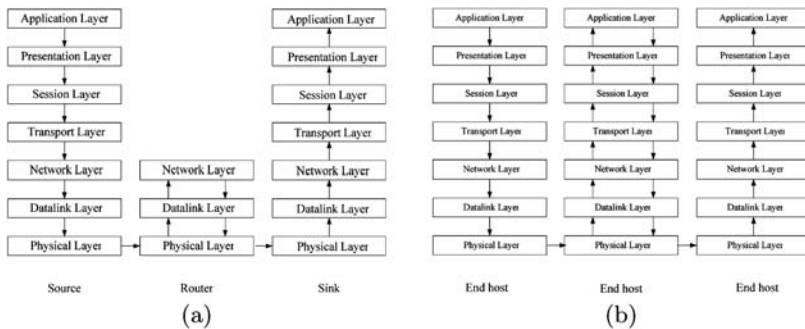


Fig. 1. Delay difference between the router forwarding architecture (a) and the end host forwarding architecture (b)

previous work, the novelty and contributions of the paper are: 1) *Scalability*: by fully utilizing the underlying network properties to layer and cluster the group members, our *cluster construction*, *layer construction* and *shared tree creation* regulations decrease the transfer of the data traffic in the *backbone domain* links and therefore reserve more *backbone domain* resources to more *local domains* which contain many more end hosts. (The *local domain* and the *backbone domain* will be defined in Section III); 2) *Efficiency*: by applying the *cluster core selection*, each cluster selects a core. The core guarantees to multicast the data packets to the cluster members with the minimum delay; by utilizing the hierarchy and cluster, SCT limits the flow transmission in the links among routers, which also improves the efficiency.

The rest of the paper is organized as follows: Previous related work is discussed in Section 2. Section 3 provides the detailed SCT design. Section 4 evaluates the well-known end host multicast protocols: CAN-multicast, NICE, and our DSCT, and gives the simulation observations and performance comparisons. Section 5 concludes the paper.

2 Related Work

In terms of the sequence of constructing the data and control topologies, the current end host multicast protocols are classified into three categories: 1) *Mesh-first multicast*. NARADA [8] is a *mesh-first* end host multicast protocol. It demonstrates the feasibility of implementing the multicast functionality at the application layer. It distributedly organizes group members into an overlay mesh topology, and runs a protocol similar to the conventional DVRMP to accomplish multicast routing among these members. To guarantee robust, each group member in NARADA periodically sends refresh packets to each of other group member, which compromises the scalability of NARADA; 2) *Tree-first multicast*. YOID [9] is one of the earliest end host multicast protocols. It constructs a shared data delivery tree among members first. The use of tree routing is to get logarithmic scaling behavior with respect to the number of receivers. But YOID would not be efficient and simple, because it has a direct control over various aspects of the tree structure (e.g. the out-degree at the members). Moreover, expensive techniques of loop detection and avoidance, and resilience to network partitions are required in YOID; 3) *Implicit multicast*. Well-known NICE [10] and CAN-based multicast [11] all adopt the implicit approach. CAN-based multicast employs the flooding method to multicast the packets to all of the neighbors who haven't received the corresponding packets before. The flooding scheme is simple, but resource-consumed and inefficient. NICE is a hierarchical multicast, each layer is composed of a set of clusters. Each cluster includes $k \sim (3k - 1)$ members, where k is a constant. In NICE, k is normally set as 3. Packets are multicast to each cluster member by the cluster leader, except in the source cluster where packets are sent to each cluster member by the source. Compared with other protocols, NICE is more scalable for the hierarchy and cluster. But the hierarchy and cluster is independent of the underlying network. Actually, by

utilizing the underlying network properties, as described in this paper, we can improve the scalability further.

3 SCT Design

We first make the following definitions. 1) *Local domain*: It is composed of group members that attach to the same router directly or through several local network components (e.g. the hubs) and other local network resources (e.g. physical links, the hubs) used to connect these members. In this definition, the router mainly refers to the local router, not the router in the backbone network. 2) *Backbone domain*: It is composed of all the routers used to connect the multicast group members and other network resources (e.g. physical links) among these routers. The routers in this definition include the local routers and the backbone routers. 3) *Unassigned end hosts*: Define the end hosts in a multicast group that haven't been assigned into any cluster as the *unassigned end hosts*.

Apart from employing the hierarchy scheme, our basic idea to design SCT is to limit the transfer of packets in the *backbone domain* links, which reserves the *backbone domain* resources to carry more end hosts and decreases the chance that packets transmit in the long delay *backbone domain* links.

3.1 Cluster Construction

In SCT tree, each *local domain* has a *local core*. The selection of *local core* is introduced in the *layer construction* regulation. Members in the same *local domain* form the “intra-clusters”. *Local cores* in different *local domains* form the “inter-clusters”. Different cluster sizes exist for these two kinds of clusters.

As for the “intra-cluster”, the size is expressed as:

$$s_{ina} = \begin{cases} (k, 3k - 1) & , \quad r_n > 3k - 1 \\ r_n & , \quad r_n \leq 3k - 1 \end{cases} \quad (1)$$

where r_n is the number of *unassigned end hosts* in the same *local domain*, and the expression $(k, 3k - 1)$ represents a random constant between k and $3k - 1$. Like NICE, k is a constant, and in our experiment, we also use $k = 3$. The cluster size bound, $(k, 3k - 1)$, is the same as the one in NICE, which avoids the frequent cluster splitting and merging (see [10]). The way to cluster *local domain* members is to partition s_{ina} *unassigned end hosts* who are the closest ones to one another among all *local domain* members into the same cluster. Expression 1 guarantees that each “intra-cluster” only contains members of the same *local domain*. Hence, the packets' traversing over the *backbone domain* links is limited.

Similarly, as the “inter-cluster” contains end hosts of different *local domains*, the size is denoted as:

$$s_{ine} = (k, 3k - 1) \quad (2)$$

3.2 Cluster Core Selection

In SCT, for the efficient communications, the core of each cluster is selected *in terms of the minimum sum of unicast distance from the core to all the members.*

Let s be the cluster size, $d(i, j)$ be the unicast distance from member i to member j , $i, j \in [1, s]$, d_i be the sum of $d(i, j)$, d_i can be expressed as:

$$d_i = \sum_{j=1, j \neq i}^s d(i, j) \tag{3}$$

Denote the cluster core as c . Then, the cluster core satisfies:

$$d_c = \min\{d_i, i \in [1, s]\} \tag{4}$$

3.3 Layer Construction

Local domain members are mapped to different layers using the following scheme: All members in the same *local domain* d , $d \in [0, n_{ld} - 1]$, belong to the lowest layer L_0 . The *Cluster construction* regulation assigns them into different clusters in L_0 . By using the *cluster core selection*, a cluster core is selected for each cluster in layer L_i . Cluster cores in L_i join in layer L_{i+1} . Also, members in L_{i+1} are partitioned into different clusters. Then, the cluster cores of clusters in L_{i+1} become the members of layer L_{i+2} . The hierarchy in the *local domain* d terminates until the number of members is not greater than $3k - 1$, and the core of these members is the *local core*, which constructs the uppest layer $L_{LHL(d)}$ of *local domain* d . $LHL(d)$ is the uppest layer number of *local domain* d . For the layer is labelled from 0, the number of layers in *local domain* d is $LHL(d) + 1$. In this layer method, each member in layer L_i is actually the cluster core of cluster in layer L_j , $j \in [0, i - 1]$ that the member belongs to. Formula 5 illuminates the relationship among the number of members n_d , the uppest layer number $LHL(d)$, and the size of the cluster s_{ina} in *local domain* d .

$$\frac{n_d}{(s_{ina})^{(LHL(d)-1)}} \leq 3k - 1 \tag{5}$$

Local cores are mapped to different layers using the similar scheme: As different *local domains* contain different numbers of end hosts, the local uppest layers $L_{LHL(d)}$ may be different from one another. Denote the maximum value of the local uppest layer numbers LHL as:

$$LHL = \max\{LHL(d), d \in [0, n_{ld} - 1]\} \tag{6}$$

All *local cores* belong to layer L_{LHL} . *Local cores* go on to be layered into several layers that are not lower than L_{LHL} , then partitioned into several clusters with the size of $(k, 3k - 1)$. The hierarchy terminates until the number of *local cores* is not greater than $3k - 1$, and the core of these members is the *SCT core*, which constructs the highest layer L_{HL} of SCT. HL is the highest layer number of SCT tree. The total layer number of SCT is $(HL + 1)$. Formula 7 illuminates the

relationship among the number of *local cores* n_{lc} , the number of layers ($HL - LHL + 1$) containing the “inter-clusters”, and the size of the clusters s_{ine} .

$$\frac{n_{lc}}{(s_{ine})^{(HL-LHL)}} \leq 3k - 1 \quad (7)$$

Formula (5) and (7) show that the cluster size is the inverse proportion to the total layer number $HL + 1$ with respect to the fixed number of group members $n = \sum_{d=0}^{n_{ld}-1} n_d$.

3.4 Shared Tree Creation

The main idea of the tree generation algorithm can be sketched as follows: SCT tree is a hierarchical shared tree. The *layer construction* assigns the group members into different layers. The *cluster construction* partitions the members of the same layer into different clusters. Each cluster is a minor shared tree on the SCT tree. The cluster core of either the “intra-cluster” or the “inter-cluster” coordinate the communications between the members inside and outside the cluster. Namely, packets from the outside of the cluster are forwarded to each of the cluster member by the cluster core. And, packets sent by a cluster member forward to the cluster core first, then, the cluster core multicasts the packets to: a) other members in the source cluster; b) group members in other clusters in which it is also the cluster core; c) its cluster core of the cluster in the upper layer that it belongs to. Similarly, these receivers go on to forward the packets along the shared tree until all group members receive the packets. The setting of the *local core* (i.e. the cluster core of “inter-cluster”) decreases the packet transmission in the *backbone domain* links greatly. One packet transmitted from *local domain a* to *local domain b* only utilizes the same *backbone domain* links between these two *local domains* one time. The following is the *SCT hierarchical shared tree creation algorithm*.

Algorithm 1 SCT Hierarchical Shared Tree Creation Algorithm

Input: Group $G = \{g_1, g_2, \dots, g_n\}$;

Output: SCT hierarchical shared tree T_{SCT} ;

1. $T_{SCT} = \{\}$;
 2. For $j = 0$ to $n_{ld} - 1$ do
 3. Apply the *layer construction* and the *cluster construction* regulations to layer and cluster group members of the *local domain j*, select the cluster cores according to the *cluster core selection* method, and add the operation results to T_{SCT} ;
 4. Select the member in $L_{LHL(j)}$ as the *local core* of the *local domain j*, add it to T_{SCT} ;
 5. Apply the *layer construction* and the *cluster construction* regulations to layer and cluster *local cores* from layer L_{LHL} , select the “inter-cluster” cores by using the *cluster core selection* method, and add the operation results to T_{SCT} ;
 6. Select the member in the highest layer L_{HL} as the *SCT core* of T_{SCT} , and add it to T_{SCT} .
-

4 Experimental Evaluation

We use *ns-2* [12] to run our simulations on a group of SUN SOLARIS workstations. The simulation backbone network adopts the MCI ISP backbone as shown in Fig. 2. End peers directly or indirectly attach to the routers in the backbone. The link bandwidth in the backbone network and the *local domain* are 1000Mbps and 100Mbps respectively. Random integers between 20 and 40 and between 4 and 8 are set as the link costs of *backbone network* and *local domain* respectively. Three criteria are used to evaluate the performances of different end host multicast protocols. 1) Average end-to-end delay (AD): AD of source *s* is define as the ratio of the sum of end-to-end delay from *s* to each group member to the number of group members; 2) Average cost stress (ACS): Define ACS as the ratio of the sum of the consuming costs of identical packet copies in each link to the sum of each link’s inherent cost; 3) Average link stress (ALS): ALS is defined as the ratio of the sum of numbers that identical packet copies traverse over the same underlying links to the number of links in the group.

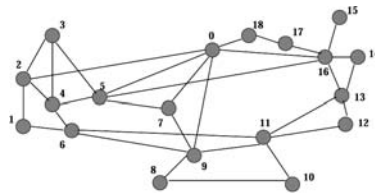


Fig. 2. The Experimental Backbone Network

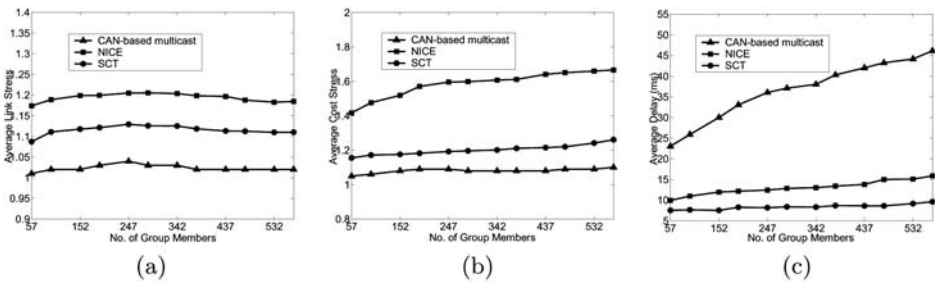


Fig. 3. The performance results. (a) is for the average link stress, (b) is for the average cost stress, (c) is for the average end-to-end delay

In this simulation, there are 5 sources, we observe the performance of proposed multicast protocols as the group members increase from 57 to 570. Fig. 3 (a) illuminates that SCT achieves lower link stress than NICE. In SCT and NICE, the worst link stress appears in the links around some cluster core (or some cluster leader in NICE). The *cluster construction* in SCT disperses the worst link stress by using different cluster sizes. For CAN-based multicast, the

flooding distributes the link stress much more evenly over all links, hence, the worst link stress of CAN-based multicast is much lower than SCT and Nice. It introduces lower ALS of CAN-based multicast. However, the flooding scheme occupies all of links in the multicast group, which makes it resource-consumed. Fig. 3 (b) illuminates that SCT achieves much lower cost stress than NICE. As we have analyzed, SCT avoids the packets' unnecessary visit to the costly links. The curves prove that the construction of SCT tree meets the demand of limiting the data traffic in the *backbone domain* links. Hence, SCT is more scalable and efficient than NICE. For the same reason of ALS, the ACS of CAN-based multicast is lower than SCT and NICE. The *AD* curves shown in Fig. 3 (c) illuminate that SCT is the most efficient one in these three multicast mechanisms in terms of *average end-to-end delay*. NICE achieves relatively much lower *AD* than CAN-based multicast. The flooding scheme in CAN-based multicast incurs much longer delay. The simulation results show that the core selection and the hierarchy and cluster in SCT are effective for improving efficiency and scalability of end host multicast in P2P systems.

5 Conclusion

Unlike previous related studies, in this paper, we utilize the underlying network properties to cluster and layer the group members, to select the cores and to construct the multicast tree. Our main contribution is to improve the scalability and efficiency of end host multicast by our novel data structure – SCT (Shared Cluster Tree). Theoretic analysis and performance data have demonstrated that our SCT is indeed scalable and efficient for multicasting in the overlay networks of P2P systems. It is our target to implement SCT multicast in practical P2P cooperative systems.

References

1. C.Lv, P.Cao, ECohen, K.Li, and S.Shenker, "Search and replication in unstructured peer-to-peer networks", Preprint, Optinal 2001.
2. E.Cohen and S.Shenker, "Optimal replication in random search networks", Preprint, Optinal 2001.
3. Q.Lv, S.Ratnasamy, and S.Shenker, "Can heterogeneity make gnutella scalable?", *Proc. of the 1st Internertional Workshop on Peer-to-Peer Systems (IPTPS '02)*, MIT Faculty Club, Cambridge, MA, USA, March 2002.
4. I.Stoica, R. Morris, D.Karger, M.Kaashoek and H.Balakrishnan, "Chord: A scalable content-addressable network," In *Proc. Of ACM SIGCOMM'01*, Aug. 2001.
5. Sylvia Ratnasamy, Mark Handley, Richard Karp and Scott Shenker, "Application-level multicast using content-addressable networks", *Proc. Of ACM SIGCOMM'01*, Volume 8 Issue 4. Aug. 2001.
6. Rowstron, A. and Druschel, P., "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems", Available at <http://research.mecrosoft.com/antr/PAST/>, 2001.

7. Zhao, B.Y., Kubiattowicz, J. and Joseph, A, "Tapestry: An infrastructure for fault-tolerant wide-area location and routing", Available at <http://www.cs.berkeley.edu/ravenben/tapestry/>, 2001.
8. Y.H.Chu, S.G.Rao and H.Zhang, "A case for end system multicast", In *Proc. of ACM SIGMETRICS*, June 2000.
9. P.Francis, "Yoid: Extending the multicast internet architecture", White paper <http://www.aciri.org/yoid/>, 1999.
10. S.Banerjee, B.Bhattacharjee and C.Kommareddy, "Scalable application layer multicast," *Proc. of ACM SIGCOMM*, August 2002.
11. Sylvia Ratnasamy, Mark Handley, Richard Karp and Scott Shenker, "Application-level multicast using content-addressable networks", *Proc. of NGC*, 2001.
12. UC Berkeley, LBL, USC/ISI, and Xerox PARC, "*ns* Notes and Documentation," October 20, 1999.