DINCast: A Hop Efficient Dynamic Multicast Infrastructure for P2P Computing

H. Q. Guo*, L. H. Ngoh+, W. C. Wong+* and J. G. Tan+

*Department of Electrical & Computer Engineering, The National University of Singapore
2 Engineering Drive 4, Singapore 117584

+Institute for Infocomm Research, A*STAR, 21 Heng Mui Keng Terrace, Singapore 119613
e-mail: {guohq, lhn, lwong, tjg}@i2r.a-star.edu.sg

Abstract—Efficient communications among computers participating in Peer-to-Peer (P2P) computing remains a problem when they are geographically distributed across multiple network domains. In this paper, we propose DINCast to support group communications required in P2P computing. DINCast is built on top of an existing P2P multicast tree and results in a hop efficient dynamic multicast infrastructure. We show that DINCast is able to reduce communications delay compared with the original tree-based multicast. DINCast is also advantageous since it can achieve load balance and data persistency.

Keywords— DINCast, tree-based multicast, DIN-Loop, hop count, peer-to-peer computing

I. INTRODUCTION

The prospect of doing computing in a Peer-to-Peer (P2P) environment, where there is potential of harnessing millions of computers to collaborate in large-scale computation, is very exciting. P2P computing is the sharing of computer resources and services by direct exchange between client systems. These resources and services often include the exchange of information, processing cycles, and disk storage for files. P2P computing achieves processing scalability by aggregating the resources of a large number of individual computers. With P2P computing applications, millions of users have contributed their computing resources to work collaboratively on common computational-intensive problems. P2P computing has offered a compelling and intuitive way for Internet users to communicate and collaborate with their peers in real time. These emerging collaborative P2P computing applications require efficient group communication. However, efficient communications among the many computers participating in the computation remains a problem when the computers are geographically distributed over a large area across multiple network domains. The fundamental challenge of communication in a P2P community is overcoming the problems associated with the dynamic nature of peers. One way in which group communications can be supported is by using the multicast concept. Using multicast, participating computers can be grouped together logically as a multicast group. Group communications are thus simplified as there is no need for individual computers to do m-to-m communications to all the other computers in the group. Multicast can also enforce the anonymity of peers.

The availability of a multicast infrastructure to support large-scale P2P computing is therefore very important. One of the ways in which this can be provided is through the use of network-level IP multicast. Network-level IP multicast was proposed over a decade ago [9, 10]. With IP Multicast, a single packet transmitted at the source is delivered to an arbitrary number of receivers by replicating the packet within the network routers along a multicast tree rooted at the traffic’s source or a router assigned as the rendezvous point. However, the use of IP multicast has been limited because of issues related to scalable interdomain routing protocols, tracking group membership and so on. Today many Internet Service providers (ISPs) are still reluctant to provide a wide-area multicast routing service [11].

Because of problems facing the deployment of a network-level multicast service, application-level multicast has gained in popularity. Algorithms and systems for scalable, reliable propagation of messages are still active research areas [2, 3, 8, 16, and 32]. For such systems, the challenge remains to build an infrastructure that can achieve low delay and effective use of network resources.
Recent work on P2P overlay network offers scalability and robustness for advertisement and discovery of services. Pastry [25], Chord [27], CAN [22] and Tapestry [31] represent typical P2P routing and location schemes. Furthermore, there has been a number of works reported on overlay multicast, e.g., Scribe [5], CAN-Multicast [23], Bayeux [32], YOID [17], TBCP [20], HMTP [30], NICE [1], Overcast [19] and ZIGZAG [28]. Each uses a different overlay and implements application-level multicast using either flooding (CAN-Multicast) or tree-building (Scribe, Bayeux, YOID, TBCP, HMTP, NICE, Overcast and ZIGZAG). The flooding approach creates a separate overlay network per multicast group and broadcast messages within the overlay. The tree approach uses a single overlay and builds a tree topology first [6, 15]. In [6], the results showed that the tree-based approach of Scribe consistently outperforms the flooding approach of CAN-Multicast.

Compared to native network-level IP multicast, application-level multicast has a number of advantages. First, a major advantage is that most proposals do not require any special support from network routers and can therefore be deployed universally. Second, the deployment of application-level multicast is easier than IP multicast by avoiding issues related to interdomain multicast.

However, application-level multicast also suffers two disadvantages. First, due to the fact that application-level multicast is being implemented at host-level and the underlying physical topology is hidden, even with topology-awareness [6], using application-level multicast still increases the delay to deliver messages relative to IP multicast. Second, in P2P tree-based multicast, the multicast trees are built at the application level and the rendezvous point is the root of the multicast tree. Subsequently, every member that joins the multicast group becomes the nodes of the multicast tree. A sender sends data to the rendezvous point. The rendezvous point forwards the data along the multicast tree to all members. The rendezvous point can potentially be subjected to overloading and single-point of failure. For example, in Real-Time Conferencing Protocol [24], to provide a multicast back-channel for a group of receivers will result in serious source implosion problem [15]. In this paper, we propose DINCast to overcome the above disadvantages.

We apply the loop concept to application-level tree-based multicast and propose DINCast. The loop idea has been adopted in several examples. The fiber loop memory [26] was proposed to resolve packet contention problems, in which the contending packet circulates in the fiber loop until the optical network resources (e.g., receiver, switches, or wavelength) become available. Wavelength Disk Drives (WDD) used a multi-wavelength optical network as a disk drive [4]. WDD attempts to store data in DWDM networks across Canada to facilitate a new content-based messaging distributed computing paradigm.

In DINCast, data circulates in a logical loop (called DIN-Loop) consisting of two or more nodes (called DIN-Nodes). In this way, data is readily made available to multiple hosts and the DIN-Loop allows them to share and access data efficiently. The concept of DIN-Loop has been applied in the areas of networked video storage and retrieval [18], distributed application data cache [12], and divisible-load computing model [13, 14]. In DINCast, DIN-Loop is introduced to application-level tree-based multicast resulting in a topologically-aware dynamic multicast infrastructure. DINCast can be built on top of the existing multicast tree and it complements the tree-based multicast by improving its performance. The remainder of this paper is organized as follows: Section II describes the overview of DINCast and elaborates on the various advantages over existing application-level multicast. Section III presents the details of DINCast and steps required to establish DINCast over an existing application-level multicast tree. To further investigate the proposed solution, we present simulation results comparing the hop-count efficiency in Section IV. Finally we conclude research results and discuss potential future work in Section V.

II. OVERVIEW OF DINCAST

A. Overview

DINCast consists of two major components: DIN-Loop and DIN Node. A DIN-Loop refers to a collection of two or more DIN Nodes connected by the underlying network in a loop.

DINCast is built based on an existing application-level multicast tree. The novel part is that multiple DIN Nodes form a DIN-Loop and DINCast uses this loop instead of the rendezvous point as multicast sources. In DINCast, any multicast message is sent to the loop
instead of one single rendezvous point. The messages are forwarded to all members. At the same time, the multicast message circulates in the DIN-Loop.

In the P2P tree-based multicast, the multicast trees are built at the application level and the rendezvous point is the root of the multicast tree. Every member that joins the multicast group becomes a node of the multicast tree. All data is sent to the rendezvous point. The rendezvous point forwards the data along the multicast tree to all members. Fig. 1 is a tree-based multicast and Node 0 is a root (rendezvous point). The dotted arrow lines are the paths for the multicast message.

Figure 1. Tree-based multicast.

An example of the proposed DINCast is shown in Fig. 2, where DIN Nodes 0, 1, 2, and 3 form a DIN-Loop $0 \rightarrow 2 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 0$ (thick line). In DINCast, any node sends the message to the DIN-Loop. The first DIN Node that receives the message will forward the message to its child-nodes (dotted arrow line, outside of the DIN-Loop), parent nodes (inside of the DIN-Loop, not indicated in the figure) and the neighbor DIN Node along the DIN-Loop. The associated child-nodes are organized in sub-tree type. The neighbor DIN Node will forward the message to its associated child-nodes, parent nodes and its neighbor DIN Node along the DIN-Loop. The process repeats itself until all DIN Nodes receive the message or when the lifetime of the message expires. In this scheme, one DIN-Loop corresponds to one rendezvous point/one multicast tree. If there are multiple multicast trees, DINCast will build multiple DIN-Loops dynamically.

Figure 2. DINCast.

B. Advantages of DINCast

DINCast therefore retains the advantages of seamless and easy deployment offered by application-level multicast. It also does not affect the scalability and robustness of tree-based multicast. DINCast further offers three more advantages that lighten or overcome the disadvantages of application-level tree-based multicast.

- Efficient Communications

In DINCast, the messages are forwarded to all members by DIN-Loop instead of from the rendezvous point which can be deep in the network and thus further away from the
members. Any message is received by one of DIN Nodes. This DIN Node will forward to
its associated child-nodes immediately and its neighbor DIN Node. So the delay for the
nodes to receive the message can potentially be reduced.

- **Load Balance**
  A DIN-Loop with multiple DIN Nodes is used to replace a single rendezvous point. All
  nodes in the loop can play the role of rendezvous point so that DINCast relieves the
  bottleneck of the rendezvous point. DINCast achieves load balance and avoids the single-
  point of failure.

- **Data Persistency**
  In DINCast, the message circulates in the DIN-Loop continuously and is therefore
  persistent in the DIN-Loop, that is, it “lives” for a certain period of time in DIN-Loop.
  DINCast can control the lifetime of the message. During the lifetime of the message,
  nodes can get the message from the DIN-Loop and the sender does not need to send the
  message again. This also reduces the delay to receive the message and reduces the traffic
  from the sender to the root.

  Data persistency is needed in various real applications. For example, the mobile
devices/components are sometimes disconnected or intermittently connected from the
network due to a number of factors: hardware mobility (e.g., out of range, device turned
off) or software mobility (e.g., agent migration, application swapped if an operating
system is single tasking). Because of data persistency, these mobile devices/components
can still get the message from DIN-Loop at a later time.

  The following subsection presents the details of forming the DIN-Loop.

### III. DETAILS OF DINCAST

Before going to the detailed steps to establish DINCast, the concept of hop depth is
introduced.

#### A. *Hop Depth*

If the rendezvous point of the tree-based Multicast is a DIN Node in DINCast, i.e., Fig. 2,
then it is called 0 hop depth. If the rendezvous point of the tree-based multicast is not the DIN
Node in DINCast and the minimum hop count from the rendezvous point to the DIN-Loop is
*n*, then it is called *n* hop depth. For example, Node C is the rendezvous point for the tree-
based multicast in Fig. 3. The thick line with arrow is the DIN-Loop. The minimum hop count
from Node C to the DIN-Loop is 1 hop. So it is said to have 1 hop depth.

![Figure 3. 1 hop depth in DINCast.](image-url)

There are two steps to establish DINCast. First, discover the DIN Nodes. Second, DIN
Nodes update the node state to form the loop based on a loop-forming algorithm. The details
are described in the following sections.
B. Discovery of DIN Nodes

Before forming DIN-Loop, DIN Nodes need be discovered first. There are some steps to discover the DIN Nodes:

1. The rendezvous point sends a discovery message to its child-nodes in the multicast tree with “TTL=n”. “TTL” means time-to-live at the application level similar to that employed in routers. When the message passes one node (1 hop), the TTL value is reduced by 1.

2. All nodes in the multicast tree that receive this message will inform the rendezvous point about their IDs and hop counts from themselves to the rendezvous point. The rendezvous point notes this information.

3. The rendezvous point informs all nodes in the n hop count distance to form a loop. (This loop is just for the purpose to get the loop information. The actual loop will not be formed until step 10.) The rendezvous point assigns series numbers to these nodes starting from 0. The rendezvous point gives all information to them so that they know each other.

4. Use the algorithm in Sub-section C, Automatic Formation of Loop, and Sub-section D, Optimization of Loop, to form the loop.

5. The loop information with node IDs comes back to the rendezvous point.

6. The rendezvous point checks the loop information and gets the minimum hop count (the hop depth) among the nodes in the loop and the number of the nodes in the loop. So the rendezvous point records the node number, the hop depth and node ID for each node as shown in Table I.

7. n=n-1, repeat step 3 till n=0.

8. So the rendezvous point has information of n possible loops in Table I. Based on Table V in Section IV, the rendezvous point will choose one suitable loop among n loops as a DIN-Loop with the available DIN Node number and the hop depth for the best performance in the current overlay network topology.

9. If no available solution is found, the rendezvous point chooses any one of its child-nodes to form the DIN-Loop with 2 DIN Nodes and 0 hop depth.

10. The nodes in the selected DIN-Loop are DIN Nodes. The rendezvous point informs the DIN Nodes to update their node state (refer to Sub-section E, DIN Node State) so that the actual DIN-Loop is formed.

The costs associated with building the DIN-Loop are the additional messages described above. However, this overhead is minimal since the messages are sent only during the initial setup of the loop. In a dynamic P2P network, as long as the rendezvous point is not changed, the DIN-Loop remains. If DIN Nodes quit or fail, the number of remaining DIN Nodes will be used by the rendezvous point to decide if the DIN-Loop must be updated based on Table V in Section IV.

<table>
<thead>
<tr>
<th>Loop ID</th>
<th>Number of nodes to form the loop</th>
<th>Hop Depth</th>
<th>Node ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>1</td>
<td>ID1, ID2, ID3</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
</tbody>
</table>

The following strategy is used to form the loop.

C. Automatic Formation of Loop

Assume there are k nodes to form the loop. The series number that the rendezvous point assigns are 0, 1, 2, …, k-1.

At the beginning, Node 0 and Node 1 form a loop (Fig. 4a). Then Node 2 adds to the loop (Fig. 4b). If Node 3 and above node adds to the loop, every node in the loop will traceroute to this new node and get the hop count. One node in the loop will be obtained with the minimum
hop count to this new node. One of its two adjacent nodes in the loop with lower hop count value to this new node will be obtained too. Then these two nodes will break the link and add in the new node (Fig. 4c and Fig. 4d).

Figure 4. Automatic formation of loop.

D. Optimization of Loop

Based on the steps described above, further optimization can be done. Assume Node \( a \) is the new node (Fig. 5a). The value on the arc in Fig. 5 is the hop count.

Node \( b \) has minimum hop count (1) to Node \( a \). Node \( c \), one of Node \( b \)’s adjacent nodes in the loop, is selected because it has lower hop count (4) to Node \( a \). Use the above algorithm to form a loop (Fig. 5b). The loop goes through \( e \rightarrow b \rightarrow a \rightarrow c \). Assume \( xy \) means the hop count from Node \( x \) to Node \( y \). However, after summing the hop count \((eb+ba+ac)\) and \((ea+ab+bc)\), we find that \((eb+ba+ac)\) is larger than \((ea+ab+bc)\).

So we further optimize the DIN-Loop. If \((eb+ba+ac) > (ea+ab+bc)\), then the optimal loop goes through \( e \rightarrow a \rightarrow b \rightarrow c \) (Fig. 5c). In this way, we obtain the minimum hop count locally.

It is possible that there is no direct link from one node to another node. The link may go through another member node. So the loop includes the new node. After Node \( k-1 \) joins the loop, Node 0 reports to the rendezvous point the loop information which includes the node ID of all nodes in the loop.

The complete loop forming procedures are summarized as shown in pseudo code in Table II.

In the beginning, the first three nodes form a loop. The loop information is recorded in Node 0. New node \( a \) is added to the loop. All nodes in the loop traceroute to this new node and find the nearest Node \( b \) with the minimum hop count \( ba \). Node \( b \)’s adjacent Nodes \( e \) and \( c \) also obtain the hop counts \( ea \) and \( ca \) respectively. Node \( e \) traceroutes to Node \( b \) and obtains the hop count \( eb \). Node \( b \) traceroutes to Node \( c \) and obtains the hop count \( bc \). Then the optimal algorithm will apply to add the new node into the loop. Sum the hop count \((eb+ba+ac)\) and \((ea+ab+bc)\). If \((eb+ba+ac) > (ea+ab+bc)\), then optimal loop goes through \( e \rightarrow a \rightarrow b \rightarrow c \). Otherwise, the loop goes through \( e \rightarrow b \rightarrow a \rightarrow c \). The loop information is recorded. In this way, we obtain the minimum hop count locally.

The rationale behind this method is that the node with smallest hop count is likely to be close physically.

From the above description of the algorithm, we know that the loop, which is finally chosen by rendezvous point as DIN-Loop, does not affect the correctness and scalability of
the underlying multicast tree. DIN-Loop based on the hop count can create a multicast infrastructure by avoiding unnecessary high latency hops, reducing traffic overhead and responding to dynamic conditions. In general, DIN-Loop complements the multicast tree and can be used to improve application-level tree-based multicast performance.

### Table II. Pseudo Code for Loop Formation Algorithm

```plaintext
// k is the number of nodes to form the loop
for (i=0; i<k; i++)
{
    if (k<3)
    {  The new node a joins in the loop;
        Record the information of loop;
    }
    else
    {
        All nodes in the loop traceroute to this new node a and find the nearest Node b with the minimum hop count ba;
        Node b’s adjacent Node e and c also obtain the hop count ea and ca respectively;
        Node e traceroutes to Node b and obtains the hop count eb;
        Node b traceroutes to Node c and obtains the hop count bc;
        if ((eb+ba+ac) > (ea+ab+bc))
        {  The loop goes through e→a→b→c;
            Record the information of loop;
        }
        else
        {  The loop goes through e→b→a→c;
            Record the information of loop;
        }
    }
}
```

### E. DIN Node State

Every node maintains a small routing table (Fig. 6). As the common tree, there are the entries for Parent Node and Child Nodes. The entries of Child Nodes are the node IDs of associated child-nodes. The entries for Precedence-DIN-Node and Successor-DIN-Node are empty by default. When the node is selected by the rendezvous point as a DIN Node and gets the updating node state instruction from the rendezvous point, the entry of Precedence-DIN-Node is the ID of the incoming DIN Node from whom the DIN Node receives the message in the DIN-Loop, the entry of Successor-DIN-Node is the ID of the outgoing DIN Node to whom the DIN Node forwards the message in the DIN-Loop. DIN Nodes forward messages to all nodes inside and outside the loop. The nodes within the DIN-Loop may receive the same message more than one time. The repeating message is discarded.

<table>
<thead>
<tr>
<th>Multicast Group Id</th>
<th>abcdefg</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Routing Table</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Parent Node</strong></td>
<td>xxxxxxxx</td>
</tr>
<tr>
<td><strong>Child Node</strong></td>
<td>cccccc1  cccccc2</td>
</tr>
<tr>
<td><strong>Precedence-DIN-Node</strong></td>
<td>yyyyyyyyy</td>
</tr>
<tr>
<td><strong>Successor-DIN-Node</strong></td>
<td>zzzzzzzzz</td>
</tr>
</tbody>
</table>

Figure 6. Node state.
F. Leaving the DIN-Loop

When a DIN Node decides to leave the DIN-Loop, it sends a quit message to its two adjacent nodes – Precedence-DIN-Node and Successor-DIN-Node. It tells Precedence-DIN-Node about its Successor-DIN-Node and tells Successor-DIN-Node about its Precedence-DIN-Node. So the Precedence-DIN-Node and Successor-DIN-Node will update the node state and set up the connection between them. The quitting DIN Node also informs the rendezvous point so that the rendezvous point uses the number of remaining DIN Nodes to decide if the DIN-Loop must be updated based on Table V in Section IV.

G. Fault Detection

If a DIN Node or a link fails, the DIN-Loop is broken. The nearest DIN Node, which detects the failure, will loop back the message with its ID. Another adjacent DIN Node, which also detects the failure, receives the loop-back message and knows the opposite DIN Node. Then it will connect to the opposite side DIN Node again to form a new DIN-Loop (Fig. 7).

![Figure 7. DIN-Loop recovery.](image_url)

If a child-node, that is associated with the failed DIN Node, detects the failure of the DIN Node, it will broadcast the joining message. Any live DIN Node in the DIN-Loop, that receives the message, adds it into the entry of child-nodes in the node state. The DIN Node also informs the child-node to update its entry of Parent Node in its node state.

DIN Nodes who detect the failure also inform the rendezvous point so that the rendezvous point uses the number of remaining DIN Nodes to decide if the DIN-Loop must be updated based on Table V in Section IV.

IV. EXPERIMENTAL RESULTS

In P2P tree-based multicast, the multicast trees are built at the application level and the rendezvous point is the root of the multicast tree. Subsequently, every member, who wants to join the multicast group, sends a message to the rendezvous point, and the registration is recorded at each node along the path in the P2P overlay network. In this way, the multicast tree is formed. When multicasting, a sender sends data to the rendezvous point and the rendezvous point forwards the data along the multicast tree to all members. In DINCast, DIN-Loop with multiple DIN Nodes is used to replace a single rendezvous point. A node sends the message to the rendezvous point. The first DIN Node that receives the message will forward the message to its child-nodes, parent nodes and the neighbor DIN Node along the DIN-Loop. The neighbor DIN Node will forward the message to its associated child-nodes, parent nodes and its neighbor DIN Node along the DIN-Loop. The process repeats itself until all DIN Nodes receive the message or when the lifetime of the message expires. The nodes inside the DIN-Loop forward the message toward the rendezvous point. These nodes inside the DIN-Loop may receive the same message more than one time and the repeating message is discarded. We assume that all nodes in the tree are the multicast group members, which are counted as total node number. Since all nodes in the loop can play the role of rendezvous point, the load balancing is obvious and we do not do detailed simulation on this part. This section focused on three parts. First, we did the simulation to prove that DINCast could reduce the delay compared with the tree-based multicast. From the simulation results, the conditions for an optimal DINCast performance were derived. Second, based on the results, a decision table was set up so that the rendezvous point can use it to select a suitable loop as
DIN Loop with the best performance. Finally, simulation was also conducted to investigate the impact of Data Persistency (see Sub-section B, Section II) on the performance of retrieving messages from the DIN-Loop instead of from the sender.

A. DINCast Reducing Delay

We use hop count in the application level to compute the delay from the sender sending out a message to the receivers receiving it. We assume that all nodes in the multicast tree are the receivers. We measured the number of hops using both DINCast and tree-based multicast. Ratios max is the ratio between the maximum hop count using DINCast and the maximum hop count using tree-based multicast, and Ratios ave is the ratio between the average hop count using DINCast and the average hop count using tree-based Multicast.

The simulations ran on the overlay network topologies, which were generated using the Georgia Tech [29] random graph generator according to the transit-stub model (as described in [21]). One random node in the transit domain was used as the rendezvous point. The rendezvous point informed the nodes to form the DIN-Loop. In DINCast, a node sends a message to the DIN-Loop and then the DIN-Loop will forward it to all the members of the group. In contrast, with tree-based multicast, the rendezvous point is the root of multicast tree. All messages are sent to the rendezvous point and the rendezvous point forwards it to all the members of the group. The simulation software program was made to compute the maximum hop count and average hop count in DINCast and the tree-based multicast.

1) Number of DIN Nodes

We used the graph generator to generate different overlay network topologies. The number of DIN Nodes in the DIN Loop was changed and the total node number was fixed, i.e., 100. The probability that a direct link between each pair of DIN Nodes exists was 0.5. The corresponding results were obtained. The same overlay network topologies were applied to tree-based multicast. Finally, we computed the Ratios max and Ratios ave. The results are shown in Fig. 8.

![Figure 8. Effect of number of DIN Nodes on maximum & average delay.](image)

Ratios ave is less than Ratios max in most scenarios. In Fig. 8, the reference line means that the performance of DINCast on this line is the same as that of the tree-based multicast. Under the reference line, DINCast’s performance is better than tree-based multicast’s for both Ratios max and Ratios ave. As the number of DIN Nodes in the DIN-Loop increases, the comparative performance gain of DINCast reduces. The deeper the root, the better the performance of DINCast. Based on these two figures, depending on the number of DIN Nodes being chosen, DINCast can achieve better performance than tree-based Multicast.

We simulated the different hop depths from 1 to 24. For simplicity, only 0 hop depth, 4 hop depth, 8 hop depth and 12 hop depth are shown in Fig. 8. Fig. 8 is obtained under the conditions that the total node number is fixed and the probability of direct link between any pair of DIN Nodes is fixed. In the following subsections, we investigate how these conditions affect DINCast.
2) Total Node Number

We used the same random graph generator to generate the different overlay network topologies so that the number of DIN Nodes in the DIN Loop was unchanged, i.e., 25 and the total node number was changed. The probability that a direct link between each pair of DIN Nodes exists was 0.5. The results of the simulation are shown in Fig. 9.

For different total node number, the Ratio_max and Ratio_ave remain almost the same. The total node number does not affect the performance very much. Thus, we can conclude that DINCast’s performance is not affected by the number nodes and is therefore scalable.

In this simulation, 25 DIN Node form a DIN-Loop. So the maximum hop delay from the first DIN Node forwarding a message in the DIN-Loop to the last DIN Node receiving a message from the DIN Loop is 24. If the first DIN Node forwards the message to the rendezvous point and then the rendezvous point forwards it to all DIN Nodes (multicast tree method), the maximum hop delay is 24 because the DIN-Loop is 12 hop depth. Therefore, in Fig. 9, the Ratio_max for 12 hop depth is close to 1. However, the Ratio_ave for 12 hop depth is less than 1. So the performance of DINCast when the DIN-Loop is 12 hop depth is the same as or better than that in the multicast tree.

![Figure 9. Effect of total node number on maximum & average delay.](image)

3) Probability of Direct Link

We used the graph generator to generate different network topologies so that the number of DIN Nodes in the DIN Loop was unchanged, i.e., 25, and the total node number was 100 and 7525 respectively. The probability that a direct link between each pair of DIN Nodes exists was changed. The simulation results are shown in Fig.10.

![Figure 10. Effect of probability for direct link on maximum & average delay.](image)
For 0 hop depth, the probability of direct link affects the DINCast performance more. The best performance of DINCast is when the probability is 0.75. But the variation is within a small range. The deeper the root, the less the variation. Thus, the probability does not affect DINCast’s performance much. In addition, on the left of Fig. 10, two crossing points on the curves related to 0 hop depth and almost overlapping curves related to 12 and 24 hop depths respectively also verify the above subsection results that the total node number does not affect the performance very much.

From the above results, it is clear that the total node number and the probability of the direct link between any pair of DIN Nodes are not dominating factors. Their effect is so marginal that it can be ignored. Since the 24-hop depth is good enough and the overlay network topology with more than 24 hop depth is rare, we choose 24 as the highest hop depth and use TTL=24 (i.e., n=24), in Step 1 for the discovery of DIN Nodes (Sub-section B, Section III).

From the above simulation results, the rendezvous point can decide which nodes to form the DIN-Loop based on Fig. 8. The details are described in the following section.

B. Decision Table

A decision table is used to facilitate the rendezvous point to make a decision on which loop is the best among n loops that were recorded in Table I. We present the details of the decision table below.

First, we define the improvement Rates $\gamma$ and $\eta$.

$$
\gamma = \frac{\text{max}_{\text{hop of _TreeBasedMulticast}} - \text{max}_{\text{hop of _DINCast}}}{\text{max}_{\text{hop of _TreeBasedMulticast}}}
$$

$$
= 1 - \frac{\text{Ratio}_{\text{max}}}{\text{Ratio}_{\text{max}}}
$$

$$
\eta = \frac{\text{average}_{\text{hop of _TreeBasedMulticast}} - \text{average}_{\text{hop of _DINCast}}}{\text{average}_{\text{hop of _TreeBasedMulticast}}}
$$

$$
= 1 - \frac{\text{Ratio}_{\text{ave}}}{\text{Ratio}_{\text{ave}}}
$$

From Fig. 8, we obtained Table III and Table IV. The best performance is given by the maximum positive value. The worse performance is given by negative values in the tables.

<table>
<thead>
<tr>
<th>Hop Depth</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>10</th>
<th>...</th>
<th>25</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.125</td>
<td>0.083</td>
<td>0</td>
<td>-0.15</td>
<td>...</td>
<td>-0.583</td>
<td>...</td>
</tr>
<tr>
<td>1</td>
<td>0.300</td>
<td>0.214</td>
<td>0.154</td>
<td>0.042</td>
<td>...</td>
<td>-0.357</td>
<td>...</td>
</tr>
<tr>
<td>2</td>
<td>0.417</td>
<td>0.313</td>
<td>0.267</td>
<td>0.179</td>
<td>...</td>
<td>-0.188</td>
<td>...</td>
</tr>
<tr>
<td>3</td>
<td>0.500</td>
<td>0.389</td>
<td>0.353</td>
<td>0.281</td>
<td>...</td>
<td>-0.188</td>
<td>...</td>
</tr>
<tr>
<td>4</td>
<td>0.563</td>
<td>0.450</td>
<td>0.421</td>
<td>0.311</td>
<td>...</td>
<td>-0.050</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>8</td>
<td>0.708</td>
<td>0.607</td>
<td>0.593</td>
<td>0.558</td>
<td>...</td>
<td>0.321</td>
<td>...</td>
</tr>
<tr>
<td>12</td>
<td>0.781</td>
<td>0.694</td>
<td>0.686</td>
<td>0.662</td>
<td>...</td>
<td>0.472</td>
<td>...</td>
</tr>
<tr>
<td>24</td>
<td>0.875</td>
<td>0.817</td>
<td>0.822</td>
<td>0.798</td>
<td>...</td>
<td>0.692</td>
<td>...</td>
</tr>
</tbody>
</table>

Based on Table III, Table IV and the loop information in Table I about the hop depth and the number of nodes in the loop, the rendezvous point can choose a suitable loop as the DIN-Loop for the best improvement rate. Besides delay, another evaluation criterion is load balance which can be achieved. If two improvement rates are close, then the one with more DIN Nodes will be selected. In order to simplify the decision tables, we used rank instead of the actual value. The range of improvement rate from 0.75 to 1 is Rank I. The range of improvement rate from 0.50 to 0.75 is Rank II. The range of improvement rate from 0.25 to
0.5 is Rank III. The range of improvement rate from 0 to 0.25 is Rank IV. The range of improvement rate below 0 is Rank V. Rank I is the highest rank. If the improvement Rates $\gamma$ and $\eta$ are in different ranks, the lower rank is used. In this way, Table III and Table IV are converted into Table V.

### TABLE IV. IMPROVEMENT RATE $\eta$

<table>
<thead>
<tr>
<th>Number of DIN Nodes</th>
<th>Hop Depth</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>0.088</td>
</tr>
<tr>
<td>1</td>
<td>0.218</td>
</tr>
<tr>
<td>2</td>
<td>0.433</td>
</tr>
<tr>
<td>3</td>
<td>0.544</td>
</tr>
<tr>
<td>4</td>
<td>0.609</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0.751</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>0.817</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>0.900</td>
</tr>
</tbody>
</table>

### TABLE V. DECISION TABLE FOR THE RENDEZVOUS POINT

<table>
<thead>
<tr>
<th>Number of DIN Nodes</th>
<th>Hop Depth</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>IV</td>
</tr>
<tr>
<td>1</td>
<td>IV</td>
</tr>
<tr>
<td>2</td>
<td>III</td>
</tr>
<tr>
<td>3</td>
<td>II</td>
</tr>
<tr>
<td>4</td>
<td>II</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>II</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>I</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>I</td>
</tr>
</tbody>
</table>

There are three criteria when the rendezvous point decides which nodes to form the DIN-Loop:

- The higher rank is the first choice.
- If the ranks are the same, the number with more DIN Nodes is chosen for better load balance.
- If the ranks are the same and the number of DIN Nodes is the same too, the higher hop depth is chosen because the improvement rate is increased with the depth.

Rank V is out of consideration since the DIN-Loop with 2 DIN Nodes in 0 hop depth can always be found. This DIN-Loop achieves about 8.8% improvement in delay in terms of hop count and reduces the burden at the rendezvous point by half.
C. Retrieving Messages from DIN-Loop

In the previous simulation of DINCast, the sender sent the message to the DIN-Loop and the DIN-Loop forwarded it to the receivers. Since all messages are made to circulate in the DIN-Loop continuously, during the lifetime of the message, the receivers can retrieve the message from the DIN-Loop. In this subsection, simulation was conducted to investigate the impact of Data Persistency on the performance of retrieving messages from the DIN-Loop instead of from the sender.

We measured the hop count to deliver a message to each member of a group using the DIN-Loop and tree-based multicast. The same overlay network topologies in Subsection A.1 were used here. The results are shown in Fig. 11. Ratio_max is the ratio between the maximum hop count from DIN-Loop to receivers and the maximum hop count using tree-based multicast, and Ratio_ave is the ratio between the average hop count from DIN-Loop receivers and the average hop count using tree-based Multicast.

![Figure 11. Delay ratio of getting messages from DIN-Loop over tree-based multicast.](image1)

When Fig. 11 was compared with Fig. 8, we found that Ratio_max and Ratio_ave in Fig. 11 are lower than Ratio_max and Ratio_ave in Fig. 8 respectively.

We then measured the hop count to deliver a message to each member of a group using DIN-Loop. We also measured the hop count from the sender sending the message to the DIN-Loop to receivers receiving it in DINCast. The results are plotted in Fig. 12.

![Figure 12. Performance improvement in retrieving messages directly from DIN-Loop.](image2)
In Fig. 12, the solid line $\text{Ratio}_{\text{max}, 3}$ is the ratio between the maximum hop count from DIN-Loop to receivers and the maximum hop count from the sender sending the message to the DIN-Loop to receivers receiving it in DINCast. The dotted line $\text{Ratio}_{\text{ave}, 3}$ is the ratio between the average hop count from DIN-Loop to receivers and the average hop count from the sender sending the message to the DIN-Loop to receivers receiving it in DINCast.

It is clear that the delay of retrieving messages directly from the DIN-Loop is less than the delay of getting messages from the sender in DINCast when the number of DIN Nodes is less than or equal to 25. Thus, retrieving messages from the DIN-Loop further reduces the delay. Retrieving messages from the DIN-Loop improves the performance of DINCast from 4% to 33%.

V. CONCLUSION

Efficient communications among the many computers participating in P2P computing remains a problem when they are geographically distributed across multiple network domains. One way in which group communications can be supported is by using the multicast concept. However, the use of IP multicast has been limited because of issues related to scalable interdomain routing protocols, tracking group membership and so on. Because of problems facing the deployment of a network-level multicast service, application-level multicast has gained in popularity. However, it also suffers from several disadvantages. First, using application-level multicast with topology-awareness still increases the delay to deliver messages relative to IP multicast. Second, in the P2P tree-based multicast, the rendezvous point can potentially be subjected to overload and single-point of failure.

In this paper, to overcome the disadvantages of application-level multicast, we proposed DINCast and applied it to application-level tree-based multicast. DINCast complements the tree-based multicast. DINCast uses the DIN-Loop instead of the rendezvous point as multicast sources.

We described the details of DINCast and steps to establish DINCast over the existing application-level multicast tree. The experimental results show that DINCast can reduce the hop-count inefficiency with the application-level multicast. DINCast’s performance is not affected by the number of nodes and is therefore scalable. The probability for direct link between any pair of DIN Nodes is also not a dominating factor. The dominating factors are the number of DIN Nodes and the hop depth. Based on these factors, the rendezvous point can select a suitable loop among $n$ loops as the DIN-Loop with the best performance in the current network topology. Retrieving messages from the DIN-Loop further reduces the delay and therefore improves the performance of DINCast.

Currently, we only consider the routing service as a complement to tree-based Multicast and ignore any upper-level service like bandwidth or congestion control. Our future work will do some research on the optimal use of the bandwidth. We will also explore applying our solution to real applications of large-scale P2P computing in a Grid environment. Research into long-term Grid architecture has been proposed in [7]. One of the projects in this research program involves the development of massively scalable middleware for collaborative virtual communities. This project will combine the experience with implementing the multicast tools that make up the access grid, but replacing the multicast IP substrate with P2P systems. This motivates us to investigate a DINCast based P2P infrastructure which can provide a collaborative environment. Such an environment will enable groups of people to find, talk to, see and share ideas with other groups. We will also investigate wide-area multicast for a large number of media streams.

REFERENCES


[29] E. Zegura, K. Calvert and S. Bhattacharjee, How to model an internetwork, in Proc of IEEE Infocom, April 1996


H. Q. Guo obtained B.Eng and M.Eng from Tianjin University in 1989 and 1991 respectively, and obtained Master of Engineering from the National University of Singapore (NUS) in 2001. She was a senior engineer, Kent Ridge Digital Labs, Singapore from 2000 to May 2001. She was a senior research staff, NUS from June 2001 to March 2003. She is an Ordinary Member of Singapore Computer Society from 2000.

She is currently a PhD student in the Department of Electrical & Computer Engineering, NUS. Her research interests include networked middleware, high capacity optical network system, P2P computing, and multimedia communication system.

L. H. Ngoh obtained BSc (Honors) in Computer Systems Engineering from University of Kent at Canterbury, UK in 1986 and obtained MSc and Ph.D. degrees in Computer Science from University of Manchester, UK in 1987 and 1989 respectively. From 1990 to 1992, he was a member of technical staff at Hewlett-Packard Research Labs, Bristol, UK. He joined Institute of Systems Science (ISS), Singapore from 1992 to 1997. He was a Member of British Computer Society (MBCS) in 1992. He was an Organizing Committee member of the IEEE Singapore International Conferences on Networks for 1998, 2000 & 2002 and a founding member of the Singapore Advance Research and Education Network (SingAREN).

He is currently a Senior Member Research Staff, Institute for Infocomm Research (I²R), formerly known as Kent Ridge Digital Labs (KRDL). His research interests are in the areas in computer networking including: Network Management, Advanced Internet Technology, P2P Computing & Sensor Networks.

W. C. Wong obtained the B.Sc. (Honors) and Ph.D. degrees in electronic and electrical engineering from Loughborough University, UK, in 1976 and 1980 respectively. He was a Member of Technical Staff at AT&T Bell Laboratories, Crawford Hill Laboratory, Holmdel, New Jersey, USA, from 1980 to 1983. He joined the Department of Electrical and Computer Engineering, National University of Singapore (NUS), in 1983. He was a visiting consultant to AT&T Bell Laboratories on numerous occasions between 1984 and 1992. He was Professor in the Electrical & Computing Engineering Department and Director of Computer Centre, NUS. He is currently Executive Director of I²R. His research interests are in high capacity mobile radio communication systems, sources-matched digital transmission systems, multimedia communication system and multimedia signal compression.

He was a co-winner of the IEE Marconi Premium Award in 1989. He is a recipient of IEEE Millenium Award (2000) and e-innovator Awards 2000, Open Category. He holds four patents and has published over 150 papers in international journals and conferences.

J. G. Tan received the B.Eng (honors) degree from the National University of Singapore in 1989 and the Post-graduate Dip. in Data Communications and Networking Software from the Information Communication Institute of Singapore(ICIS) in 1992. She has been with the Institute for Infocomm Research (I²R), formerly known as Kent Ridge Digital Labs (KRDL) since 1992. At I²R, she was involved in R&D of networked multimedia applications for broadband networks. From 1997 to 1999, Ms. Tan was the project leader in multi-party videoconferencing and broadband network. From 1999 to 2001, she was the team leader of an integrated Active Internet Monitoring System (AIMS). Ms. Tan is presently with the Services and Applications Division in I²R where she is involved in the research on innovative ways of supporting in-network processing in distributed sensor networks to provide new types of applications and services. Her research interests are in the areas of high-speed networks, network protocols and distributed sensor networks.