Future Generation Computer Systems 26 (2010) 291–303

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

DHT-based lightweight broadcast algorithms in large-scale computing infrastructures

Kun Huang [a], Dafang Zhang [b,*]

[a] School of Computer and Communication, Hunan University, Changsha, Hunan Province 410082, PR China
[b] School of Software, Hunan University, Changsha, Hunan Province 410082, PR China

ARTICLE INFO

Article history:
Received 20 June 2008
Received in revised form
12 August 2009
Accepted 29 August 2009
Available online 6 September 2009

Keywords:
Grid
Peer-to-Peer network
Distributed hash table
Broadcast
Load balancing

ABSTRACT

Scalable and efficient broadcast is essential to large-scale computing infrastructures such as PlanetLab and Grids. Previous broadcast algorithms exploit the greedy routing mechanisms of a Distributed Hash Table (DHT) to achieve the scalability. However, they suffer from load unbalancing and high overhead for constructing and maintaining a distributed broadcast tree (DBT). This paper presents DHT-based lightweight broadcast algorithms to overcome these limitations. Our algorithms leverage the topology and routing mechanism of Chord to select appropriate children of each node in a top-down approach. According to the node identifier distribution of Chord, we propose two broadcast algorithms over DHT. When nodes are uniformly distributed in the identifier space, a token-based broadcast algorithm is proposed, where each node selects the finger nodes as its children by a token value. When nodes are arbitrarily distributed in the identifier space, a partition-based broadcast algorithm is proposed, where each node partitions its identifier space into two subspaces and selects the agent nodes in the subspaces as its children. We show theoretically and experimentally that both token-based and partition-based algorithms can implicitly build a balanced DBT from the novel routing paths of DHT, where the branching factor is at most two and the tree height is $O(\log n)$ in a Chord of n nodes, without extra space for storing children and additional overhead for explicitly maintaining the parent–child membership.

1. Introduction

Large-scale computing infrastructures such as PlanetLab [1], E-science Grids [2], and Enterprise Desktop Grids [3] have become increasingly important for developing and deploying many emerging distributed applications such as content distribution networks, peer-to-peer systems, distributed games, and scientific computing. These computing infrastructures typically consist of large numbers of personal workstations and dedicated servers scattered around the world. For example, PlanetLab (2008) currently consists of 870 nodes at 460 sites [4], and the planet-scale Grid has 100,000 CPUs, mostly PCs and workstations [5]. As the size of the computing infrastructures continues to grow, it is very challenging for administrators to efficiently manage such large-scale dynamic distributed systems.

Distributed information management systems [6–9] have been extensively used in the large-scale computing infrastructures for a broad range of network services such as network monitor and management, service placement, resource management and task scheduling, and content distribution. Distributed broadcast is one of the fundamental primitives of distributed information management systems to disseminate information on a global scale. For instance, researchers replicate their programs on tens of thousands of nodes before launching a distributed application, and administrators distribute software releases and patch updates on all the federated nodes. Thus it is required to perform scalable and efficient content dissemination across large-scale computing infrastructures.

In recent years, Peer-to-Peer (P2P) based broadcast algorithms have been proposed to achieve scalability. There are two design principles for a P2P-based broadcast algorithm according to overlay structures: tree-based and mesh-based approaches. The tree-based approach constructs a tree overlay as a content delivering structure, such as ESM [10], NICE [11], Scribe [12], and Bayeux [13]. Since the single tree structure is vulnerable to failure of an interior node, multiple-tree structures such as SplitStream [14] and CoopNet [15] are proposed to improve the resilience, where each substream of content is delivered along one of multiple disjoint trees. On the other hand, the mesh-based approach, such as Bullet [16], FastReplica [17], Bullet' [18], BitTorrent [19], and CoBlitz [20], constructs a data-driven mesh overlay as a swarm system, where each node has a small set of neighbors to exchange data. Despite

* Corresponding author. Tel.: +86 0731 88821570.
E-mail addresses: huangkun@hunu.edu.cn (K. Huang), dfzhang@hunu.edu.cn (D. Zhang).

0167-739X/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2009.08.013

these arguments [21,22] against the two approaches, the tree-based approach is more suitable for large-scale computing infrastructures. This is because there is a large fraction of relatively stable dedicated nodes, and the tree structure has the simplicity and controlled overhead. Hence, we mainly focus on the tree-based approach to broadcasting in large-scale computing infrastructures.

Structured P2P systems have been recently proposed, such as Chord [23], CAN [24], Pastry [25], and Tapestry [26]. These structured systems use a Distributed Hash Table (DHT) as the routing substructure, with good characteristics of scalability, load balancing, and fault tolerance. Thus, DHT-based broadcast is a promising way in large-scale computing infrastructures. It is critical and crucial for a DHT-based broadcast algorithm to meet scalability, robustness, and load balancing. First, the algorithm should scale well to a large number of participating nodes with only a limited number of messages passing. Also it should have a low overhead of constructing and maintaining a distributed broadcast tree (DBT). Second, the algorithm should be robust to dynamics of node arrivals and departures. Finally, the algorithm should ensure good load balancing in the sense that the broadcast workload is evenly distributed among all participating nodes in the DBT, without any bottlenecks or hotspots of content delivery. Hence, load balancing is essential for scalability and robustness of a P2P-based broadcast algorithm.

Most DHT-based broadcast algorithms [6–8,12,14,15,27–32] have been recently proposed to support efficient broadcasting on large scale. These broadcast algorithms use the routing mechanisms of DHT to build up an overlay routing path between a source node and each destination node. Then a DBT rooted at the source node is constructed by merging these routing paths. A DHT-based broadcast algorithm has two approaches to constructing a DBT: top-down and bottom-up approaches. For example, the k-ary search based broadcast algorithm [31] adopts the top-down approach, where starting with a source node, each node selects its appropriate children, while the reverse-path forwarding based broadcast algorithm [12] adopts the bottom-up approach, where starting with all destination nodes, each node selects its appropriate parent to a source node. The bottom-up approach facilitates both aggregation and broadcast in the same DBT but consumes extra overhead on a node churn, while the top-down approach has low overhead due to not explicitly maintaining the child-parent membership. In this paper, our algorithms build a balanced DBT in a top-down approach.

However, existing DHT-based broadcast algorithms suffer from load unbalancing and high overhead. First, DHT is a greedy routing algorithm. In the DHT, each node always forwards a searched key to the closest preceding node in its finger table, whose identifier is closer to the key in the identifier space. The greedy essence of DHT results in that previous DHT-based broadcast algorithms, such as the k-ary search based broadcast algorithm [31] and the reverse-path forwarding based broadcast algorithm [12], construct a flat and unbalanced DBT. Such a DBT causes a node to be a performance bottleneck and a single failure on broadcasting. Second, these DHT-based broadcast algorithms have high overhead of constructing and maintaining a DBT with respect to a large number of participating nodes. For example, the reverse-path forwarding based broadcast algorithm [12] requires interior nodes to consume extra space for storing their children on reverse forwarding, which leads to additional maintenance overhead on node dynamics. Furthermore, these algorithms typically adopt the pushdown and anycast methods [11,13] to tackle the node overloading by adjusting the branching factors between nodes in a DBT. But, Bharambe et al. [41] indicated that these adjustment methods result in a significant number of non-DHT links that are present in the DBT but are not part of the routing links of DHT. The non-DHT links not only restrict the scalability of DBT, but also incur higher maintenance overhead in the DBT due to the dynamic nodes.

To address above issues, this paper proposes DHT-based lightweight broadcast algorithms in the large-scale computing infrastructures to achieve both scalability and load balancing. Our broadcast algorithms leverage a motivating observation of the topology and routing mechanisms of Chord. In a 2^m-node Chord, when a source node broadcasts a large file such as software patches to all other nodes, each node can appropriately select two of m finger nodes as its children, instead of selecting one or all finger nodes. The two-choice approach guarantees that each node has at most two children and all nodes are traversed. Thus, the basic idea behind our algorithms is that a balanced DBT is implicitly constructed from the novel routing paths of Chord, without explicit parent–child membership maintenance. According to the node identifier distribution of Chord, we propose two broadcast algorithms over DHT. (1) When nodes are uniformly distributed in the identifier space, a token-based broadcast algorithm is proposed, where each node selects the finger nodes as its children by a token value. (2) When nodes are arbitrarily distributed in the identifier space, a partition-based broadcast algorithm [33] is proposed, where each node partitions its identifier space into two subspaces and selects the agent nodes in the subspaces as its children. The partition-based algorithm generalizes the token-based algorithm, and is also suitable for the uniform distribution of node identifiers. We show theoretically experimental results show that both token-based and partition-based algorithms can construct and maintain a scalable and balanced DBT, where the branching factor is at most two and the tree height is $O(\log n)$ in a Chord of n nodes, and needs no extra space for storing children and no additional overhead for explicitly maintaining the parent–child membership.

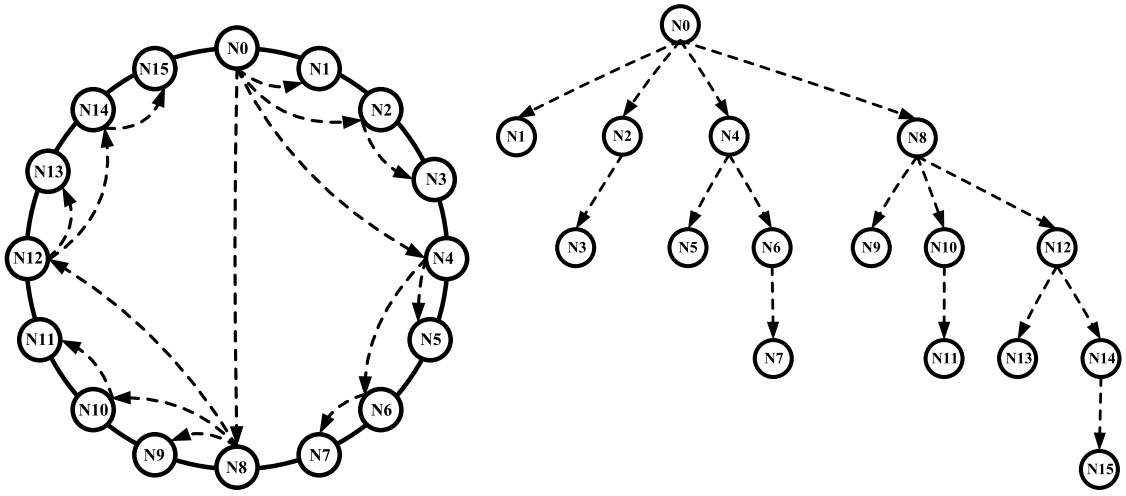
The rest of the paper is organized as follows. Section 2 introduces the background of Chord and typical DHT-based broadcast algorithms. In Section 3, we describe in detail our token-based and partition-based broadcast algorithms over DHT. Section 4 presents our experimental results. Related work is introduced in Sections 5 and 6 and concludes the paper.

2. Background

2.1. Chord overview

The Chord network is modeled as an undirected graph $G = (V, E)$, where the vertex set V contains n nodes and E is the set of overlay links between nodes. Chord utilizes a SHA-1 hash function to assign m-bit identifiers to both data objects and participating nodes. A node's identifier is produced by hashing the node's IP address, while an object's identifier is produced by hashing the object's key. Both objects and nodes have the same identifier space $[0, 2^m)$, and identifiers are ordered in an identifier circle modulo 2^m. According to the node identifiers, Chord organizes nodes as a ring topology in the circular space. An object's identifier k is assigned to the first node whose identifier is equal to or follows k in the identifier space. This node is called the successor node of the identifier k, denoted by $Succ(k)$. In Chord, $Pred(u)$ refers to the immediate predecessor of node u, while $Succ(u)$ refers to the immediate successor of node u. Besides its immediate predecessor and successor, each node u maintains a set of m finger nodes that are spaced exponentially in the identifier space. The ith finger node $Finger(u, i)$ of node u is the first node that succeeds u by at least 2^i in the identifier space, that is $Finger(u, i) = Succ((u+2^i) \bmod 2^m)$, where $0 \le i \le m-1$.

Chord adopts a greedy finger routing algorithm [23] to recursively (or iteratively) forward a query message with an object's identifier k to its successor node $Succ(k)$ that contains a pair (k, v), where v is the object's value. When node u wants to lookup an object's identifier k, it forwards a query message with the identifier k to its finger node $Finger(u, j)$, which is closest to the successor node



(a) Finger routing paths rooted at N_0. (b) Unbalanced DBT rooted at N_0.

Fig. 1. DBT construction using k-ary search in a 16-node Chord.

$Succ(k)$ in the circular identifier space, satisfying $Finger(u, j) \in (u, Succ(k)]$ and $Min\{Dist(Finger(u, j), k), 0 \leq j \leq m - 1\}$, where $Dist(u_1, u_2)$ is the numeric distance between two identifiers u_1 and u_2, that is $Dist(u_1, u_2) = (u_1 - u_2 + 2^m) \bmod 2^m$. And then, node $Finger(u, j)$ continues to forward the query message to the next node using the similar routing algorithm, to the successor node $Succ(k)$. Since the finger nodes of a node u are spaced exponentially in the identifier space, each hop in the finger routing paths covers at least half of the identifier space between the node u and the successor node $Succ(k)$. Therefore, the finger routing algorithm of Chord provides a scalable and efficient lookup service, with average routing path length of $O(\log n)$ and average node state space of $O(\log n)$ in a Chord of n nodes.

2.2. DHT-based broadcast overview

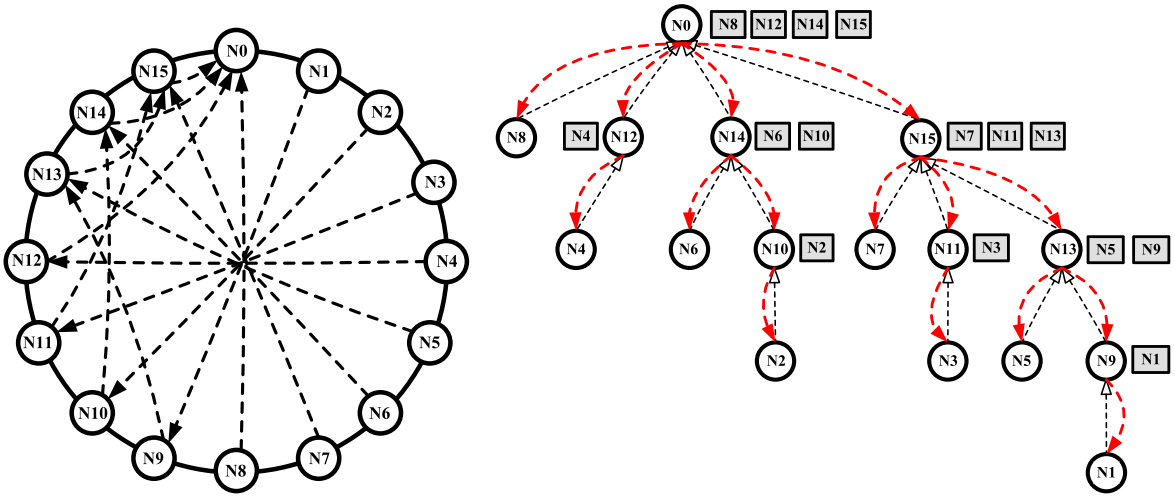
This section presents two classic DHT-based broadcast algorithms including the k-ary search based broadcast algorithm and reverse-path forwarding based broadcast algorithm, and indicates that the two algorithms have limitations of scalability and load balancing.

(1) El-Ansary et al. [31] proposed the k-ary search based broadcast algorithm, where each node selects all finger nodes in the limited forwarding space as its children in a top-down approach, and then a DBT rooted at the source node is constructed from the finger routing paths. Fig. 1 depicts an example of DBT construction using k-ary search in a 16-node Chord. As seen in Fig. 1(a), the source node N_0 selects the finger nodes N_8, N_4, N_2, and N_1 in the forwarding space $[N_0, N_0)$ as its children, and forwards new limited forwarding subspaces $[N_8, N_0)$, $[N_4, N_8)$, $[N_2, N_4)$, and $[N_1, N_2)$ to N_8, N_4, N_2, and N_1 respectively; other nodes continue to select their children in the forwarding space, and further broadcasts new limited forwarding subspaces, until no child is selected. Fig. 1(b) shows the corresponding DBT rooted at N_0 constructed from the finger routing paths in Fig. 1(a).

The k-ary search based broadcast algorithm has a low overhead of construction and maintenance, due to the finger routing algorithm and stabilization algorithm of Chord [23]. However, the algorithm suffers from the imbalance of node loads in the DBT. As seen in Fig. 1(b), node N_0 and N_8 have four and three branching factors respectively, while node N_2 and N_4 have one and two branching factors respectively. The imbalance of the DBT lies in the fact that the finger routing algorithm of Chord is greedy, which means that each node selects the shortest routing path to a destination node. For example, when node N_0 forwards a broadcast message to node N_2, the shortest routing path $N_0 \rightarrow N_2$ is selected by applying the greedy routing algorithm, while the longer path $N_0 \rightarrow N_1 \rightarrow N_2$ is not selected. Thus, this greedy routing algorithm incurs an unbalanced DBT constructed by the k-ary search based broadcast algorithm.

(2) Castro et al. [12] proposed a reverse-path forwarding based broadcast algorithm, where each destination node selects a finger routing path to a source node in a bottom-up approach, and then a DBT rooted at the source node is constructed by merging all the finger routing paths. When broadcasting a message, the algorithm forwards a broadcast message along the reverse finger routing paths in the DBT to all destination nodes. In the DBT, each parent node has a table for storing several children for reverse forwarding. Fig. 2 depicts an example of DBT construction using reverse-path forwarding in a 16-node Chord. As seen in Fig. 2(a), the finger routing paths from node N_1 and N_5 to node N_0 are $N_1 \rightarrow N_9 \rightarrow N_{13} \rightarrow N_{15} \rightarrow N_0$ and $N_5 \rightarrow N_{13} \rightarrow N_{15} \rightarrow N_0$ respectively, where both N_5 and N_9 select the same parent N_{13} that consumes extra space by storing two children for reverse forwarding. Fig. 2(b) shows the corresponding DBT rooted at N_0 constructed by merging overlapped nodes among all the finger routing paths in Fig. 2(a). In the DBT, each node except leaf nodes needs to store the children depicted by squares. When broadcasting a message, the source node N_0 forwards the broadcast message to N_8, N_{12}, N_{14}, and N_{15} along the reverse finger routing paths according to its children table.

However, the reverse-path forwarding based broadcast algorithm suffers from high overhead of construction and maintenance and the imbalance of node loads in the DBT. First, the algorithm exploits the bottom-up approach to constructing a DBT, where each node has a children table for reverse forwarding, incurring the high overhead of refreshing the children table. Second, each destination node exploits the greedy finger routing algorithm of Chord to select the shortest routing path to a source node, which results in an imbalance of loads among interior nodes in the DBT. As seen in Fig. 2(b), node N_0 and N_{15} have four and three branching factors respectively, while node N_{12} and N_{14} have one and two branching factors respectively. Furthermore, the pushdown and anycast methods [12,14] are proposed to adjust the branching factors between high-load nodes and low-load nodes in the DBT. But these adjustment methods in turn lead to higher maintenance overhead due to a large number of non-DHT in the DBT.



(a) Finger routing paths to N_0. (b) Unbalanced DBT rooted at N_0.

Fig. 2. DBT construction using revere-path forwarding in a 16-node Chord.

3. DHT-based lightweight broadcast algorithms

We propose DHT-based lightweight broadcast algorithms for scalability and load balancing. Our broadcast algorithms leverage the topology and routing mechanisms of Chord to implicitly construct a balanced DBT in a top-down approach. In this section, we present a token-based broadcast algorithm over DHT in the uniform identifier space, and a partition-based broadcast algorithm over DHT in the arbitrary identifier space.

3.1. Token-based broadcast algorithm

In Chord, the uniform identifier space is that nodes are uniformly distributed in the identifier space and the identifier ranges between two immediately adjacent nodes are the same. In the uniform identifier space, we propose a token-based broadcast algorithm over DHT. In the token-based algorithm, each node selects the finger nodes as its children by a token value, and then a balanced DBT is constructed in a top-down approach.

The balanced DBT construction process of the token-based algorithm is as follows. First, when receiving a broadcast message with a token value t, each node N_i selects the tth finger node $Finger(N_i, t)$ as its left child and the $(t+1)$th finger node $Finger(N_i, t+1)$ as its right child. Second, N_i forwards the broadcast message with the incremental token value $t+1$ to its children, until no child is selected. Note that when receiving a broadcast message with the token value t, a destination node N_d selects its left child N_l and right child N_r, and then checks to see whether its children's identifiers are not beyond the identifier of a source node N_s, satisfying $Dist(N_d, N_l) < Dist(N_d, N_s)$ and $Dist(N_d, N_r) < Dist(N_d, N_s)$. If not, then N_d does not forward the broadcast message to N_l or N_r. When initiating a broadcast message with a token value $t = 0$, a source node N_s must adjust the token value t to skip duplicated finger nodes in the finger table, by checking whether $Finger(N_s, t) \neq Finger(N_s, t+1)$ is satisfied or not. If not, then N_s increments the token value as $t+ = 1$ and continues to check the above condition.

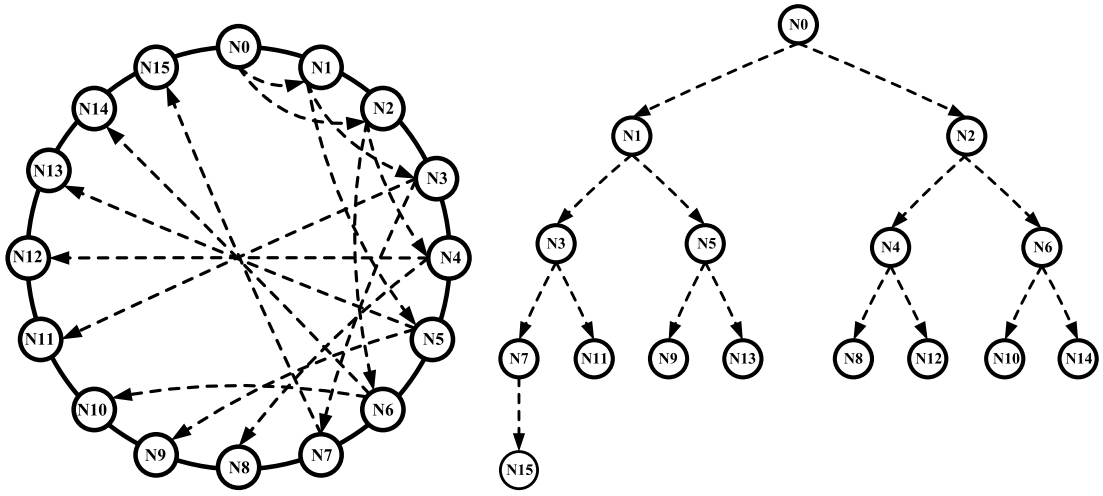
Fig. 3 illustrates the pseudo-codes of token-based broadcast algorithm. As seen in Fig. 3, *receive*$(P, R, Token)$ denotes that a node P receives a broadcast message with a token value *Token* and a source node R, and *send*$(P, R, Token)$ denotes that a broadcast message with a token value *Token* and a source node R is forwarded to a node P. Lines 3rd and 4th indicate that the algorithm exits if *Token* is equal to or more than m bits of node identifier; lines

```
1:  // receive (P, R, Token) represents that node P receives
      a broadcast message from source node R with Token.
      The initial value of Token is 0.
2:  // Skip redundant fingers by adjustting token value
3:  if Token >= m then
4:    exit (0);
5:  else
6:    for i = Token to m-2 do
7:      if Finger(P, i) ≠ Finger(P, i+1) then
8:        break;
9:      else
10:       Token = i+1;
11:     endif
12:   endfor
13: endif
14: // Select left child for broadcasting
15: if Finger(P, Token) in (P, R) then
16:   Left = Finger(P, Token);
17:   send (Left, R, Token+1);
18: endif
19: // Select right child for broadcasting
20: if Token < m then
21:   if Finger(P, Token+1) in (P, R) then
22:     Right = Finger(P, Token+1);
23:     send (Right, R, Token+1);
24:   endif
25: endif
```

Fig. 3. Pseudo-codes of token-based broadcast algorithm.

from 5th to 13th indicate that P skips duplicated finger nodes by adjusting *Token*; lines from 15th to 18th indicate that P selects the *Token*th finger node as its left child, whose identifier falls in the range (P, R); lines from 20th to 25th indicate that P selects the $(Token+1)$th finger node as its right child, whose identifier falls in the same range (P, R); lines 17th and 23rd indicate that P forwards the broadcast message with $Token+1$ and R to its left and right children respectively.

Fig. 4 depicts a balanced DBT construction example using a token-based broadcast algorithm in a 16-node Chord with a uniform identifier space. In Fig. 4(a), the source node N_0 initiates a broadcast message with a token value $t = 0$, and does not adjust the token value due to $Finger(N_0, 0) \neq Finger(N_0, 1)$; N_0 selects $N_1 = Finger(N_0, 0)$ and $N_2 = Finger(N_0, 1)$ as its left and right children respectively, and then forwards the broadcast message with the incremental token value $t = 1$ to N_1 and N_2; N_1 and N_2 continue to select their children and increment the token



(a) Finger routing paths rooted at N_0. (b) Balanced DBT rooted at N_0.

Fig. 4. Balanced DBT construction using token in a 16-node Chord.

value, and further forward the broadcast message, until no child is selected. When receiving a broadcast message with the token value $k = 3$, nodes from N_8 to N_{14} have no children and do not forward the broadcast message, because the identifiers of their children are more than the identifier of the source node N_0. Fig. 4(b) illustrates the corresponding balanced DBT rooted at N_0 constructed from the finger routing paths in Fig. 4(a). Fig. 4 shows that the token-based broadcast algorithm constructs a balanced DBT, where the branching factors are at most two, and the tree height is $\lceil \log_2 n \rceil = 4$, in which $n = 16$ is the number of nodes in Chord.

3.2. Analysis of token-based algorithm

In essence, the token-based broadcast algorithm is that each node selects appropriate children using a token value to alternatively cover all destination nodes. Unlike the k-ary search based broadcast algorithm that selects the children with the exponential decay of the interval distance, the token-based broadcast algorithm selects the children with the exponential growth of the interval distance. Hence, the token-based algorithm leverages the novel child selection strategy to guarantee that a balanced DBT is constructed and duplicated messages between nodes are eliminated on broadcasting.

We assume that a node i_1 selects the finger nodes $i_1 + 2^{t_1}$ and $i_1 + 2^{t_1+1}$ as the left and right children respectively, and a node i_2 selects the finger nodes $i_2 + 2^{t_2}$ and $i_2 + 2^{t_2+1}$ as the left and right children respectively. If $i_2 > i_1$ and $t_2 \geq t_1$, then $i_2 + 2^{t_2} > i_1 + 2^{t_1}$ and $i_2 + 2^{t_2+1} > i_1 + 2^{t_1+1}$, which denotes that the identifiers of $i'_2 s$ left and right children are larger than those of $i'_1 s$ left and right children respectively. If $i_2 > i_1$ and $t_2 > t_1$, then $t_2 = t_1 + \Delta$ and $\Delta \geq 1$, so that $i_2 + 2^{t_2} = i_2 + 2^{t_1+\Delta} > i_1 + 2^{t_1+1}$, which denotes that the identifier of $i'_2 s$ left child is larger than that of $i'_1 s$ right child. If $i_2 > i_1$ and $t_2 = t_1$, due to $i_2 = i_1 + 2^{t'}$ and $t' < t_1$, then $i_1 + 2^{t_1+1} = i_1 + 2^{t_1} + 2^{t_1} > i_2 + 2^{t_2} = i_1 + 2^{t_2} + 2^{t'}$, which denotes that the identifier of $i'_2 s$ left child is smaller than that of $i'_1 s$ right child. Therefore, we prove that different nodes do not select the same finger nodes as their children, so that the token-based algorithm alternatively covers all destination nodes, without any overlap.

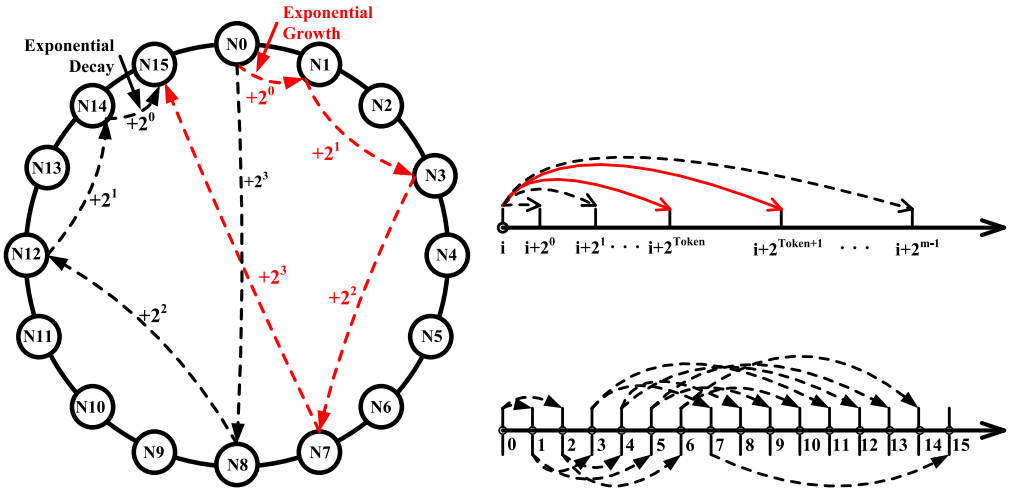
Fig. 5 illustrates the properties of a token-based broadcast algorithm in a 16-node Chord. Fig. 5(a) depicts the case where a source node N_0 forwards a broadcast message to a destination node N_{15}. The token-based algorithm selects the children with the exponential growth of the interval distance such as 2^0, 2^1, 2^2, and 2^3, whereas the k-ary search based algorithm selects the children with exponential decay such as 2^3, 2^2, 2^1, and 2^0. As seen in Fig. 5(b), when a source node N_0 forwards a broadcast message to all destination nodes, each node i selects the $Token$th finger node as its left child and the $(Token + 1)$th finger node as its right child respectively. The interval distance between parent and children in such circular space is exponentially growing and all destination nodes $N_1, N_2 \cdots N_{15}$ are alternately covered to construct a balanced DBT root at N_0.

Also the DBT constructed by the token-based algorithm is scalable, where the tree height is $O(\log n)$ in a n-node Chord. We assume that a source node is u and a destination node is v, satisfying $u \neq v$. Since Chord has the uniform identifier space, u adjusts the token value $Token = t$ to skip duplicated finger nodes in the finger table, and then increments the token value, until $Token = m$. Thus, the finger routing path from u to v is $u + 2^t \rightarrow u + 2^t + 2^{t+1} \rightarrow \cdots \rightarrow v$, with $h = m - t$ hops. We compute $Max(Dist(u, v)) = 2^m - 2^t$, in which 2^t is the average identifier range between two immediately adjacent nodes in Chord. Due to $2^t = 2^m/n$, we deduce $h = m - t = \log_2 n$. Hence, we prove that the tree height is $O(\log n)$ so that the DBT is scalable.

Moreover, the token-based algorithm needs no additional overhead for constructing and maintaining such a balanced DBT. First, since children are selected by a token value in a top-down approach, each parent needs no extra space for storing its children, so that the balanced DBT is implicitly constructed from the finger routing paths of Chord. Second, on node arrivals and departures, Chord adopts the finger stabilization algorithm [23] to periodically update the finger tables of nodes. But the token-based algorithm needs no extra cost to repair the parent–child membership in the balanced DBT, which significantly reduces the maintenance overhead of DBT. Therefore, the DBT constructed by the token-based algorithm has no explicit extra overhead of construction and maintenance.

3.3. Partition-based broadcast algorithm

In Chord, the arbitrary identifier space is that nodes are arbitrarily distributed in the identifier space of Chord and the identifier ranges between two immediately adjacent nodes are not the same. Especially, when a node joins in and leaves from Chord, the uniform identifier space becomes the non-uniform identifier space. In the arbitrary identifier space, we propose a partition-based broadcast algorithm over DHT. In the partition-based algorithm, each node



(a) Finger routing paths rooted at N_0.

(b) Alternately covering all destination nodes.

Fig. 5. Properties of token-based broadcast algorithm in a 16-node Chord.

hierarchically partitions its identifier space into two subspaces, and selects the agent nodes in the subspaces as its children, and then a balanced DBT is constructed in a top-down approach.

The balanced DBT construction process of the partition-based algorithm is as follows. First, when receiving a broadcast message with a limitation value l of identifier space, each node N_i partitions its limited identifier space (N_i, l) into two subspaces $(N_i, Finger(N_i, j))$ and $[Finger(N_i, j), l)$, where $Finger(N_i, j)$ is the closest finger node to the limitation value l in the circular identifier space. Second, N_i selects the agent node $Finger(N_i, k)$ in $(N_i, Finger(N_i, j))$ as its left child, where $Finger(N_i, k)$ is the farthest finger node from $Finger(N_i, j)$ in the circular identifier space, and selects the agent node $Finger(N_i, j)$ in $[Finger(N_i, j), l)$ as its right child. Finally, N_i forwards the broadcast message with the limitation value $Finger(N_i, j)$ to $Finger(N_i, k)$ and the broadcast message with the limitation value l to $Finger(N_i, j)$, until no child is selected. Note that when initiating a broadcast message with a limitation value l, the source node N_s sets the limitation value $l = N_s$, which denotes that its identifier space (N_s, N_s) is the whole identifier space. When a destination node N_d selects its left child N_l and right child N_r, if $N_l = N_r$, then N_d only forwards the broadcast message to N_r.

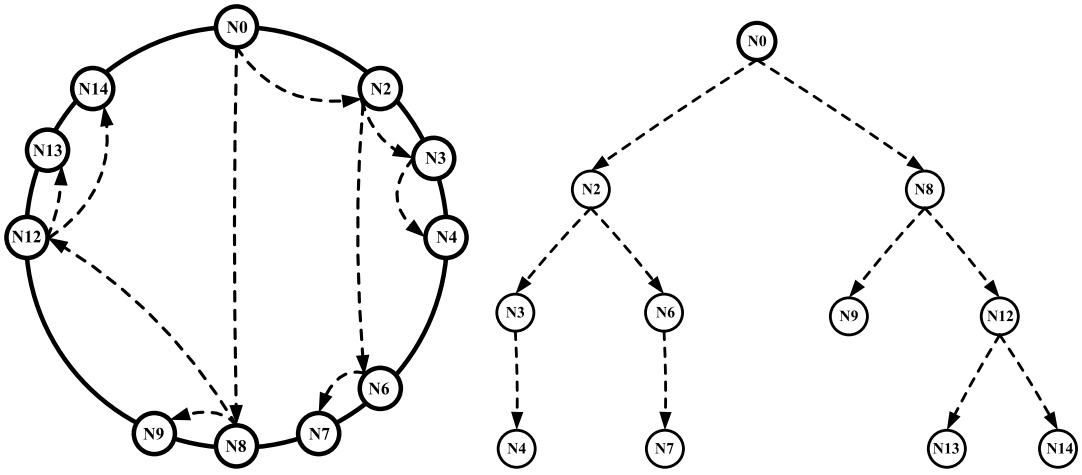
Fig. 6 illustrates the pseudo-codes of partition-based broadcast algorithm. As seen in Fig. 6, *receive*$(P, R, Limit)$ denotes that a node P receives a broadcast message with a limitation value *Limit* of identifier space and a source node R, and *send*$(P, R, Limit)$ denotes that a broadcast message with a limitation value *Limit* and a source node R is forwarded to a node P. Lines from 3rd to 9th indicate that P selects its right child *Right* who is the closest finger node to *Limit* in the circular identifier space, and partitions its identifier space $(P, Limit)$ into two subspaces $(P, Right)$ and $[Right, Limit)$; lines 10th and 11th indicate that the algorithm exits if there is no right child; lines from 14th and 20th indicate that P selects its left child *Left* who is the farthest finger node from *Right* in $(P, Right)$; lines from 22nd and 25th indicate that P forwards the broadcast message with the limitation value *NewLimit* to its left child; line 26th indicates that node P forwards the broadcast message with the limitation value *Limit* to its right child.

Fig. 7 depicts a balanced DBT construction example using a partition-based broadcast algorithm in an 11-node Chord with the non-uniform identifier space. In Fig. 7(a), the source node N_0 sets the limitation value $l = N_0$, and partitions the whole identifier space (N_0, N_0) into two subspaces (N_0, N_8) and $[N_8, N_0)$, where N_8 is the closest finger node to the limitation value $l = N_0$ in the

```
1:  // receive (P, R, Limit) represents that node P receives
      abroadcast message from source node R with subspace Limit.
      The initial value of Limit is R, when starting to broadcast.
2:  // Select right child for broadcasting
3:  for j = m-1 to 0 do
4:    if Finger(P, j) in (P, Limit) then
5:      Right = Finger(P, j);
6:    else
7:      Right = Null;
8:    endif
9:  endfor
10: if Right = Null then
11:   exit(0);
12: endif
13: // Select left child for broadcasting
14: for i = 0 to m-1 do
15:   if Finger(P, i) in (P, Right) then
16:     Left = Finger(P, i);
17:   else
18:     Left = Null;
19:   endif
20: endfor
21: // Eliminate redundant child and set new limit for subspace
22: if (Left ≠ Null) and (Left ≠ Right) then
23:   NewLimit = Right;
24:   send (Left, R, NewLimit);
25: endif
26: send (Right, R, Limit);
```

Fig. 6. Pseduo-codes of partition-based broadcast algorithm.

circular identifier space; N_0 selects the agent node N_2 in (N_0, N_8) as its left child and the agent node N_8 in $[N_8, N_0)$ as its right node, and then forwards the broadcast message with the limitation value N_8 to N_2 and the broadcast message with the limitation value N_0 to N_8; similarly, when receiving the broadcast message, N_2 and N_8 further partitions their identifier space into two subspace respectively, and then continue to forwards the broadcast message with the new limitation value to their children, until no child is selected. Fig. 7(b) illustrates the corresponding balanced DBT rooted at N_0 constructed from the finger routing paths in Fig. 7(a). Fig. 7 shows that the partition-based broadcast algorithm constructs a balanced DBT, where the branching factors are at most two, and the tree height is $3 < \lceil \log_2 n \rceil = 4$, in which $n = 11$ is the number of nodes in Chord.



(a) Finger routing paths rooted at N_0. (b) Balanced DBT rooted at N_0.

Fig. 7. Balanced DBT construction using partition in an 11-node Chord.

3.4. Analysis of partition-based algorithm

In essence, the partition-based broadcast algorithm is that each node selects its children by partitioning its limited identifier space to construct a binary partition tree. Nevertheless, in the k-ary search based broadcast algorithm, each node partitions its limited identifier space into k subspaces to construct a k-ary partition tree. Hence, the partition-based algorithm leverages the novel binary partition to guarantee that a balanced DBT is constructed and all destination nodes are hierarchically covered.

Fig. 8 illustrates the properties of partition-based broadcast algorithm in an 11-node Chord. As seen in Fig. 8(a), the source nod N_0 partitions the whole identifier space into two subspaces, each of which is further partitioned into two subspace, until the identifier subspace contains only one node. So the hierarchical binary partition constructs a balanced DBT rooted at N_0. As seen in Fig. 8(b), when the source node N_0 forwards a broadcast message to all destination nodes, each node i partitions its identifier space into two subspace $(i, i + 2^j)$ and $[i + 2^j, Limit)$, and selects $i + 2^0$ as its left child and $i + 2^j$ as its right child; i forwards the broadcast message to its children, until all destination nodes $N_2, N_3, \ldots, N_{14}$ are covered in a hierarchical approach.

The DBT constructed by the partition-based algorithm is scalable, where the tree height is $O(\log n)$ in a n-node Chord. We assume that Chord has the identifier space $[0, 2^m)$ and the ith binary partition divides the whole identifier space into 2^i subspaces, each of which has the size of $2^m/2^i$, where $1 \leq i \leq h$ and h is the tree height of DBT. After the hth binary partition, each subspace contains only one node, which means that it is $\lceil n/2^h \rceil = 1$. We deduce $h = \lceil \log_2 n \rceil$. Hence, we prove that the tree height is $O(\log n)$ so that the DBT is scalable.

Especially, when Chord has a uniform identifier space, the partition-based broadcast algorithm is also applied to construct a scalable and balanced DBT. Fig. 9 depicts a balanced DBT construction example using partition in a 16-node Chord with a uniform identifier space. As seen in Fig. 9(a), the source node N_0 partitions the whole identifier space (N_0, N_0) into two subspaces (N_0, N_8) and $[N_8, N_0)$, and selects N_1 in (N_0, N_8) as its left child and N_8 in $[N_8, N_0)$ as its right child, and then forwards the broadcast message with the limitation value N_8 to N_1 and the broadcast message with the limitation value N_0 to N_8; similarly, other nodes continue to partition their identifier space into two subspaces, and further forward the broadcast message. Fig. 9(b) shows the corresponding balanced DBT rooted at N_0 constructed from the finger routing paths in Fig. 9(a). Fig. 9 shows that in the uniform identifier space, the partition-based broadcast algorithm also constructs a scalable and balanced DBT, where the branching factors are at most two and the tree height is $\lceil \log_2 n \rceil = 4$, in which $n = 16$ is the number of nodes in Chord. In other words, the partition-based algorithm generalizes the token-based algorithms in both the uniform and non-uniform identifier spaces of Chord.

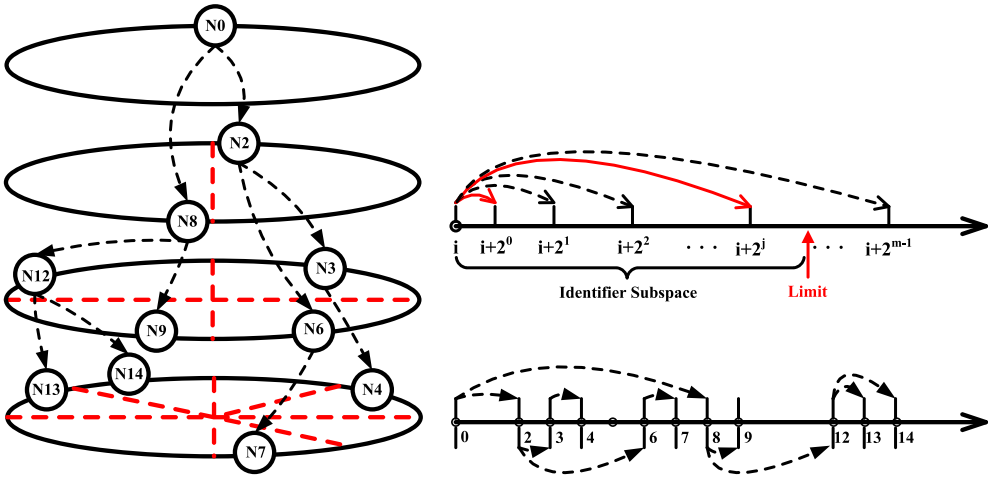
Moreover, the partition-based algorithm needs no additional overhead for constructing and maintaining a balanced DBT. First, since children are selected by partitioning the identifier space into two parts in a top-down approach, each parent needs no extra space for storing its children, so that the balanced DBT is implicitly constructed from the finger routing paths of Chord. Second, due to the finger stabilization algorithm [23], the partition-based algorithm needs no overhead for explicitly maintaining the parent–child membership in the balanced DBT. Therefore, the DBT constructed by the partition-based algorithm has no explicit extra overhead of construction and maintenance.

4. Experimental evaluation

This section presents simulation experiments to validate our DHT-based lightweight broadcast algorithms. We designed and implemented a discrete event-driven Chord network simulator based on p2psim-0.3 [34]. The Chord simulator can simulate not only the finger routing and stabilization algorithms of Chord, but also the dynamics of node arrivals and departures. We implemented four DHT-based broadcast algorithms, including the k-ary search based algorithm, reverse-path forwarding based algorithm, token-based algorithm, and partition-based algorithm.

In the simulation experiments, the physical network consists of 2500 nodes with real measured Internet latencies from the Meridian data set [35]. In Chord, we choose 2^m overlay nodes with a m-bit identifier, each of which is randomly hosted on a unique physical node; each overlay node has m finger nodes and two successor nodes, and periodically refreshes its finger table and successor list during every 1000 ms; when a constant number of overlay nodes leaves the system, the same number of overlay nodes joins in the system again during every 1000 ms; 100 messages are broadcast across all overlay nodes during every 100 ms; the simulation process runs about 10 h. In our experiments, from $2^4 = 16$ to 2^{11} nodes are simulated in the uniform identifier space, while from 200 to 2000 nodes are simulated in the non-uniform identifier space.

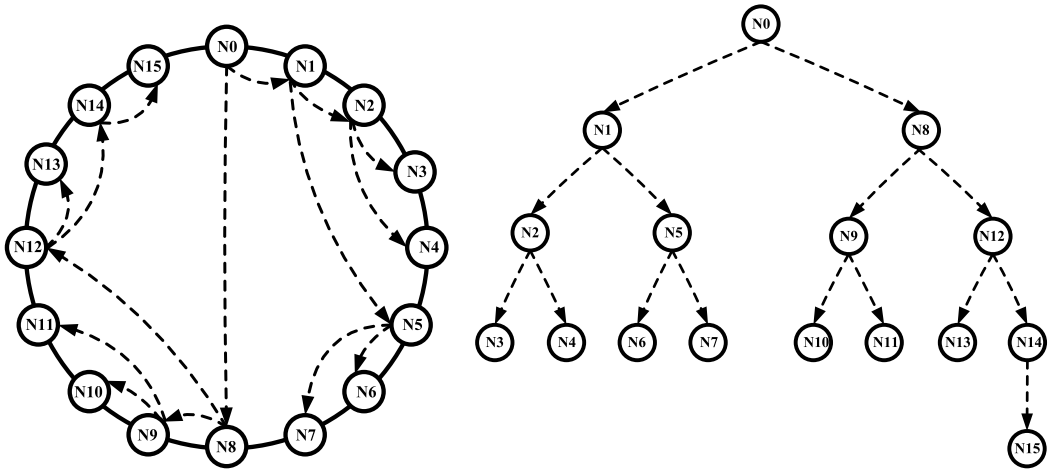
The performance metrics of DHT-based broadcast algorithm include the branching factor, path length, message overhead, and message imbalance factor. The branching factor refers to the



(a) Hierarchical binary partition of the node identifier space from N_0.

(b) Alternately covering all the destination nodes to construct balanced DBT rooted at N_0.

Fig. 8. Properties of partition-based broadcast algorithm in an 11-node Chord.



(a) Finger routing paths rooted at N_0.

(b) Balanced DBT rooted at N_0.

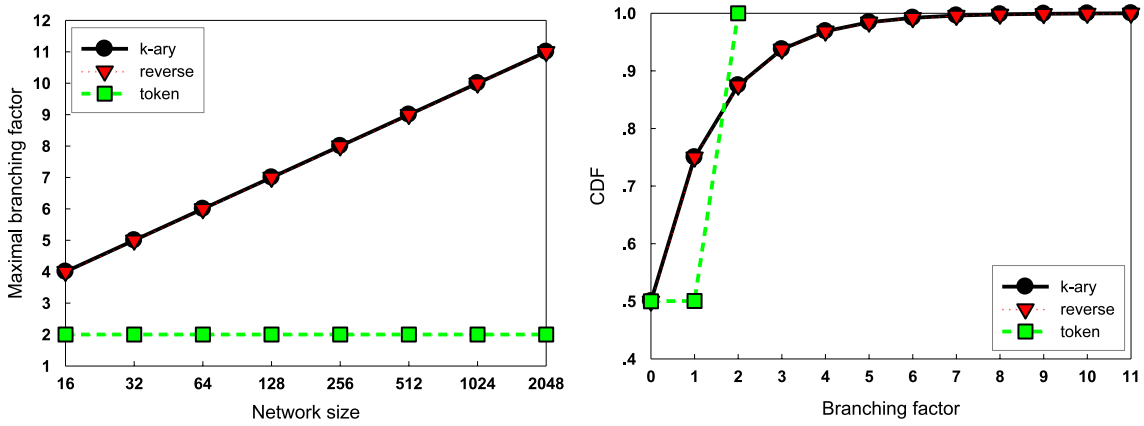
Fig. 9. Balanced DBT construction using partition in a 16-node Chord.

number of a node's children in a DBT, also called out-degrees of a node. The path length refers to hops of a finger routing path from a source node to a destination node, including the maximal and average path lengths. The message overhead refers to the number of messages for constructing and maintaining a DBT. The message imbalance factor refers to the ratio between the maximal number of messages and average number of messages on each node in a DBT.

4.1. Uniform identifier space

Fig. 10 depicts the comparison of a branching factor in the uniform identifier space. Fig. 10(a) depicts the maximal branching factor with various network sizes from 16 to 2048, while Fig. 10(b) depicts the cumulative distribution function of a branching factor in a 2048-node Chord. As seen in Fig. 10(a), there is the same maximal branching factor of $\log_2 n$ for both the k-ary search based and reverse-path forwarding based algorithms, whereas there is a constant maximal branching factor of 2 for the token-based algorithm. For example, in a 2048-node Chord, both the k-ary search based and reverse-path forwarding based algorithms have the same maximal branching factor of 11, while the token-based algorithm has the maximal branching factor of 2. As seen in Fig. 10(b), both the k-ary search based and reverse-path forwarding based algorithms construct a skewed DBT, while the token-based algorithm constructs a balanced DBT. In both the k-ary search based and reverse-path forwarding based algorithms, about 88% of nodes in the DBT have the branching factors of equal to or less than 2, and the remaining 12% have the branching factors from 3 to 11. In the token-based algorithm, about 50% of nodes in the DBT have the branching factors of 2, and the remaining 50% are almost leaf nodes without any branching factor. Fig. 10 demonstrates that the token-based broadcast algorithm constructs a balanced DBT, where the branching factors are at most two.

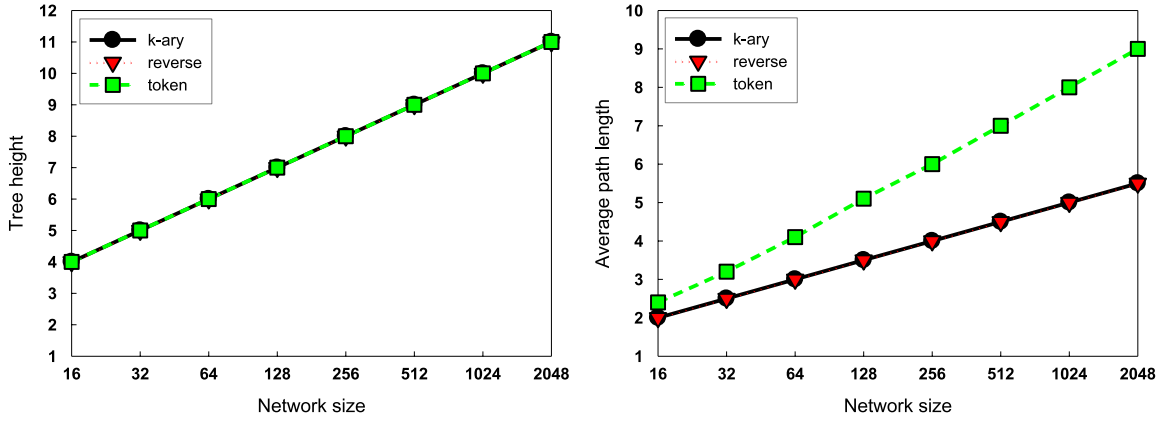
Fig. 11 depicts the comparison of path length in the uniform identifier space. Fig. 11(a) and Fig. 11(b) depict the maximal path length and average path length with various network sizes from 16 to 2048 respectively. As seen in Fig. 11(a), the token-based algorithm has the same maximal path length of $\log_2 n$ with both the k-ary search based and reverse-path forwarding based algorithms. As seen in Fig. 11(b), compared with both the k-ary search based and reverse-path forwarding based algorithms, the token-based algorithm increases 20%–40% of average path length. The root cause lies in the fact that both the k-ary search based and reverse-path forwarding based algorithms construct a skewed flat DBT from the shortest finger routing paths, whereas the token-based algorithm constructs a balanced narrow DBT from the novel



(a) Maximal branching factor with various network sizes from 16 to 2048.

(b) Cumulative distribution function of branching factor in a 2048-node Chord.

Fig. 10. Comparison of branching factor in the uniform identifer space.



(a) Maximal path length with various network sizes from 16 to 2048.

(b) Average path length with various network sizes from 16 to 2048.

Fig. 11. Comparison of path length in the uniform identifer space.

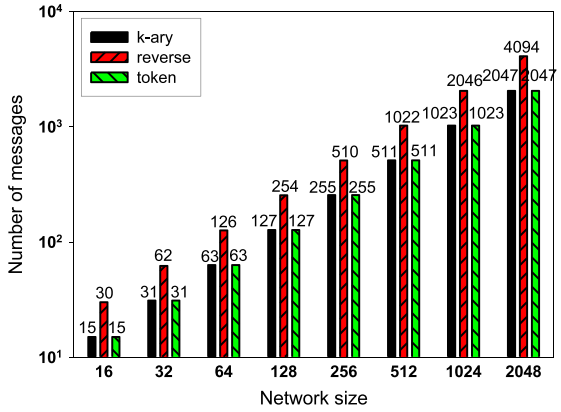


Fig. 12. Comparison of message overhead in the uniform identifier space.

finger routing paths. Despite all that, the token-based algorithm construct the balanced DBT to achieve the same max broadcast latency of $\log_2 n$ with both the k-ary search based and reverse-path forwarding based algorithms. Fig. 11 demonstrates that the token-based broadcast algorithm construct a scalable DBT, where the tree height is $O(\log n)$ in which n is the number of nodes in Chord.

Fig. 12 depicts a comparison of the message overhead in the uniform identifier space. As seen in Fig. 12, both the k-ary search based and token-based algorithms have the same messages of $n-1$, whereas the reverse-path forwarding based algorithm has $2(n-1)$ messages. The root cause lies in the fact that the reverse-path forwarding based algorithm constructs a DBT in a bottom-up approach and a broadcast message is forwarded along the reverse finger routing paths, while both the k-ary search based and our token-based algorithms construct a DBT in a top-down approach and a broadcast message is directly forwarded along the finger routing paths. When a node arrives and departs, the finger stabilization algorithm of Chord [23] can consume the average messages of $O(\log^2 n)$ for maintaining the finger tables of relative nodes. Hence, the reverse-path forwarding based algorithm needs to consume extra average messages of $O(\log^2 n)$ to maintain the parent–child membership in the DBT. Fig. 12 shows that the token-based broadcast algorithm constructs and maintains a balanced DBT with low overhead, which needs no additional space for storing the children and no overhead for explicitly maintaining the parent–child membership in the DBT.

Fig. 13 depicts the comparison of message imbalance factor in the uniform identifier space. As seen in Fig. 13, both the k-ary search based and reverse-path forwarding based algorithms have the same message imbalance factor as a function of network sizes, while the token-based algorithm has the constant message imbalance factor independent of network sizes. For example, in a 2048-node Chord, the message imbalance factor is about 5.5 for both the k-ary search based and reverse-path forwarding based algorithms, while the message imbalance factor is kept about 1.0 for the token-based algorithm. Fig. 13 shows that compared with both the k-ary search based and reverse-path forwarding based algorithms, the token-based broadcast algorithm reduces 50%–82%

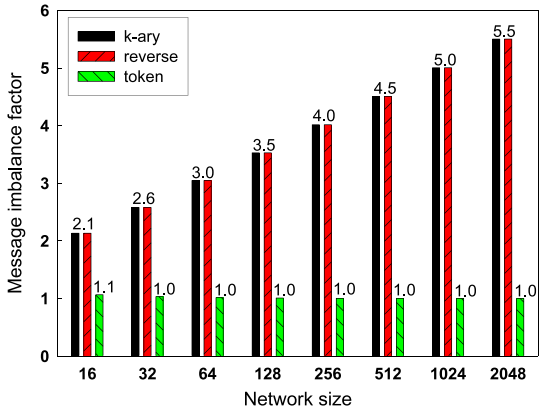


Fig. 13. Comparison of message imbalance factor in the uniform identifier space.

of the message imbalance factor, which further demonstrates that a DBT constructed by the token-based algorithm is balanced.

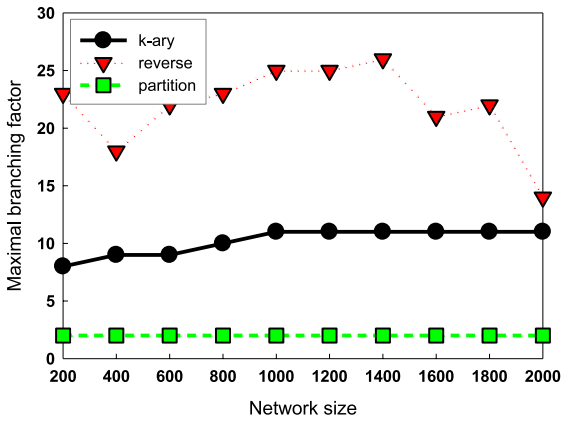
4.2. Non-uniform identifier space

Fig. 14 depicts the comparison of a branching factor in the non-uniform identifier space. Fig. 14(a) depicts the maximal branching factor with various network sizes from 200 to 2000, and Fig. 14(b) depicts the cumulative distribution function of branching factor in a 2000-node Chord. Fig. 14(a) shows that the partition-based algorithm keeps the constant maximal branching factor, while the maximal branching factor of both the k-ary search based and reverse-path forwarding based algorithms fluctuates with various network sizes. The partition-based algorithm has a constant maximal branching factor of 2, whereas the k-ary search based algorithm has a maximal branching factor of about 11 and the reverse-path forwarding based algorithm has a maximal branching factor varying from 14 to 26. Fig. 14(b) shows that both the k-ary search based and reverse-path forwarding based algorithms construct a skewed DBT, while the partition-based algorithm constructs a balanced DBT. In both the k-ary search based and reverse-path forwarding based algorithms, about 88% of nodes in the DBT have the branching factor of equal to or less than 2, and the remaining 12% have the branching factor of from 3 to 14. In the partition-based algorithm, about 50% of nodes in the DBT have the branching factor of 2, and the remaining 50% are almost leaf nodes without any branching factor. Fig. 14 demonstrates that the partition-based broadcast algorithm constructs a balanced DBT, where the branching factors are at most two.

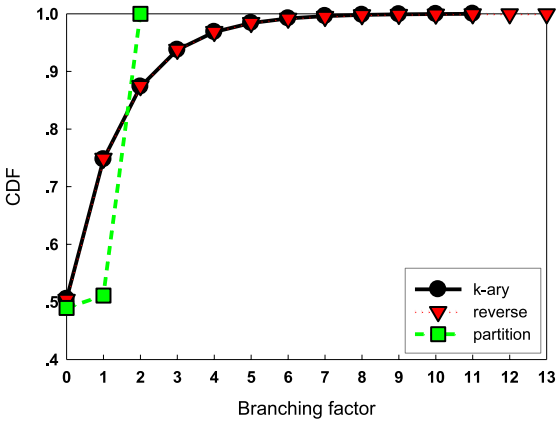
Fig. 15 depicts a comparison of path length in the non-uniform identifier space. Fig. 15(a) and (b) depict the maximal and average path lengths with various network size from 200 to 2000 respectively. As seen in Fig. 15(a), the maximal path length of the partition-based algorithm is larger than that of both the k-ary search based and reverse-path forwarding based algorithms. When Chord has no more than 1400 nodes, the maximal path length of the partition-based algorithm is kept at 10, while the maximal path length of both the k-ary search based and reverse-path forwarding based algorithms varies with different network sizes, but not beyond 10. When Chord has up to 2000 nodes, the three algorithms have the same maximal path length of $O(\log n)$. As seen in Fig. 15(b), compared with both the k-ary search based and reverse-path forwarding based algorithms, the partition-based algorithm separately increases 55%–68% and 58%–100% of the average path length. This is because that both the k-ary search based and reverse-path forwarding based algorithms construct a skewed flat DBT, while the partition-based algorithm constructs a balanced narrow DBT. Despite all that, the balanced DBT constructed by the partition-based algorithm has the same max broadcast latency of $O(\log n)$ with that of both the k-ary search based and reverse-path forwarding based algorithms. Fig. 15 demonstrates that the partition-based broadcast algorithm constructs a scalable DBT, where the tree height is $O(\log n)$ in which n is the number of nodes in Chord.

Fig. 16 depicts a comparison of message overhead in the non-uniform identifier space. As seen in Fig. 16, both the k-ary search based and partition-based algorithms have the same messages of $n - 1$, while the reverse-path forwarding based algorithms have a message overhead of $2(n - 1)$. The reason is that both the k-ary search based and partition-based algorithms construct a DBT in a top-down approach, which not only reduces the number of broadcast messages but also eliminates extra space for storing the children in the DBT. However, the reverse-path forwarding based algorithm needs additional messages of $O(\log^2 n)$ to maintain the parent–child membership in the DBT, due to the finger stabilization protocol [23]. Fig. 16 demonstrates that the partition-based broadcast algorithm has low construction and maintenance overhead of a balanced DBT, with no extra space of children and no overhead for explicitly maintaining the parent–child membership.

Fig. 17 depicts a comparison of the message imbalance factor in the non-uniform identifier space. As seen in Fig. 17, compared with both the k-ary search based and reverse-path forwarding based algorithms, the partition-based algorithm has a lower constant message imbalance factor. For example, in a 2000-node Chord, the partition-based algorithm has the message imbalance factor of 1.0,

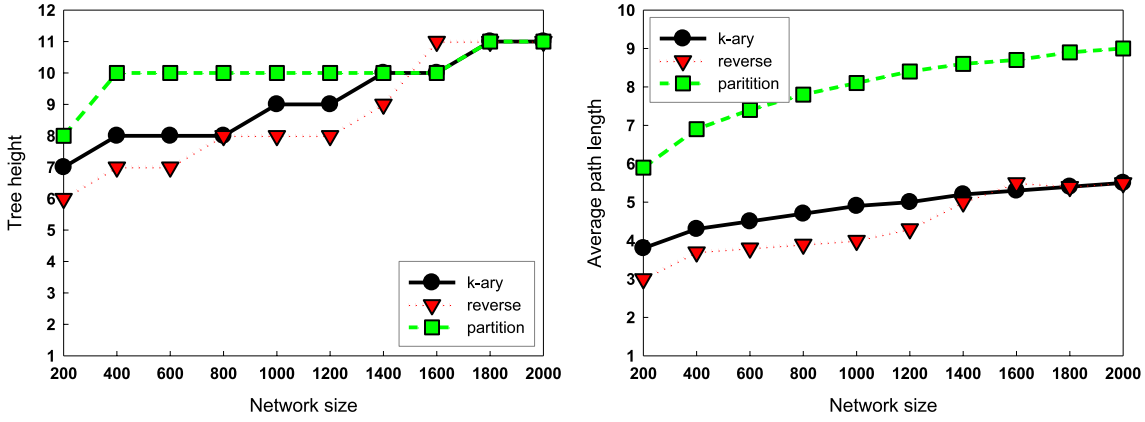


(a) Maximal branching factor with various network sizes from 200 to 2000.



(b) Cumulative distribution function of branching factor in a 2000-node Chord.

Fig. 14. Comparison of branching factor in the non-uniform identifer space.



(a) Maximal path length with various network sizes from 200 to 2000.

(b) Average path length with various network sizes from 200 to 2000.

Fig. 15. Comparison of path length in the non-uniform identifer space.

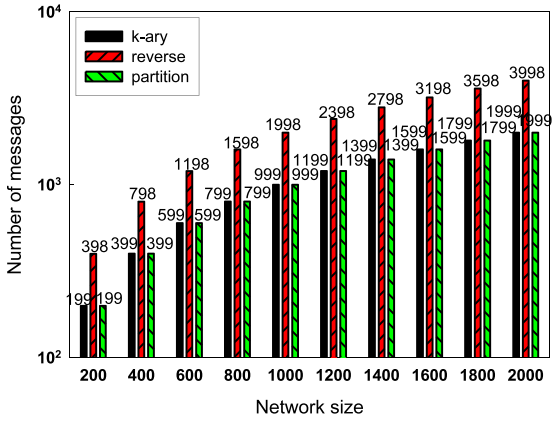


Fig. 16. Comparison of message overhead in the non-uniform identifier space.

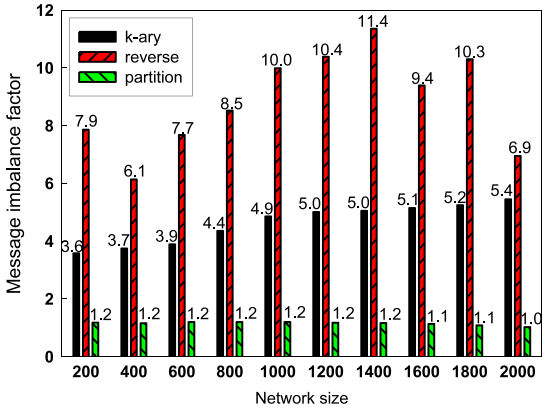


Fig. 17. Comparison of message imbalance factor in the non-uniform identifier space.

while both the k-ary search based and reverse-path forwarding based algorithms have a message imbalance factor of 5.4 and 6.9 respectively. Fig. 17 shows that compared with both the k-ary search based and reverse-path forwarding based algorithms, the partition-based broadcast algorithm separately reduces 67%–81% and 81%–90% of the message imbalance factor, which further demonstrates that the DBT constructed by the partition-based algorithm is balanced.

4.3. Discussion

In this section, we discuss the comparisons between our token-based and partition-based broadcast algorithms. The token-based algorithm is only applied in the uniform identifier space, while the partition-based algorithm is applied in both the uniform and non-uniform identifier spaces. We mainly compare the performance between the token-based and partition-based algorithms in the uniform identifier space of Chord.

Table 1 gives comparisons between token-based and partition-based algorithms in the uniform identifier space. The partition-based algorithm has the same maximal branching factor, maximal path length, message overhead, and message imbalance factor with the token-based algorithm. The partition-based algorithm has a constant maximal branching factor of 2, a maximal path length of $\log_2 n$, a message overhead of $n - 1$, and a constant message imbalance factor of about 1, where n is the number of nodes in Chord. Table 1 shows that the partition-based broadcast algorithm also constructs a scalable and balanced DBT in the uniform identifier space.

We summarize the experimental results as follows. In the uniform identifier space, the token-based broadcast algorithm constructs a scalable and balanced DBT, where the maximal branching factor is at most two, the tree height is $O(\log n)$, and the construction overhead is $n - 1$ messages without explicit maintenance overhead. Compared with existing DHT-based algorithms, the token-based algorithm reduces 50%–82% of the message imbalance factor.

The partition-based algorithm constructs a scalable and balanced DBT in both the uniform and non-uniform identifier space. In the balanced DBT, the branching factor is at most two, and the tree height is $O(\log n)$, without any explicit construction and maintenance overhead. Compared with existing DHT-based algorithms, the partition-based algorithm reduces 67%–90% of the message imbalance factor in the non-uniform identifier space.

5. Related work

In general, broadcast is a special case of multicast. Since IP multicast suffers from deployment and business issues, recent focus has been shifted on application-level multicast and broadcast algorithms for massive content distribution in past decades [36]. These algorithms are categorized into tree-based and mesh-based approaches.

In the tree-based approach, a tree overlay rooted at a source node is constructed as the content delivering structure. For example, ESM [10] is a classic application-layer multicast system

Table 1
Comparisons between token-based and partition-based algorithms in the uniform identifier space.

Nodes	Max branching factor		Max path length		Message overhead		Message imbalance factor	
	Token	Partition	Token	Partition	Token	Partition	Token	Partition
16	2	2	4	4	15	15	1.1	1.1
32	2	2	5	5	31	31	1.0	1.0
64	2	2	6	6	63	63	1.0	1.0
128	2	2	7	7	127	127	1.0	1.0
256	2	2	8	8	255	255	1.0	1.0
512	2	2	9	9	511	511	1.0	1.0
1024	2	2	10	10	1023	1023	1.0	1.0
2048	2	2	11	11	2047	2047	1.0	1.0

towards small-sized groups for audio and video conferencing applications. Since each node of ESM needs to maintain a complete list of nodes in a group, it is hard to scale to large-scale multicast groups. NICE [11] is proposed to support a larger number of nodes using a hierarchical clustering approach. In NICE, periodic messages are exchanged between clusters to maintain the peer relationship, which incurs a high maintenance overhead. Bayeux [13] is proposed to construct a scalable and fault-tolerant application-layer multicast system by replicating root nodes and clustering receivers via an identifier over Tapestry [26]. Since the single tree structure is vulnerable to node dynamics, the multiple-tree approach has been proposed, where a forest with multiple disjoint sub-trees is constructed and each sub-stream of the content is delivered along each sub-tree. For example, SplitStream [14] is proposed to construct an interior-node disjoint forest of multiple Scribe trees [12] on top of Pastry [25], where a file is split into multiple stripes, each of which is pushed along each trees. CoopNet [15] is proposed to compute locally random or node-disjoint forests of trees in a manner similar to SplitStream, primarily designed for resilience to node departures.

Different from the flooding-based or gossip-based approach [37,38], the mesh-based approach constructs a data-driven mesh overlay as a swarm system, where each node has a small number of neighbors to exchange blocks of the content. For example, Bullet [16] and Bullet' [18] are proposed for large-file distribution in a wide area, where an overlay mesh is constructed over any overlay tree and each node transmits a disjoint set of blocks to its children. BitTorrent [19,39] is one of the most popular P2P content distribution systems, where a file is divided into multiple equal-sized blocks and all nodes upload and download blocks in parallel. CoBlitz [20] is proposed to provide large file transfer service by a simple URL-interface publicly accessed on the HTTP content distribution infrastructure. Although these mesh-based swarm systems can mitigate link stresses and performance bottlenecks of the origin, a large number of blocks traverse shared links repeatedly, which incurs enormous traffic stresses on the Internet Service Providers (ISP), especially on the large-scale computing infrastructures.

The tree-based and mesh-based approaches have advantages and disadvantages of content distribution. The content or each sub-stream is pushed along a well-defined parent–child path in the tree-based approach, while multiple blocks of the content are pulled from neighbors in the mesh-based approach. In the tree-based approach, the failure of a node may cause data outage in all its descendants and it is difficult to optimize the tree performance due to the presence of dynamic nodes. Yet the mesh-based approach suffers from longer delays and higher control overhead because of the lack of a well-defined parent–child relationship in the mesh overlay. Despite of these arguments [21,22] against the pros and cons of the two approaches, the tree-based approach is more suitable for the large-scale computing infrastructures because there is a large fraction of relatively stable dedicated nodes and the approach has the simplicity and controlled overhead. To succeed, the tree-based approach should have simple construction and fast maintenance algorithms. In this paper, we propose DHT-based lightweight broadcast algorithms to construct and maintain a scalable and balanced DBT with low overhead.

Moreover, distributed aggregation is also one of the fundamental primitives in the large-scale computing infrastructures. This primitive is used in recursively computing the global information by applying an aggregate function such as min, max, count, and sum on a set of local status. Related to our work, DHT-based aggregation algorithms [40] have been recently proposed for global resource monitor and discovery. For example, Astrolabe [6] is a DNS-like distributed management service by organizing the resources into a hierarchy of domains, specifying an aggregation tree between domains, and exchanging information across domains using a gossip protocol. SDIMS [7] is a scalable distribution information system, where each attribute is hashed to a key and the aggregation tree rooted at the key is built upon the routing mechanisms of Plaxton. SOMO [28] constructs a single aggregation tree by recursively dividing the identifier space into disjoint regions, each of which is assigned to a node of DHT. Willow [29] constructs a single aggregation tree on the hypercube-based Kademlia-like DHT to support multiple operations such as multicast, aggregation and publish/subscribe. A broadcast/aggregation tree [30] over DHT is implemented in a bottom-up approach by mapping nodes to their parents in the tree with a parent function. DAT [27] constructs a balanced aggregation tree for scalable Grid resource monitoring implicitly constructed from the native routing mechanisms of Chord in a bottom-up approach, without maintaining an explicit parent–child membership. STAR [9] constructs a multi-level aggregation tree for continuous aggregate queries over distributed data streams by self-tuning arithmetic imprecision to minimize the communication overhead under a fixed error budget, based on SDIMS [7]. These DHT-based aggregation algorithms are designed to provide a scalable all-to-one operation for collecting global information. On the contrary, our DHT-based lightweight broadcast algorithms are proposed to provide a scalable one-to-all operation for disseminating global information. Hence, the two types of algorithm are complemented to support scalable and efficient distributed information management in the large-scale computing infrastructures.

6. Conclusions

It is very challenging to support scalable and efficient broadcast in large-scale computing infrastructures. Although most DHT-based broadcast algorithms have been recently proposed using the greedy routing mechanisms of DHT, they suffer from the limitations of scalability and load balancing, incurring a high overhead of constructing and maintaining a DBT over DHT. This paper proposes DHT-based lightweight broadcast algorithms to overcome these limitations. Our algorithms leverage the topology and routing mechanisms of Chord to select appropriate children from the finger tables in a top-down approach. Thus, a balanced DBT is implicitly constructed from the novel routing paths of Chord, without explicitly maintaining the parent–child membership. We

propose the token-based broadcast algorithm over DHT in the uniform identifier space, and the partition-based broadcast algorithm over DHT in the arbitrary identifier space. In the token-based algorithm, each node selects the finger nodes as its children by a token value. In the partition-based algorithm, each node hierarchically partitions its identifier space into two subspaces and selects the agent nodes in the subspaces as its children. Theoretical analysis and experimental results shows that both our token-based and partition-based algorithms construct a scalable and balanced DBT, where the branching factor is at most two, and the tree height is $O(\log n)$, without any extra space for storing children and with no additional overhead for explicitly constructing and maintaining the DBT. Compared with existing DHT-based broadcast algorithms, the token-based and partition-based algorithms reduce 50%–82% and 67%–90% of the message imbalance factor respectively. Therefore, our proposed DHT-based lightweight broadcast algorithms are suitable for scalable and efficient distributed information management in large-scale computing infrastructures.

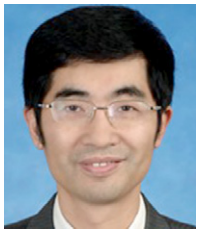
Acknowledgment

This work is supported by the National Science Foundation of China under Grant 60673155, 90718008 and 60873242, and the National Basic Research Program of China (973) under Grant 2007CB310702.

References

[1] A. Bavier, M. Bowman, B. Chun, et al., Operating system support for planetary-scale network services, in: Proc. of NSDI, San Francisco, California, 2004.
[2] I. Foster, C. Kesselman, S. Tuecke, The Anatomy of the grid: Enabling scalable virtual organizations, International Journal of Supercomputer Applications 2150 (2001).
[3] D. Kondo, G. Fedak, F. Cappello, A. Chien, H. Casanova, Characterizing resource availability in enterprise desktop grids, Future Generation Computer Systems 23 (7) (2007) 888–903.
[4] http://www.planet-lab.org.
[5] P. Thibodeau, Planet-Scale Grid, ComputerWorld, October 10, 2005.
[6] R.V. Renesse, K.P. Birman, W. Vogels, Astrolabe: A robust and scalable technology for distributed system monitoring, management, and data mining, ACM Transaction on Computer Systems 21 (2) (2003) 164–206.
[7] P. Yalagandula, M. Dahlin, A scalable distributed information management system, in: Proc. of Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, New York, 2004.
[8] D. Oppenheimer, J. Albrecht, D. Patterson, A. Vahdat, Design and implementation tradeoffs for wide-area resource discovery, ACM Transactions on Internet Technology 8 (2) (2008) 1–40.
[9] N. Jain, D. Kit, P. Mahajan, P. Yalagandula, M. Dahlin, Y. Zhang, STAR: Self-tuning aggregation for scalable monitoring, in: Proc. of VLDB, Vienna, Austria, 2007.
[10] Y. Chu, S.G. Rao, S. Seshan, H. Zhang, A case for end system multicast, in: Networking Support for Multicast, IEEE Journal on Selected Areas in Communication 20 (8) (2002) 1456–1471 (special issue).
[11] S. Baerjee, B. Bhattacharjee, C. Kommareddy, Scalable application layer multicast, in: Proc. of ACM SIGCOMM, 2002.
[12] M. Castro, P. Druschel, A.-M. Kermarrec, A. Rowstron, SCRIBE: A large-scale and decentralized application-level multicast infrastructure, IEEE Journal on Selected Areas in Communication 20 (8) (2002) 1489–1499.
[13] S. Zhuang, B. Zhao, A. Joseph, R. Katz, J. ubiatowicz, Bayeux: An architecture for scalable and fault-tolerant wide-area data dissemination, in: Proc. of ACM NOSSDAV, Port Jefferson, New York, 2001.
[14] M. Castro, P. Druschel, A. Kermarrec, A. Nandi, A. Rowstron, A. Singh, Splitstream: High-bandwidth multicast in cooperative environments, in: Proc of ACM SOSP, Bolton Landing, 2003.
[15] V.N. Padmanbhan, H.J. Wang, P.A. Chou, K. Sripanid-Kuchai, Distributed streaming media content using cooperative networking, in: Proc. of ACM NOSSDAV, Miami Beach, 2002.
[16] D. Kostic, A. Rodriguez, J. Albrecht, A. Vahdat, Bullet: High bandwidth data dissemination using an overlay mesh, in: Proc of ACM SOSP, Bolton Landing, 2003.
[17] L. Cherkasova, J. Lee, FastReplica: Efficient large file distribution within content delivery networks, in: Proc. of USITS, Seattle, Washington, 2003.
[18] D. Kostic, R. Braud, C. Killian, E. VandeKieft, J.W. Anderson, A.C. Snoeren, A. Vahdat, Maintaining high bandwidth under dynamic network conditions, in: Proc. of USENIX Annual Technical Conference, Anaheim, 2005.
[19] B. Cohen, Incentives build robustness in BitTorrent, in: Proc. of Workshop on Economics of Peer-to-Peer Systems, Berkeley, 2003.
[20] K. Park, V.S. Pai, Scale and performance in the CoBlitz large-file distribution service, in: Proc of NSDI, San Jose, California, 2006.
[21] V. Venkataraman, K. Yoshida, P. Fancis, Chunkspread: Heterogeneous unstructured end system multicast, in: Proc. of IEEE ICNP, Beijing, China, 2007.
[22] F. Wang, Y. Xiong, J. Liu, mTreebone: A hybrid tree/mesh overlay for application-layer live video multicast, in: Proc. of IEEE ICDCS, Toronto, Canada, 2007.
[23] I. Stoica, R. Morris, D. Karger, F. Kaashoek, H. Balakrishnan, Chord: A scalable peer-to-peer lookup service for Internet applications, in: Proc. of ACM SIGCOMM, San Diego, 2001.
[24] S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Shenker, A scalable content addressable network, in: Proc. of SIGCOMM, San Diego, 2001.
[25] A. Rowstron, P. Druschel, Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems, in: Proc. of IFIP/ACM International Conference on Distributed Systems Platforms, Heidelberg, 2001.
[26] B. Zhao, J. Kubiatowicz, A. Joseph, Tapestry: A fault-tolerant wide-area application infrastructure, ACM Computer Communication Review 32 (1) (2002).
[27] M. Cai, K. Hwang, Distributed aggregation algorithms with load-balancing for scalable grid resource monitoring, in: Proc. of IPDPS, Long Beach, California, 2007.
[28] Z. Zhang, S.-M. Shi, J. Zhu, SOMO: Self-organized metadata overlay for resource management in P2P DHT, in: Proc. of IPTPS, Berkeley, 2003.
[29] R.V. Renesse, A. Bozdog, Willow: DHT, aggregation, and publish/subscribe in one protocol, in: Proc. of IPTPS, La Jolla, 2004.
[30] J. Li, K. Sollins, D.-Y. Lim, Implementing aggregation and broadcast over distributed hash tables, SIGCOMM Computer and Communication Review 35 (1) (2005) 81–92.
[31] S. El-Ansary, L.O. Alima, P. Brand, S. Haridi, Efficient broadcast in structured P2P networks, in: Proc. of IPTPS, Berkeley, 2003.
[32] S. Ratnasamy, M. Handley, R. Karp, S. Shenker, Application-level multicast using content-addressable networks, in: Proc. of International Workshop on Networked Group Communicaion, London, UK, 2001.
[33] K. Huang, D. Zhang, A partition-based broadcast algorithm over DHT for large-scale computing infrastructures, in: Proc. of 4th International Conference on Grid and Pervasive Computing, Geneva, Switzerland, 2009.
[34] P2PSim. http://pdos.csail.mit.edu/p2psim.
[35] Meridian data set. http://www.cs.cornell.edu/People/egs/meridian.
[36] M. Hosseini, D.T. Ahmed, S Shirmohammadi, N.D. Georganas, A survey of application-layer multicast protocols, IEEE Communications Surveys & Tutorials 9 (3) (2007) 58–74.
[37] A. Demers, D. Greene, C. Houser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, D. Terry, Epidemic algorithms for replicated database maintenance, in: Proc. of ACM PODC, Vancouver, Canada, 1987.
[38] M. Lin, K. Marzullo, S. Massini, Gossip versus deterministically constrained flooding on small networks, in: Proc. of 14th International Conference on Distributed Computing, Toledo, Spain, 2000.
[39] K. Huang, L. Wang, D. Zhang, Y. Liu, Optimizing the BitTorrent performance using adaptive peer selection strategy, Future Generation Computer Systems 24 (7) (2008) 621–630.
[40] S. Zanikolas, R. Sakellariou, A taxonamy of grid mnitoring sytems, Future Generation Computer Sciences 21 (1) (2005) 163–188.
[41] A.R. Bharambe, S.G. Rao, V.N. Padmanabhan, S. Seshan, H. Zhang, The impact of heterogeneous bandwidth constraints on DHT-based multicast protocols, in: Proc. of IPTPS, Ithaca, 2005.

Kun Huang received his BS and MS degrees in Computer Science from Hunan University, China, in 2001 and 2004, respectively. He is currently a Ph.D. student in School of Computer and Communication at Hunan University. His research interests include peer-to-peer computing, intrusion detection, and survival systems. His email address is kylehuang@163.com.



Dafang Zhang is a Professor of Computer Science at Hunan University. He received the Ph.D. degree from Hunan University. He specializes in dependable systems and networks, fault-tolerant computing, network measurement, network security, and peer-to-peer computing. He has published over 200 original scientific papers and 11 popular books in these area. Presently, he leads 3 projects funded by the National Science Foundation of China and the Eleventh Five-Year Plan of National Defense Basic Scientific Research of China. Contact him at dfzhang@hunu.edu.cn.