DINCast: Optimizing Application-Level Shared-Tree Multicast

Huaqun Guo, Lek Heng Ngoh
Institute for Infocomm Research, A*STAR
1 Fusionopolis Way
#21-01 Connexis (South Tower)
Singapore 138632
{guohq, lhn}@i2r.a-star.edu.sg

W. C. Wong, Ping Yang, Nazreen Beevi
Department of Electrical & Computer Engineering,
National University of Singapore
2 Engineering Drive 4,
Singapore 117576
elewwcl@nus.edu.sg

Abstract—Application-level multicast suffers some disadvantages in terms of high multicast delay, overloading at Rendezvous Point (RP) and single-point of failure. In this paper, we propose DINCast to optimize application-level shared-tree multicast. The general idea of DINCast is to form a special logical data loop and use this loop instead of the RP as multicast sources. To demonstrate its effectiveness, we first analyze multicast delay with the inclusion of transmission delay, queuing delay and propagation delay of a typical symmetrical shared-tree multicast. We evaluate three placement schemes of DINCast. Our simulation results show that DINCast is effective in optimizing the multicast delay over the shared-tree multicast and the reduced rate of multicast delay may be up to 50%. Next, for any multicast tree, we carry out further simulations to optimize multicast delay. The experimental results show that DINCast can achieve better performance than original shared-tree multicast.

Keywords-DINCast; Application Level Multicast; Shared-Tree Multicast; Multicast Delay; Data Loop

I. BACKGROUND & MOTIVATION

In the age of multimedia and high-speed networks, there are many applications of sending information to a selective, usually large, number of clients. Common examples of such applications include audio/video conferencing, distance learning, video-on-demand, distributed interactive games, data distribution, service location/discovery, collaborative computing, and so on [1]. To support such applications, multicast is considered as a very efficient mechanism since it uses some delivery structures to forward data from senders to receivers, with the aim that the overall utilization of resources in the underlying network is minimized [2].

There has been a number of works reported on application-level multicast, e.g., Scribe [3], CAN-Multicast [4], Bayeux [5], YOID [6], and so on. Each uses a different overlay network to implement application-level multicast, using either flooding (CAN-Multicast) or tree-building scheme (e.g. Scribe, Bayeux, YOID etc.). The flooding approach creates a separate overlay network per multicast group and broadcasts messages within the overlay. The tree approach uses a single overlay and builds a tree topology first [7]. In [7], the results showed that the tree-based approach of Scribe consistently outperformed the flooding approach of CAN-Multicast. At the heart of all tree-based approaches are two basic types of trees: source tree and shared tree [8]. Source tree requires more memory O(G*S) for routing table (G is the number of multicast groups and S is the average source number of each multicast group), forms an optimal path from source to receivers, and is good for small number of senders and many receivers. Shared tree, on the other hand, requires less memory O(G) for routing table, forms sub-optimal path from source to receiver (but may introduce extra delay from source to root), and is good for many senders with low bandwidth environment [9]. Shared trees are considered an important part of the multicast routing architecture because only one routing table entry is needed for an entire group, instead of one per source [8].

Compared with Internet IP-level 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.

However, application-level multicast also suffers certain drawbacks. First, due to the fact that application-level multicast is implemented at host-level and the underlying physical topology is hidden, even with topology-awareness [10], application-level multicast can still increases the delay to deliver messages when compared with IP multicast. This can lead to inefficient routing because every application-level hop could potentially be between two geographically distant nodes. Second, in shared-tree multicast, the multicast tree is built at the application level and the Rendezvous Point (RP) is the root of the multicast tree. The RP can potentially be subjected to overloading and single-point of failure. For example, in Real-Time Conferencing Protocol, to provide a multicast back-channel for a group of receivers will result in serious source implosion problem [11].

In this paper, we design a unique solution referred to as Data-In-Network multicast (DINCast) to optimize the application-level shared tree multicast, and overcome the drawbacks described above. The rest of the paper is organized as follows. Section II describes the overview and key features of DINCast. Section III analyzes multicast delay and presents three schemes to optimize symmetrical shared-tree multicast. Section IV presents the generalization of DINCast to optimize multicast delay based on hop count for any shared-tree multicast. Section V outlines our conclusion.
II. PROPOSED SOLUTION: DINCast

A. Overview

DINCast is a network data ‘loop’ formed on top of an existing application-level shared multicast tree. In DINCast, the novel part is that multiple DIN Nodes, which are hosts, form a logical loop and DINCast uses the loop instead of one single RP as multicast sources. In DINCast, any multicast message is sent to the loop. From the loop, the messages are then forwarded to all members. At the same time, the multicast message circulates in the loop, that is, it “lives” for a certain period of time in the loop. In other words, DINCast uses the loop as a networked wide-area cache and utilizes the propagation delay to “buffer” data in the loop. Thus, DINCast can provide real-time update and offline access at any time since the data in the loop is persistent.

An example of the proposed DINCast is shown in Fig. 1, where DIN Nodes 0, 1, 2, and 3 form a logical loop 0→2→1→2→3→0 (thick line). When a source node sends a message to the loop, the DIN Node that receives the message (hereafter referred to as “First DIN Node”) will forward the message to its child-nodes (dotted arrows, outside of the loop), its parent node (inside of the loop if any, not indicated in the figure) and the neighbour DIN Node along the loop. The associated child-nodes are organized in sub-tree type. The neighbour DIN Node will forward the message to its associated child-nodes, the parent node, if any, and its neighbour DIN Node along the loop. The process repeats itself until all DIN Nodes receive the message. In this scheme, one loop corresponds to one RP/one multicast tree. If there are multiple multicast trees, DINCast will build multiple loops.

B. Key Features of DINCast

Notice that the link load along the loop in DINCast is higher than that in application-level shared-tree multicast because DINCast may continuously circulate the data in the loop. Thus, we only circulate a small amount of important messages, such as, control messages, in the loop. The link load inside the loop of DINCast is lower than that in application-level shared-tree multicast, while the link load outside the loop remains the same for both scenarios. DINCast therefore retains the advantages of seamless and easy deployment offered by application-level shared-tree multicast.

DINCast further offers three more advantages that lighten or overcome the disadvantages of existing application-level shared-tree multicast.

1) Minimize Inter-node Delay: In DINCast, the messages are forwarded to all members via a loop instead of the RP which can be deep in the network and thus further away from the members. Whenever a message is received by one of DIN Nodes, this DIN Node will forward to its associated child-nodes immediately and its neighbour DIN Node. So the delay for the nodes to receive the message can be reduced.

2) Robustness with Load Balancing: All DIN Nodes in the loop can play the role of the RP so that DINCast relieves the bottleneck of the RP. Thus, DINCast achieves load balancing and avoids the single-point of failure.

3) Data Persistency: In DINCast, the message circulates in the loop continuously and is therefore exhibits persistency in the loop. During the lifetime of the message, nodes can get the message from the 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 RP. Data persistency is needed in various real applications. For example, mobile devices/components are sometimes intermittently disconnected 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). With data persistency, these mobile devices/components can still get the message from the loop at a later time.

III. MULTICAST DELAY ANALYSIS

The objective of the multicast delay analysis is to optimally form a DINCast loop based on a shared multicast tree so as to reduce overall multicast delay with less formation overhead.

To calculate the multicast delay, we include transmission delay, queuing delay and propagation delay. For easy analysis, we make the following assumptions:

- There are \( h \) messages to be sent to all nodes in the multicast tree.
- All multicast messages have the same size \( S_{\text{msg}} \).
- Multicast tree is symmetric and there are \( L \) multicast tree levels.
- The non-leaf nodes have the same number of branches \( n_1 \).
- Source nodes are leaves and belong to the family of DIN Nodes.
- The propagation delay per hop in the application level is the same and assigned as \( f_{\text{hop}} \).

We further define the value of hop depth in DINCast as \( l_{\text{loop}} \), which is the value of hop count from the RP to DINCast, and assign the number of DIN Nodes as \( N \) (see Fig. 5).

Before we present the analysis, we further define four functions, namely Functions A to D, representing different
data transmission patterns. In Function A, a node receives a multicast message and forwards it to another node. In Function B, a node receives one multicast message, copies \( n_1 \) messages and forwards them to its \( n_1 \) branches nodes. In Function C, a DIN Node receives one multicast message, copies \( (n_1+2+k) \) messages and forwards them to its \( n_1 \) branches (child-nodes), its one parent node in the multicast tree, its one neighbour DIN Node along the loop and \( k \) other nodes at the same level of multicast tree. Finally in Function D, a parent/grandparent of a DIN Node receives one multicast message, copies \( (j+1+k) \) messages and forwards them to its \( j \) branches (child-nodes), its one parent node in the multicast tree, and \( k \) other nodes at the same level of the multicast tree.

### A. Delay Analysis at RP & the First DIN Node

A multicast message departing from one node and arriving at another node can be seen as departing from one queue and entering another queue. We assume that there are \( h \) multicast messages which will arrive at the RP for shared-tree multicast or DIN Node for DINCast. Thus, the queuing delay at the RP or DIN Node becomes important and we therefore include the queuing delay at the RP and DIN Node when calculating multicast delay. For simplicity, we assume that before the messages arrive at the RP or DIN Node, there is no queuing delay but only transmission delay at the nodes. After the messages depart from the RP or DIN Node, we only consider the transmission delay because the inter-arrival times at the second queue are strongly correlated with the transmission times in the first queue, and if the message transmission times are equal, each message arriving at the second queue will complete its transmission at or before the time the next message arrives, and therefore there is no waiting time at the second queue [12]. Detail delay analysis is presented next.

1. We analyze the delay at the RP. The operation of the RP can be described as follows. The RP works as Function B wherein it receives \( h \) multicast messages, copies \( n_1 \) messages and forwards them to its \( n_1 \) branch nodes (Fig. 2). The overall time can be divided into a portion used for receiving a multicast message (arrival interval) and another portion used for multicast message transmission (transmission interval).

The incoming/outgoing bandwidth of the RP is \( V_{RP} \). The arrival processes of all \( h \) multicast messages are independent Poisson with rate \( \lambda_1 = n_1 \lambda \). The service rate is \( \mu = V_{RP}/S_{pm} \). The utilization factor \( \rho_1 \) is:

\[
\rho_1 = n_1 \lambda / \mu = n_1 \rho < 1
\]  

The first and second moments of the packet transmission times are \( X = 1/\mu \) and \( X^2 \), respectively. Since all multicast messages have the same size, the transmission intervals of all messages have equal average length in steady state. Thus,

\[
X^2 = 1/\mu^2 \tag{2}
\]

We define \( A_n \) as the overhead or total retrieval time of all multicast messages. Since all \( h \) multicast messages have the same size, the receiving interval for each multicast message is a constant \( A_n/h \). We then have

\[
A_n = h * S_{pm} / V_{RP} \tag{3}
\]

Thus, based on an M/G/1 queue with vacations [12], we obtain,

\[
W_n = \frac{\lambda_1 X^2}{2(1-\rho_1)} + \frac{A_n (1+\rho_1 / h)}{2(1-\rho_1)} \tag{4}
\]

The total delay time \( T_{mb\_RP} \) for a multicast message at the RP with Function B is

\[
T_{mb\_RP} = W_n + \frac{1}{\mu} = \frac{\lambda_1 X^2}{2(1-\rho_1)} + \frac{A_n (1+\rho_1 / h)}{2(1-\rho_1)} + \frac{1}{\mu} \tag{5}
\]

Combining (1), (2) and (5), we obtain

\[
T_{mb\_RP} = \left[ \frac{n_1 \rho}{2(1-n_1 \rho)} + \frac{h(1+n_1 \rho / h)}{2(1-n_1 \rho)} + 1 \right] S_{pm} / V_{RP} \tag{6}
\]

2. We use queuing theory to analyze the delay at the First DIN Node with Function C when the multicast messages pass through the DIN Node. The incoming/outgoing bandwidth of the first DIN Node is \( V_{mc\_1} \). In Function C, a DIN Node receives \( h \) multicast messages, copies \( (n_1+2+k)*h \) messages and forwards them to its \( n_1 \) child-nodes, its one parent node, its one neighbour DIN Node, and \( k \) other nodes at the same level if such other nodes exist. (The maximum value of \( k \) may be 1, zero or \( n_1 \) for Schemes A, B or C in Sub-sections D, E and F respectively.) The arrival processes of all \( h \) multicast messages are independent Poisson with rate \( \lambda_2 = (n_1+2+k) \lambda \). The service rate is \( \mu = V_{mc\_1} / S_{pm} \) (Fig. 3). The utilization factor \( \rho_2 \) is:

\[
\rho_2 = (n_1 + 2 + k) \lambda / \mu = (n_1 + 2 + k) \rho < 1 \tag{7}
\]

Similarly, based on (5), the total delay time \( T_{mc\_DIN} \) for a multicast message at the First DIN Node with Function C is

\[
T_{mc\_DIN} = \frac{\lambda_2 X^2}{2(1-\rho_2)} + \frac{A_m (1+\rho_2 / h)}{2(1-\rho_2)} + \frac{1}{\mu} \tag{8}
\]
Combining (7) and (8), we obtain

\[
T_{\text{mc, div}} = \left\{ \begin{array}{ll}
\left( n_l + 2 + k \right) \rho \left( \frac{S_{pm}}{V_{nd}^m} \right) + \frac{h\left( n_l + 2 + k \right) \rho \rho \left( \frac{S_{pm}}{V_{nd}^m} \right)}{2 \left[ 1 - \frac{n_l + 2 + k \rho \rho \left( \frac{S_{pm}}{V_{nd}^m} \right)}{h} \right]} + 1 \\
\end{array} \right. \\
\]

(9)

In the following sub-sections, we calculate and compare multicast delay so as to obtain the optimal placement of DINCast on a shared tree. For easy comparison, we assume that every node has the same incoming bandwidth and outgoing bandwidth \( V_{nd} \). In addition, for simplicity, we introduce a propagation delay factor \( f_{\text{mp}} \) of the propagation delay over the delay at a node with Function A, i.e.

\[
t_{\text{mp}} = f_{\text{mp}} \cdot \frac{S_{pm}}{V_{nd}}, \quad f_{\text{mp}} > 0
\]

(10)

B. Delay Analysis in Shared-tree Multicast

For shared-tree multicast, an example of the multicast traffic is shown in Fig. 4. Node 7 is a source node. The dotted arrows are the traffic from the source to the RP. The solid arrows are the traffic from the RP to all nodes except the source node.

We assign \( T_{\text{maj, it}} \) as the delay from the source node to the RP, and \( T_{\text{maj, oit}} \) as the delay from the RP to Node i. Thus, the total multicast delay that all nodes except the source node on the multicast tree receive the multicast message is

\[
T_{\text{maj, total}} = T_{\text{maj, it}} + \sum_{j=1}^{k} \left( \left[T_{\text{maj, it}} - T_{\text{maj, oit}} \right] n_j \right)
\]

\[
= \left\{ T_{\text{maj, it}} + T_{\text{maj, oit}} \left( L - 1 \right) n_l \frac{S_{pm}}{V_{nd}^m} + L \cdot f_{\text{mp}} \cdot \frac{S_{pm}}{V_{nd}^m} \right\}
\]

(11)

C. Placement Strategy of DINCast

In this paper, we adopt the following placement strategy for DINCast. First, we find a suitable level in the multicast tree with suitable DIN Nodes number to form a loop. There is a trade-off in determining the position of DINCast and number of DIN Nodes in DINCast. If the hop depth is too small, the delay for nodes at levels more than the hop depth will increase. If the hop depth is too large, the delay for nodes at levels less than the hop depth will increase.

If the number of DIN Nodes is small, this implies that DINCast is far away from the sender and receivers, and load balance is not so good with just a few DIN Nodes. If the number of DIN Nodes is large, it would therefore be closer to the sender and receivers with better load balance, but the delay along the loop could be large. Based on the above, we consider three schemes to place a DINCast and integrate it with the original shared multicast tree. In Scheme A, we add a loop and serial links on a shared multicast tree, and the remainder of shared multicast tree is kept. In Scheme B, we only form a loop at a level of shared multicast tree and keep the rest part of shared multicast tree. In Scheme C, we add a loop and \( n_j \) links on a shared multicast tree, and the remainder of shared multicast tree is kept.

D. Placement Scheme A

In DINCast Scheme A, the multicast traffic is shown in Fig. 5. The loop is formed at the center position. Node 7 is a source node. The dotted arrows are the traffic from the source to the loop. The solid arrows are the traffic from the loop to all nodes except the source node.

At Level \( l_{\text{loop}} \) and above, non-DIN Nodes or non-ancestors/grandparents of DIN Nodes receive the multicast message from the DIN Nodes or parents/grandparents of DIN Nodes at the same level via serial links (dotted lines in Fig. 5). In Fig. 5, Node A and Node B are DIN Nodes. At Level \( l_{\text{loop}} \), Node x receives the multicast message from DIN Node A, Node y receives the multicast message from Node x, and Node z receives the multicast message from Node y. Node u receives the multicast message from DIN Node B, Node v receives the multicast message from Node u, and Node w receives the multicast message from Node v. At Level 2, Node d and Node e are the parents of DIN Nodes. Thus, Node c receives the multicast message from Node d, and Node f receives the multicast message from Node e. In this scheme, DIN Node receives \( h \) multicast messages, copies \( (n_k+2+k)^h \) messages and forwards them to its \( n_k \) child-nodes if any, its one parent node, its one neighbour DIN Node, and one other node at the same level if such other node exists (\( k=1 \)). If there is no other node linked to a particular DIN Node, \( k \) is equal to 0 for this DIN Node. A parent of DIN Node receives \( h \) multicast messages, copies \( (0+1+k)^h \) messages and forwards them its one parent node, and one other node at the same level if such other node exists. Thus, for (9), the maximum value of \( k \) is equal to 1, and the value of \( k \) for a particular DIN Node depends on the existence of the other nodes linked to this DIN Node.

Figure 4. Multicast traffic in shared-tree multicast.

Figure 5. Multicast traffic in Scheme A.
The delay \( T_{Dmd, i1} \) from the source to the loop in Function A is:

\[
T_{Dmd, i1} = (L - l_{\text{loop}}) \ast \frac{S_{pm}}{V_{nd}} + (L - l_{\text{loop}}) \ast f_{\text{mp}} \ast \frac{S_{pm}}{V_{nd}} \quad (12)
\]

Since the multicast tree is symmetric, we use the half delay along the loop for the calculation. So the delay \( T_{Dmd, i2} \) in the loop is:

\[
T_{Dmd, i2} = \begin{cases} 
T_{\text{wc, DIN}} + \left( (N - 1) \ast (n_1 + 2 + k) \ast \frac{S_{pm}}{V_{nd}} \right) \ast 0.5 \\
+ (N - 1) \ast f_{\text{mp}} \ast \frac{S_{pm}}{V_{nd}} & \text{if } (k_2 - N) \neq 0 \\
\end{cases} 
\]

(13)

In the following delay calculation, we introduce a function \( \text{floor}(m) \). The function \( \text{floor}(m) \) function means to obtain the largest integer that is less than or equal to \( m \).

(1) Multicast delay for nodes at Level \( l_{\text{loop}} \)

At Level \( l_{\text{loop}} \), there are \( N \) DIN Nodes and \( (n_1 - l_{\text{loop}} - N) \) non-DIN Nodes. Non-DIN Nodes receive the multicast message from the DIN Nodes via serial links in Function A. The delay encountered by non-DIN Nodes to receive the multicast message is:

\[
T_{\text{DIN, non-DIN}} = \sum_{k_1 = N + 1}^{n_1 \text{loop}} \left\{ \frac{T_{Dmd, i1} + T_{Dmd, i2} + \left( (k_2 - N) \ast f_{\text{mp}} \ast \frac{S_{pm}}{V_{nd}} \right)}{N} \right\} \quad (14)
\]

The delay for all nodes at Level \( l_{\text{loop}} \) to receive the multicast message is:

\[
T_{\text{Dmd, i1}} = T_{\text{DIN}} + T_{\text{Dmd, i2}} = (T_{Dmd, i1} + T_{Dmd, i2}) \ast N + T_{\text{Dmd, non-DIN}} \quad (15)
\]

(2) Multicast delay for nodes above Level \( l_{\text{loop}} \)

At Level \( l \) \((l \in [0, l_{\text{loop}} - 1])\), the number of parents or grandparents of DIN Nodes is:

\[
N_{l_1} = \begin{cases} 
\text{floor} \left( \frac{N}{n_1_{\text{loop}, i-1}} \right) + 1 & \text{if } \frac{N}{n_1_{\text{loop}, i-1}} \neq 0 \\
\text{floor} \left( \frac{N}{n_1_{\text{loop}, i-1}} \right) & \text{if } \frac{N}{n_1_{\text{loop}, i-1}} = 0 \\
\end{cases} 
\]

(16)

If \( n_1 \) is an even number, the multicast tree is symmetric and two Nodes at the centre have two parents at the upper level. So we refine the number of parents or grandparents of DIN Nodes at Level \( l \) as:

\[
N_l = \begin{cases} 
2 & (n_1 \text{ is an even number}) \& (N_l = 1) \& (l \neq 0) \\
N_l, & \text{if } \text{not the above condition} \\
\end{cases} 
\]

(17)

At Level \( l \), the nodes, which are not parents/grandparents of DIN Nodes, receive the multicast message from parents/grandparents of DIN Nodes at the same level via series links in Function D and Function A. So the delay for the nodes, which are not parents/grandparents of DIN Nodes, to receive the multicast message from parents or grandparents of DIN Nodes at Level \( l \) is:

\[
T_{\text{non-DIN, p}} = \left\{ \begin{array}{l}
\frac{(1 + k) \ast S_{pm}}{V_{nd}} + \text{floor} \left( \frac{k_4 - N_l}{N_l} \right) \ast S_{pm} \\
\text{while remainder of } \left( \frac{k_4 - N_l}{N_l} \right) \neq 0 \\
\left( \frac{(k_4 - N_l)}{N_l} \right) \ast S_{pm} \\
\text{while remainder of } \left( \frac{k_4 - N_l}{N_l} \right) = 0 \\
\end{array} \right\} + \sum_{k_4 = N_l + 1}^{l} \left\{ \frac{(1 + k) \ast S_{pm}}{V_{nd}} + \text{floor} \left( \frac{k_4 - N_l}{N_l} \right) \ast S_{pm} \right\} \\
\right.
\]

(18)

The delay for all nodes above Level \( l_{\text{loop}} \) (from Level \( l_{\text{loop}} - 1 \) to Level 0) to receive the multicast message is:

\[
T_{\text{Dmd, i1, loop}} = \sum_{l=l_{\text{loop}, i} - 1}^{0} \left\{ \frac{T_{Dmd, i1} + T_{Dmd, i2} + \left( (l_{\text{loop}} - l - 1) \ast f_{\text{mp}} \ast \frac{S_{pm}}{V_{nd}} \right)}{N} \right\} \quad (19)
\]

(3) Multicast delay for nodes below Level \( l_{\text{loop}} \)

At Level \( l \) \((l \in [l_{\text{loop}} + 1, L])\), there are \( (l_{\text{loop}} - l) \) DIN Nodes, which are child-nodes or grandchild-nodes of DIN Nodes. The delay for child-nodes or grandchild-nodes of DIN Nodes at Level \( l \) to receive the multicast message in Function B is:

\[
T_{\text{DIN, child}} = \left\{ \frac{T_{Dmd, i1} + T_{Dmd, i2} + \left( (l_{\text{loop}} - l - 1) \ast f_{\text{mp}} \ast \frac{S_{pm}}{V_{nd}} \right)}{N} \right\} \ast n_1_{\text{loop, i-1}} \ast N \\
\]

(20)

At Level \( l \), non-child-nodes or non-grandchild-nodes of DIN Nodes receive the multicast message from their own parents/grandparents at Level \( l_{\text{loop}} \). So the delay for non-child-nodes or non-grandchild-nodes of DIN Nodes at Level \( l \) to receive the multicast message is:
trees with \( n_1 \) branches rooted at DIN Nodes or parents/grandparents of DIN Nodes respectively. One example of Scheme C for binary multicast tree (\( n_1 = 2 \)) is shown in Fig. 7. Node A and Node B are DIN Nodes. Node x and Node y receive the multicast message from DIN Node A, and Node z receives the multicast message from Node y. Node u and Node v receive the multicast message from DIN Node B, and Node w receives the multicast message from Node v. Node d and Node e are the parents of DIN Nodes. Thus, Node c receives the multicast message from Node d, and Node f receives the multicast message from Node e. In Scheme C, a DIN Node receives \( h \) multicast messages, copies \((n_1 + 2 + h)h\) messages and forwards them to its \( n_1 \) child-nodes if any, its one parent node, its one neighbour DIN Node, and \( k \) other nodes if such other nodes exist. For (9), the maximum value of \( k \) is equal to \( n_1 \). If there is no other node linked to a DIN Node, \( k \) is equal to 0 for this DIN Node. If there are \( n_1 \) other nodes linked to a DIN Node, \( k \) is equal to \( n_1 \) for this DIN Node.

G. Simulation Results

In order for us to study and compare the above three placement schemes, simulations are conducted to study their behaviour.

We define the delay ratio as the multicast delay in DINCast over the multicast delay in shared-tree multicast. We carry out simulations for different sizes of multicast tree for values of \( L \) from 1 to 15, i.e., the multicast group size ranges from 3 to 65535 for \( n_1 = 2 \). We examine loops at each level with different DIN Nodes. Finally, we run simulations to obtain the delay ratio for various hop depths and various DIN Node numbers at each level. From a comprehensive search of all possible loops, we determine the optimal place to form the loop, which gives the minimum delay ratio.

\[
\begin{align*}
T_{\text{non-DIN}} &= \left( T_{\text{Dmd} \rightarrow \text{child}} + T_{\text{Dmd} \rightarrow \text{other}} \right) - T_{\text{source}} \\
&= \sum_{k=1}^{n_1} \left( T_{\text{DIN} \rightarrow \text{child}} + T_{\text{non-DIN} \rightarrow \text{child}} \right) - T_{\text{source}} \\
&= \sum_{k=1}^{n_1} \left( T_{\text{DIN} \rightarrow \text{child}} + T_{\text{non-DIN} \rightarrow \text{child}} \right) - \\
&\quad \left( T_{\text{Dmd} \rightarrow \text{child}} + T_{\text{Dmd} \rightarrow \text{other}} \right) \quad (22)
\end{align*}
\]

We define the delay ratio as the multicast delay in DINCast over the multicast delay in shared-tree multicast. We carry out simulations for different sizes of multicast tree for values of \( L \) from 1 to 15, i.e., the multicast group size ranges from 3 to 65535 for \( n_1 = 2 \). We examine loops at each level with different DIN Nodes. Finally, we run simulations to obtain the delay ratio for various hop depths and various DIN Node numbers at each level. From a comprehensive search of all possible loops, we determine the optimal place to form the loop, which gives the minimum delay ratio.
First, the optimal placement in Scheme C changes with the levels of multicast tree, while the optimal placement in Schemes A and B is almost fixed. Thus, the overhead to form the DINCast in Scheme C is the highest, followed by the overhead in Scheme A, and the overhead of DINCast in Scheme B is the lowest, since Scheme B uses the original tree to forward the multicast message to receivers.

Next, we plot the optimal delay ratio values at each optimal place for the three schemes in Fig. 8. From Fig. 8, it is clear that DINCast can reduce multicast delay since all delay ratios for three schemes are less than 1. When the level of multicast tree is 1, the optimal DINCast is formed at 1 hop depth and it is only one hop away from the RP. Thus, the performance of DINCast is slightly better than shared-tree multicast. When the level of multicast tree is 2, the optimal DINCast is formed at 2 hop depth with two DIN Nodes for Scheme A, Scheme C and binary tree of Scheme B, and at 1 hop depth with three DIN Nodes for multicast tree with three branches of Scheme B. The DINCast gets closer to the majority of receivers, and hence DINCast performance becomes better and the delay ratio reduces.

When the level of the binary multicast tree ($n_1=2$) is 3, the optimal DINCast is formed at 3 hop depth with two DIN Nodes for the three scheme. Schemes A and C add additional links, and hence the delay ratio continuously reduces, while Scheme B uses the original multicast to forward the multicast message so that the delay ratio increases. When the level of multicast tree ($n_1=3$) is 3, the optimal DINCast is formed at 2 hop depth with three DIN Nodes for Scheme A, at 1 hop depth with three DIN Nodes for Scheme C, and at 3 hop depth with two DIN Nodes for Scheme B. The DINCast in Schemes A and B is further away from the majority of receivers so the delay ratio increases, while the DINCast in Scheme C is close to the source so the delay ratio is kept low.

When the level (more than 3) of multicast tree increases, the optimal delay ratio increases for Schemes A and B. The reason is that as the level of multicast tree increases, the relatively fixed DINCast in Schemes A and B causes the delay advantage of DINCast to become less and less effective. Since the DINCast in Scheme C is always close to the source, the delay ratio maintains low.

When we compare the optimal delay ratio among the three schemes, the optimal delay ratio in Scheme B is the largest, while the optimal delay ratio in Scheme C is the smallest. In addition, the multicast delay in Scheme C is about half of the multicast delay in the share-tree multiscasts. The reason is that non-DIN Nodes in Scheme C receive the multicast message from the same level of DIN Nodes or their parents via trees with $n_1$ branches, and therefore the multicast delay is less than that in Scheme A with series links. In Scheme B, it uses the original tree to forward the multicast message without adding the links, and therefore the delay is close to the original shared-tree multicast and is increased when compared with Scheme A and Scheme C.

Although, we only carry out the simulations for $n_1=2$ and $n_1=3$, it is reasonable to predict that DINCast can find a good way to optimize the multicast tree with less multicast delay.

To discover the optimal placement of DINCast on an $L$-level shared-multicast tree with $n_i$ branches, we examine all possible DINCasts at Level $l$ with the number of DIN Nodes from 2 to $n_i^l$ ($l \in (0, L)$). Thus, the complexity to discover the optimal placement of DINCast is $O(n_i^{L-1})$.

IV. GENERALIZATION OF DINCAST

In the previous multicast delay analysis, we assume that the shared multicast tree is symmetric. In this section, we remove this constraint and extend DINCast to optimize any shared multicast tree. Given the asymmetric property of the multicast tree, we would need a different solution to determine the optimal placement of DINCast. We present here an experimental approach, coupled with simulations to accomplish this objective. For simplicity, our solution is based on number of tree hop count -- which has direct relationship with delay. More details of this scheme are presented in [13]. In this section, we present our more intensive simulation results.

We assume that all nodes in the shared multicast tree are the receivers. Ratio_max is the ratio between the maximum hop count using DINCast and the maximum hop count using shared-tree multicast, and Ratio_ave is the ratio between the average hop count using DINCast and the average hop count using shared-tree Multicast.

In our simulations, to generate the network topology, we use the GT-ITM (Georgia Tech Internetwork Topology Models) topology generator [14] which was developed by Georgia Tech and also adopted by NS-2 [15]. We use the graph generator of GT-ITM to generate different network topologies. The number of DIN Nodes is varied from 2 to 40. The probability that a direct link between each pair of DIN Nodes exists is varied from 0.1 to 1. We further
develop software to realize our DINCast scheme and calculate the max hop count and average hop count to receive multicast messages in DINCast scheme. We use the same network topologies as inputs for shared-tree multicast. Finally, we compute the Ratio_max and Ratio_ave.

We conduct simulations for different hop depths from 0 to 24. For brevity, only 0 hop depth and 12 hop depth are shown in Fig. 9. From Fig. 9, it is clear that Ratio_ave is less than Ratio_max in most scenarios.

When the number of DIN Nodes is small, the performance of DINCast is better than that of the shared-tree multicast since both Ratio_max and Ratio_ave are less than 1. However, as the number of DIN Nodes in the loop increases, the comparative performance gain of DINCast reduces because the delay in the loop increases. Thus, the number of DIN Nodes cannot be very large. In addition, the larger the hop depth, the better the performance of DINCast is being observed. The reason is that in DINCast, the messages are forwarded to all members by the loop instead of from the RP. When the hop depth is larger, the RP is further away from the majority of members and thus the multicast delay in shared-tree multicast increases. Thus, based on figures, depending on the number of DIN Nodes and the hop depth being chosen, DINCast can achieve better performance than shared-tree Multicast.

V. CONCLUSION

This paper presents DINCast to optimize application-level multicast with shared tree topologies. First, in the analysis of symmetric shared-tree multicast delay, we include propagation delay, transmission delay, and queuing delay using M/G/1 models.

We evaluate three DINCast schemes and results show that: (1) DINCast for all three schemes are effective in optimizing the multicast delay over the shared-tree multicast. (2) The optimal placement of DINCast in Scheme C changes with the levels of multicast tree, while the optimal placement in Scheme A and Scheme B is relatively fixed. (3) The optimal delay in Scheme C is, however, the best and is about half of the multicast delay in the original shared-tree multicast.

Next, for a generalized multicast tree, we carry out further simulations to optimize multicast delay. The experimental results show that for any multicast tree, based on the number of DIN Nodes and the hop depth being chosen, DINCast can achieve better performance than original shared-tree Multicast.

REFERENCES


Figure 9. Effect of the number of DIN Nodes & hop depth on maximum & average delays.