Design of Broadcast Protocols for Non DHT-Based Pyramid Tree P2P Architecture

Koushik Maddali[*], Indranil Roy[*], Swathi Kaluvakuri[*], Bidyut Gupta[*]
Southern Illinois University; Carbondale, Il, USA

Narayan Debnath[†]
Eastern International University; VIETNAM

Abstract

In this paper, we have considered a recently reported 2-layer non-DHT-based structured P2P network. We have first proposed a bandwidth efficient inter-cluster broadcast protocol for the architecture assuming that the layer-1 tree is a complete one. The protocol does not generate any duplicate packet. However, when the layer-1 tree is an incomplete one, the proposed protocol will generate only one duplicate packet per broadcast packet and the number of such duplicate packets is independent of the number of peers in the architecture. In either case, complexity of broadcasting is *O*(d) in terms of overlay hops, d being the number of levels of the tree; so, it is independent of the total number of peers present in the architecture.

Key Words: Structured P2P network; residue class, interest-based; non-DHT; complete and incomplete pyramid trees; virtual neighbors; broadcast.

1 Introduction

Peer-to-Peer (P2P) overlay networks are widely used in distributed systems due to their ability to provide computational and data resource sharing capability in a scalable, self-organizing, distributed manner. There are two classes of P2P networks: unstructured and structured ones. In unstructured systems [3] peers are organized into arbitrary topology. It takes help of flooding for data look up. Problem arising due to frequent peer joining and leaving the system, also known as churn, is handled effectively in unstructured systems. However, it compromises with the efficiency of data query and the much-needed flexibility. Besides, in unstructured networks, lookups are not guaranteed. On the other hand, structured overlay networks provide deterministic bounds on data discovery. They provide scalable network overlays based on a distributed data structure which actually supports the deterministic behavior for data lookup. Recent trend in designing structured overlay architectures is the use of distributed hash tables (DHTs) [17-18, 26]. Such overlay architectures can offer efficient, flexible, and robust service [14, 17-18, 26, 28]. However, maintaining DHTs is a complex task and needs substantial amount of effort to handle the problem of churn. So, the major challenge facing such architectures is how to reduce this amount of effort while still providing an efficient data query service. In this direction, there exist several important works, which have considered designing DHT-based hybrid systems [7, 13, 15, 25, 29]; these works attempt to include the advantages of both structured and unstructured architectures. However, these works have their own pros and cons. Another design approach has attracted much attention; it is non-DHT based structured approach [9, 16-17, 21-23]. It offers advantages of DHT-based systems, while it attempts to reduce the complexity involved in churn handling. Authors in [21, 23] have considered one such approach and have used an already existing architecture, known as Pyramid tree architecture, originally applied to the research area of 'VLSI design for testability' [8]. Our structured architecture is an interest-based peer-to-peer system [1, 4, 10-12, 16, 21-24, 27]. In such a system, in general, peers with a common interest are clustered together. Its main focus is to improve the efficiency of data lookup protocols in that a query for an instance of a particular resource type is always directed to the cluster of peers which possess different instances of this resource type. So, success or failure to get an answer for the query involves a search in that cluster only instead of searching the whole overlay network as in the case of unstructured networks.

It may be noted that we have earlier reported simulation results [20] comparing the performance of data lookup protocols suitable for our architecture with those of some noted DHT based architectures [17-18, 26]. In addition, a comparison of the proposed architecture with some existing interest-based ones [1, 4, 10-12, 24, 27] and some important results on churn handling have also been reported in [20].

The overlay network considered in this paper is a 2-layer non DHT based architecture [21, 23]. At layer-1, there exists a tree like structure, known as pyramid tree. It is not a conventional tree. A node *i* in this tree represents the cluster-head of a cluster of peers which are interested in a particular resource of type R_i (i.e., peers with a common interest). The cluster-head is the first among these peers to join the system. Layer 2 consists of the different clusters corresponding to the cluster-heads. Details of the architecture appears in the next section.

Data lookup protocols [20] in the architecture are dependent on both intra and inter-cluster broadcasting. For intra-cluster broadcast, a capacity constrained approach has been reported in

[*] School of Computing. Emails: koushik@siu.edu, indranil.roy@siu.edu, swathi.kaluvakuri@siu.edu, bidyut@cs.siu.edu,
[†]School of Computing and Information Technology. Email: ndebnath@gmail.com.

[19]. Therefore, in the present work we consider only inter-cluster broadcast. In the present work, our objective is three-fold:

1) to design a broadcast protocol for a layer-1 complete tree with 100% bandwidth utilization, i.e., the protocol does not generate any duplicate packet,
2) to design a broadcast protocol for a layer-1 incomplete tree in a bandwidth efficient way such that it will not generate more than one duplicate packet per broadcast packet,
3) to achieve reasonably low latency for the protocols which must be independent of the total number of peers present in the architecture.

Residue Class based on modular arithmetic has been used to realize the overlay topology. In this paper, we have first proposed a bandwidth efficient inter-cluster broadcast protocol for the architecture assuming that the layer-1 tree is a complete one; the protocol does not generate any duplicate packet. However, we have observed that when the layer-1 tree is an incomplete one, whether the above-mentioned broadcast protocol designed for layer-1 complete tree will work correctly, depends on the location of a broadcast source in the tree. For broadcasting in an incomplete tree, we have borrowed some idea of Core-based tree (CBT) multicast [2] in which a multicast source always unicasts its packets to the core and the core in turn sends the packets to the intended receivers present in the core-based tree. In our present work, a broadcast source unicasts its packets to the root of the tree like in CBT. Then the root will execute a modified version of the proposed broadcast protocol designed for complete tree at layer-1.

The organization of the paper is as follows. In Section 2, we talk about some related preliminaries. Our contributions in the present paper appear in Section 3 in which we present the broadcast protocol for complete tree and in Section 4 in which we present the broadcast protocol for incomplete tree. Section 5 draws the conclusion.

2 Preliminaries

In this section, we present some relevant results from our recent work on the Pyramid tree based P2P architecture [21, 23] for interest-based peer-to-peer system.

Definition 1. *We define a resource as a tuple <R_i, V>, where R_i denotes the type of a resource and V is the value of the resource.*

Note that a resource can have many values. For example, let R_i denote the resource type 'songs' and V' denote a particular singer. Thus <R_i, V'> represents songs (some or all) sung by a particular singer V'.

Definition 2. *Let S be the set of all peers in a peer-to-peer system with n distinct resource types (i.e,. n distinct common interests). Then S = {C_i}, $0 \leq i \leq n-1$, where C_i denotes the subset consisting of all peers with the same resource type R_i. In this work, we call this subset C_i as cluster i. Also, for each cluster C_i, we assume that C_i^h is the first peer among the peers in C_i to join the system. We call C_i^h as the cluster-head of cluster C_i.*

2.1 Pyramid Tree

The following overlay architecture has been proposed in [21].

- The tree consists of n nodes. The ith node is the ith cluster head C_i^h. The tree forms the layer-1 and the clusters corresponding to the cluster-heads form the layer-2 of the architecture.
- Root of the tree is at level 1.
- Edges of the tree denote the logical link connections among the n cluster-heads. Note that edges are formed according to the pyramid tree structure [8].
- A cluster-head C_i^h represents the cluster C_i. Each cluster C_i is a completely connected network of peers possessing a common resource type R_i, resulting in the cluster diameter of 1.
- The tree is a complete one if at each level j, there are j number of nodes (i.e., j number of cluster-heads). It is an incomplete one if only at its leaf level, say k, there are less than k number of nodes.
- Any communication between a peer $p_i \in C_i$ and a peer $p_j \in C_j$ takes place only via the respective cluster-heads C_i^h and C_j^h and with the help of tree traversal wherever applicable.
- Joining of a new cluster always takes place at the leaf level.
- A node that does not reside either on the left branch or on the right branch of the root node is an internal node.
- Degree of an internal non-leaf node is 4.
- Degree of an internal leaf node is 2.

2.2 Residue Class

Modular arithmetic has been used to define the pyramid tree architecture of the P2P system.

Consider the set S_n of nonnegative integers less than n, given as $S_n = \{0, 1, 2, \ldots (n-1)\}$. This is referred to as the set of residues, or residue classes (mod n). That is, each integer in S_n represents a residue class (RC). These residue classes can be labelled as [0], [1], [2], …, [n – 1], where [r] = {a: a is an integer, a ≡ r (mod n)}.

For example, for n = 3, the classes are:

$$[0] = \{\ldots, -6, -3, 0, 3, 6, \ldots\}$$
$$[1] = \{\ldots, -5, -2, 1, 4, 7, \ldots\}$$
$$[2] = \{\ldots, -4, -1, 2, 5, 8, \ldots\}$$

In the P2P architecture, we use the numbers belonging to different classes as the logical (overlay) addresses of the peers with a common interest (i.e., peers in the same cluster) and the number of residue classes is the number of distinct resource types; for the sake of simplicity we shall use only the positive integer values.

Before we present the mechanism of logical address assignments, we state the following relevant property of residue class.

Lemma 1. Any two numbers of any class r of S_n are mutually congruent.

Proof. Let us consider any two numbers N' and N'' of class r. these numbers can be written as

$N' \equiv r \pmod{n}$; therefore, $(N' - r) / n$ = an integer, say I' (1)

and $N'' \equiv r \pmod{n}$; therefore, $(N'' - r) / n$ = an integer, say I'' .. (2)

Using (1) and (2) we get the following, $(N' - N'') / n = ((N' - r) - (N'' - r)) / n = I' - I''$ = an integer.

Therefore, N' is congruent to N''; that is, $N' \equiv N'' \pmod{n}$; also, $N'' \equiv N' \pmod{n}$ because congruence relation ($\equiv$) is symmetric. Hence, the proof.

2.3 Assignments of Overlay (Logical) Addresses

Assume that in an interest-based P2P system there are n distinct resource types. Note that n can be set to an extremely large value a priori to accommodate large number of distinct resource types. Consider the set of all peers in the system given as $S = \{C_i\}$, $0 \le i \le n-1$. Also, as mentioned earlier, for each subset C_i (i.e., cluster C_i) peer C_i^h is the first peer with resource type R_i to join the system and hence, it is the cluster-head of cluster C_i.

The assignment of overlay addresses to the peers in the clusters and the resources happens as follows:

1) The first cluster-head to join the system is assigned with the logical (overlay) address 0 and is denoted as C_0^h. It is also the root of the tree formed by newly arriving cluster-heads (see the example in Figure 1).
2) The $(i+1)^{th}$ newly arriving cluster-head possessing the resource type R_i is denoted as C_i^h and is assigned with the minimum nonnegative number (*i*) of *residue class i (mod n)* of the residue system S_n as its overlay address.
3) In this architecture, cluster-head C_i^h is assumed to join the system before the cluster-head C_{i+1}^h.
4) All peers having the same resource type R_i (i.e., 'common interest' defined by R_i) will form the cluster C_i. Each new peer joining cluster C_i is given the cluster membership address $(i + j.n)$, for i = 0, 1, 2, …
5) Resource type R_i possessed by peers in C_i is assigned the code *i* which is also the logical address of the cluster-head C_i^h of cluster C_i.

Definition 3. *Two peers of a cluster C_r are logically linked together if their assigned logical addresses are mutually congruent.*

Lemma 2. Each cluster C_r forms a complete graph.

Proof. According to Definition 3, two peers of any cluster C_r are logically linked together if their assigned logical addresses are mutually congruent. Also, from Lemma 1, we note that any two numbers of any class r of S_n are mutually congruent. Therefore, every peer has direct logical connection with every other peer in the same cluster C_r. Hence, the proof. □

Observation 1. *Any intra-cluster data look up communication needs only one overlay hop.*

Observation 2. *Search latency for inter-cluster data lookup algorithm is bounded by the diameter of the tree.*

Scalability: It may be noted that number of distinct resource types is very small compared to the number of peers in any overlay network [6]. To avoid the possibility of redesigning the architecture as new clusters are formed, a very large value of n can be selected at the design phase to a accommodate very large number of possible resource types (if needed in future). It means that if at the beginning number of resource types present is small, only the first few of the residue classes will be used initially for addressing; and as new clusters are formed in the future with new resource types in the system, more residue classes in sequence will be available for their addressing. For example, say initially n is set at 1000; so, there are 1000 possible residue classes, starting from [0], [1], [2], [4], [5], …., [999]. If initially there are only three clusters of peers present with three distinct resource types, the residue classes [0], [1], [2] will be used for addressing the peers in the three respective clusters. If later two new clusters are formed with two new resource types, the residue classes [3] and [4] will be used for addressing the peers in the two new clusters in sequence of their joining the system. Moreover, as we see, there is no limit on the size of any cluster because any residue class can be used to address logically up to infinite number of peers with a common interest. Therefore, the proposed architecture does not have any negative issue with scalability.

2.4 Virtual Neighbors [23]

An example of a complete pyramid tree of 5 levels is shown in Figure. 1. It means that it has 15 nodes/clusters (clusters 0 to 14, corresponding to 15 distinct resource types owned by the 15 distinct clusters). It also means that residue class with mod 15 has been used to build the tree. The nodes' respective logical addresses are from 0 to 14 based on their sequence of joining the P2P system.

Each link that connects directly two nodes on a branch of the tree is termed as a *segment.* In Figure. 1, a bracketed integer on a segment denotes the difference of the logical addresses of the two nodes on the segment. It is termed as *increment* and is denoted as *Inc* this increment can be used to get the logical address of a node from its immediate predecessor node along a branch. For example, let X and Y be two such nodes connected via a segment with increment *Inc*, such that node X is the immediate predecessor of node Y along a branch of a tree which is created using *residue class with mod n.* Then, *logical address of Y = (logical address of X + Inc) mod n.*

Thus, in the example of Figure 1,

Logical address of the leftmost leaf node = (logical address of its immediate predecessor along the left branch of the root + Inc) mod 15 = (6 + 4) mod 15 = 10.

Also, note that a *left branch* originating at node 2 on the right branch of the root node is *2 → 4 → 7 → 11*. Similarly, we can identify all other left branches originating at the respective nodes on the right branch of the root node. In a similar way, we can identify as well all right branches originating at the respective nodes on the left branch of the root node as well.

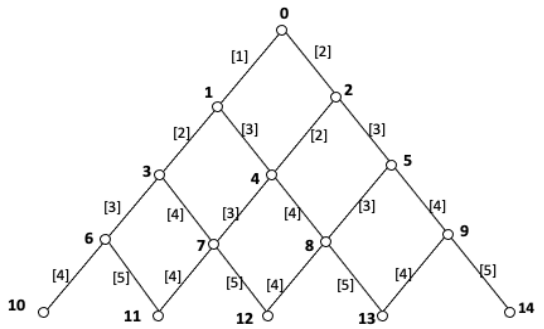

Figure 1: A complete pyramid tree with root 0

Remark 1. *The sequence of increments on the segments along the left branch of the root, appears to form an AP series with 1st term as 1 and common difference as 1.*

Remark 2. *The sequence of increments on the segments along the right branch of the root, appears to form an AP series with 1st term as 2 and common difference as 1.*

Remark 3. *Along the 1st left branch originating at node 2, the sequence of increments appears to form an AP series with 1st term as 2 and common difference as 1. Note that the 1st term is the increment on the segment 0 → 2.*

Remark 4. *Along the 2nd left branch originating at node 5, the sequence of increments is an AP series with 1st term as 3 and common difference as 1. Note that the 1st term is the increment on the segment 2 → 5.*

Authors [9] have presented some important structural properties of the pyramid tree P2P system. According to the authors, no existing structured P2P system, either DHT or non-DHT based, possesses these properties. These are stated below.

Let S_Y be the set of logical links that connect a node Y to its neighbors in a complete pyramid tree T_R with root R. Assume that the tree has n nodes (i.e., n cluster heads / n clusters). Let another tree T'_R be created with the same n nodes but with a different root R'. Let S'_Y be the set of logical links connecting Y to its neighbors in the tree T'_R.

Property 1. *$S_Y \neq S'_Y$*

Property 2. *Diameter of T_R = Diameter of T'_R*

Property 3. *Number of levels of T_R = Number of levels of T'_R*

Property 4. *Complexity of broadcasting in T_R with root R as the source of broadcast is the same for T'_R with root R'*

Property 5. *Both T_R and T'_R are complete pyramid trees.*

An example: Consider the complete pyramid tree of 5 levels as shown in Figure 2. Note that root of this tree is node 13, whereas root of the tree of Figure 1 is 0.

It is seen that S'_4 = {1,8,9} and S_4 = {1,2,7,8}. Therefore, Property 1 holds.

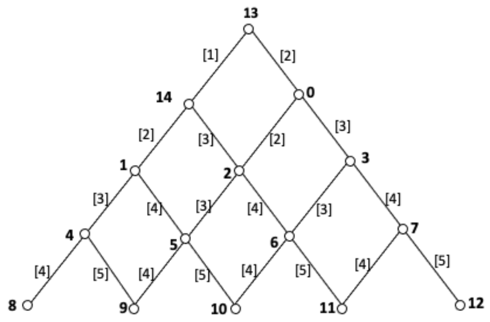

Figure 2: A complete pyramid tree with root 13

Diameters of both trees are same; it is 8 in terms of number of overlay hops. Besides, both trees use the same 15 nodes and have the same total number of levels. Complexity of broadcasting from either root 0 in the tree of Figure 1 or from root 13 in the tree of Figure 2 is bounded by the number of levels of each of the trees (here it is 4 in each). Finally, both trees are complete pyramid trees. Thus, all properties as mentioned above hold.

Remark 5. *Set of the neighbors of a given node Z may vary as the root of the tree varies. Hence, it is termed 'virtual'. However, time complexity of broadcasting remains same, i.e., it is O(d) where d denotes the number of levels of the tree.*

3 Broadcast in Complete Pyramid Tree

We assume that when a new cluster-head joins an existing tree, it contacts cluster-head 0, i.e., C_0^h which, in turn, assigns the newly joined one with the next logical address available for assignment (i.e., the next integer in the set S_n = {0, 1, 2,.... (n – 1)}, not yet assigned to any cluster-head). Therefore, this logical address becomes the largest address assigned so far in the tree. C_0^h broadcasts this information to all other nodes on the tree following the broadcast protocols (whichever is appropriate) proposed in this paper. Each receiving node, C_i^h then updates its table of information (*TOI*) that contains a tuple for each node C_k^h (cluster-head h). The tuple for C_k^h appears as < logical address, IP address, and resource code of the resource owned by C_k^h >. Recall that C_0^h is the first one to arrive and it forms the tree with only one node. It is seen that the table remains sorted after addition of any latest entry based on the logical addresses. In this work, 'cluster-head' and 'node' are used interchangeably.

Every node uses the knowledge of the largest existing logical address to determine the number of levels of the tree and if the current tree is complete or incomplete. Note that a d-level tree is complete if it has d number of nodes at level d; otherwise it is an incomplete one. This process takes place each time there is a new node joining the tree. The broadcast protocol works in one of two modes: 1) for complete tree, we name it *Broadcast-Complete* and 2) for incomplete tree, it is *Broadcast-Incomplete*. Therefore, we present first the algorithm executed by each node on the tree to determine its level and the completeness of the tree. Then the appropriate broadcast protocol is executed. Recall that root is at level 1; so, at any internal level k the tree has k number of nodes.

3.1 Determination of the Completeness of the Tree

The algorithm used to determine the completeness of the tree is presented below in Figure 3.

```
Algorithm 1. Algorithm-determine
1.    d = 1, Sum = 1            // d = number of levels;
                                Sum = number of nodes,
                                N_max = largest logical address
2.    J = d + 1
3.    do {
4.        Sum = Sum + J
5.        if  N_max = Sum
6.            d = J             // the tree is complete
7.        else
8.            if N_max <  Sum   // the tree is incomplete
9.                d = J
10.       else
11.           J = J+1
12.   continue
13.   }
```

Figure 3: Algorithm-determine

3.2 Broadcast in Complete Pyramid Tree

Broadcasting in Pyramid tree P2P systems can be either intra-cluster or inter-cluster broadcasting. Intra-cluster broadcast can be done using only one overlay hop because a cluster is a completely connected network. Note that for intra-cluster broadcast, a capacity constrained approach has already been reported [19]. In inter-cluster broadcast, a peer in cluster C_i may want to broadcast to all peers in the P2P system; effectively, inter-cluster broadcast involves always intra-cluster broadcast as well, with the exception when a cluster-head wants to update some control information maintained only by different cluster-heads in the system. Therefore, we focus specifically on broadcasting by a cluster-head C_i^h of a cluster C_i on the tree to all other cluster-heads on it.

Note that a broadcast source may be a member in a cluster say C_i or it can be the cluster-head C_i^h itself. For inter-cluster broadcast eventually, the cluster-head assumes the role of the source for further propagation of the packets in the rest of the architecture. Therefore, in this paper we denote a source as X which is a node (a cluster-head) on the tree.

In the following broadcast protocol, whenever node X wishes to broadcast, it will assume itself to be the root of the overlay tree during broadcasting. Note that according to the properties 1 to 5 related to *Virtual Neighbors* the tree with node X as its root is also a complete tree and hence, is the importance of the concept of *Virtual Neighbors*. We also assume that there are n distinct resource types and hence, the tree has n cluster-heads corresponding to n different clusters of respective common interests. Before we state the protocol formally, we state an informal sketch of the working of the protocol which may help in understanding clearly the formal presentation.

3.2.1 An Informal Sketch of the Broadcast Protocol

Step 1: Root X sends packets to its neighbors on left and right branches.

Step 2: Each receiving node on the left branch sends packets to its neighbor on this branch till a receiving node is a leaf node.

Step 3a: The i^{th} receiving node on the right branch sends packets to its neighbor on the i^{th} left branch originating at the i^{th} node until the i^{th} receiving node is a leaf node.

Step 3b: The i^{th} receiving node sends packets to its neighbor, the $(i+1)^{th}$ node on the right branch until the i^{th} receiving node is a leaf node.

Step 4: Propagation along the i^{th} left branch continues as in Step 2.

Note that instead of using the left branches originating at nodes on the right branch (as in step 3 above), the protocol can use the right branch and all right branches emanating from the nodes on the left branch. In this way, it will not generate any duplicate packet as well. The protocol is presented in Figure. 4. We use the following data structures and notations.

The structure of broadcast packet, BP appears as: < # hops (N_h), increment (Inc), flag (L/R), Information (Info) > Interpretation of the different entries in the broadcast packet P is stated below.

# hops (N_h): is initialized by the broadcast source X with (d-1); each receiving node on the tree along a propagation path will decrement N_h by 1, before forwarding the received packet to the next node along the path.

Increment (Inc): is used to determine the logical address of the next node for packet forwarding.

Flag (L/R): it is either L or R. Flag L denotes that a received packet needs to be propagated along a left branch until the leaf level is reached. Similarly, flag R denotes packet propagation along a right branch. For ease of understanding the protocols we name the broadcast packet BP as BP_L if flag is L; otherwise we name it BP_R.

Info: denotes the actual information to broadcast.

Theorem 1. Each node in a complete pyramid tree receives only one copy of a broadcast packet.

Proof. Propagation of the broadcast information (Info) takes place along the left and right branches of the root node; also, it takes place along all left branches originating at all nodes on

Algorithm 2. Protocol broadcast-complete

```
Executed by the broadcast source X (root of the tree):
1.  node X builds a broadcast packet $BP_L$
2.  $N_h = N_h-1$                                   // n = number of residue classes = number of distinct resource types
    Inc = 1
    flag = L                                        // propagation along the left branch of X takes place
    $BP_L$ packet = < $N_h$, Inc, L, Info >
    X forwards the $BP_L$ packet to the node with address, [(address (X) + Inc) mod n]
3.  node X builds broadcast packet $BP_R$
    $N_h = N_h-1$
    Inc = 2
    flag = R
    $BP_R$ packet = < $N_h$, Inc, R, Info >         // propagation along the right branch of X takes place
    X forwards the $BP_R$ packet to the node with address, [(address (X) + Inc) mod n]
Executed by a receiving node $C_i^h$
4.  if $N_h = 1$                                    // it is a leaf level node
5.      $C_i^h$ keeps a copy
6.      stops forwarding
7.  else
8.      if flag = L in the received packet          //build a new $BP_L$ packet
9.          $N_h = N_h-1$
10.         Inc = Inc+1                             // n = number of residue classes = number of distinct
11.         new $BP_L$ packet = < $N_h$, Inc, L, Info >   resource types
12.         $C_i^h$ forwards the $BP_L$ packet to the node with address,   // propagation along the left branch of $C_i^h$ continues
    [(address ($C_i^h$) + Inc) mod n]

13. else                                            // flag is R and $C_i^h$ is on the right branch of the
14.         Inc = Inc in the received packet          broadcast source
15.         $N_h = N_h-1$                           // build a new $BP_L$ packet
16.         flag = L
17.         new $BP_L$ packet = < $N_h$, Inc, L, Info >
18.         $C_i^h$ forwards the new $BP_L$ packet to the node with address,   // propagation along the left branch of $C_i^h$ continues
19.         [(address ($C_i^h$) + Inc) mod n]       // build a new $BP_R$ packet
20.         $N_h = N_h-1$
21.         Inc = Inc+1                             // propagation along the right branch of $C_i^h$ continues
22.         flag = R
23.         new $BP_R$ packet = < $N_h$, Inc, R, Info >
            $C_i^h$ forwards to the node with address, [(address ($C_i^h$) + Inc) mod n]
```

Figure 4: Protocol broadcast-complete

the right branch. Propagation stops when a receiving node is a leaf node. Therefore, all nodes receive the broadcast information. Besides, no internal node at any level that includes also the leaf level, receives a copy from more than one of its overlay uplinks. Hence, every node receives only one copy of the broadcast information.

Complexity

In the protocol, we observe that as a broadcast packet propagates along the right branch of the root (broadcast source), copies of the packet propagate simultaneously along the left branch of the root and along all other left branches originating at nodes (cluster-heads) lying on the right branch of the root (i.e., the idea of pipelining is implicitly present). Therefore, when the packet arrives at the leaf node (i.e., cluster-head C_{n-1}^h) on the right branch of the root, all other nodes on the tree must have received already a copy of the packet each. It also means that by the time broadcasting finishes in 1 hop in the cluster C_{n-1} (diameter of each cluster is 1 overlay hop), it is finished in all other clusters as well. We note that a broadcast packet takes 1 hop in the source cluster to arrive at its cluster-head if the source peer is not the cluster-head itself, (d-1) hops to arrive at the cluster-head C_{n-1}^h from the root cluster-head, and 1 more hop to complete broadcasting in the cluster C_{n-1}. Therefore, all together a maximum of (d+1) hops are required to finish broadcasting in C_{n-1}; and it ensures also completeness of broadcasting in the whole network. Hence, the time complexity is *O*(d). In other words, complexity depends only on the number of the distinct resource types n present in the system, because n defines the value of d and the complexity is independent of the number of peers in the network. Note that a *mod value of n* is used to build the pyramid tree and $n \approx 2^d$.

Bandwidth utilization

The protocol offers 100% bandwidth utilization since it does not generate any duplicate packet (Theorem 1). In this context, it is worth mentioning that since it is not truly a tree, so there is probably only one other option left for broadcasting which is via flooding by the source of broadcast (root). It is observed that if flooding is used, total number of duplicate packets par broadcast packet for a d-level pyramid tree is, $N_{dup} = (d-1).\ (d-2)/2$. This number, N_{dup} increases substantially with d (see Figure. 5). As a result, these duplicate packets per broadcast packet will chew up reasonably good amount of network's precious bandwidth, especially when multiple broadcast sessions take place simultaneously with large number of packets per broadcast session. On the other hand, if flooding is controlled to avoid the generation of duplicate packets, nodes at any level i whose downlinks are incident on a common node at level (i+1) must exchange control packets in order to avoid sending duplicate copies of a broadcast packet to the common node; the number of such control packets is the same as the number of duplicate packets, N_{dup}. In addition, controlled flooding-based broadcasting will take longer time to complete than the uncontrolled flooding one because of the exchanges of control information. Therefore, when compared with the proposed protocol, flooding, controlled or uncontrolled, is not an option from the viewpoints of efficient bandwidth utilization and time to complete a broadcasting session.

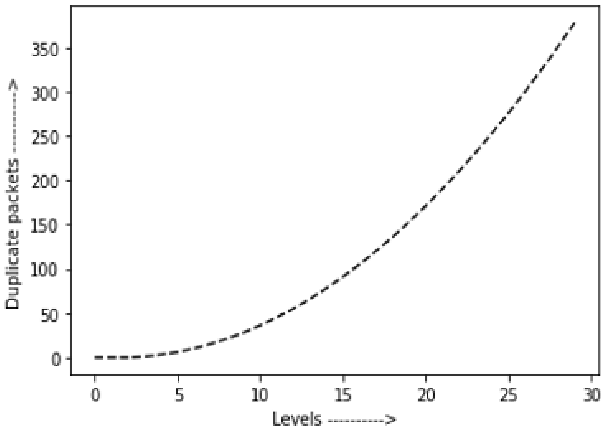

Figure 5: Variation of number of duplicate packets (N_{dup}) with level (d) of a pyramid tree when flooding is used

4 Broadcast in Incomplete Pyramid Tree

We will now analyze why the proposed broadcast protocol designed for complete pyramid trees cannot work correctly for incomplete trees. Consider the incomplete pyramid tree as shown in Figure. 6. Let us assume that *mod* value used to form the tree is 10. We find that in this tree, so far 8 clusters have joined, i.e., peers possessing 8 distinct resource types are present currently in the P2P system. There is still the provision for 2 more clusters to join the tree to make the tree a complete one with 4 levels. Otherwise, it remains incomplete. We note that if the current root node with logical address 0 broadcasts using the broadcast protocol stated earlier, each existing node in the incomplete tree will receive exactly one copy. Besides, each node will receive the broadcast packet from only one of its uplinks. Therefore, there is no possibility of duplicate packet generation. Also, note that node 5 will not forward further because it should have the knowledge that the largest logical address present in the tree is now 7 and it could only forward to nodes 8 and 9 if they would exist (shown as isolated ones). Besides, with node 0 at the root we see that the tree remains connected because of the way the architecture is defined.

Let us assume that node 7 wants to broadcast. Therefore, if we follow the Broadcast-complete protocol, node 7 will assume the position of the root. However, we find that the tree cannot be built because the two children of node 7, namely nodes 8 and 9 do not exist (Figure. 7). This leads us to design the broadcast protocol in incomplete pyramid trees in a bit incomplete one (Figure. 6).

A broadcast source node X will unicast its packets to the root node 0, which in turn, will execute a modified version of the Broadcast-Complete protocol in the incomplete tree with itself being the root. That is, node 0 will act as the pseudo broadcast

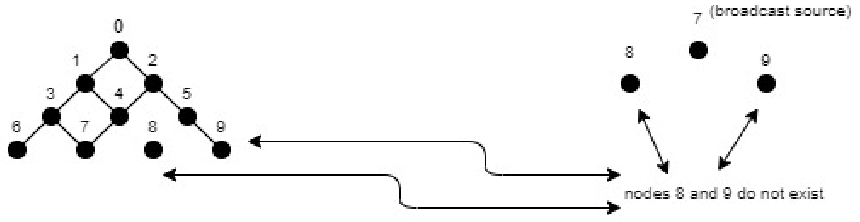

Figure 6: Connected Incomplete tree with root

Figure 7: Tree cannot be built: nodes 8 and 9 do not exist

different way. It is observed that with node 0 at the root, the structure is always a connected tree, complete (Figure.1) or an Working principle of the protocol is stated below.

source (like in a CBT multicast [2] the core is the pseudo multicast source). Since node X is a part of the tree, so eventually it will participate in the broadcast by node 0 and will receive a copy of the packet which it already unicasted to node 0. Note that node X may need to forward the received packet downward depending on its location on the tree. Therefore, this approach will generate only one duplicate packet per broadcast packet irrespective of the size of the tree. The protocol appears in Figure. 8.

In this context, it may be mentioned that if cluster-head C_0^h (i.e., node 0) along with its all member peers in C_0 have left the network, the cluster-head with current logical address as 1 assumes the role of the root of the tree and its logical address becomes 0 and at the same time any other cluster-head with logical address H will have its newly assigned logical address as (H-1); the table of information (TOI) will be updated accordingly, which will reflect a new, possibly incomplete, yet connected, tree with its root as node 0 (formerly node 1). However, it is all about 'churn handling' which we have not included in this paper, and has already been reported in [20]. Therefore, in the following algorithm by 'node 0' it means the current root.

Theorem 2. The protocol generates exactly one extra copy per packet broadcast.

Proof. As in the case of protocol Broadcast-complete, each broadcast packet is received only once by each node on the tree except the source node X. Since node X is a part of the tree, so eventually it will participate in the broadcast by node 0 and will receive a copy of the packet which it already unicasted to node 0. Therefore, there is only one extra packet generated per packet broadcast. □

Complexity

The hop complexity is O(d) and as in Broadcast-Complete protocol, complexity depends only on the number of the distinct resource types n present in the system.

Bandwidth utilization

It offers very high bandwidth utilization because it generates only one duplicate packet per broadcast packet and the number of such duplicate packets is independent on the total number of peers present in the network.

5 Discussion on Some Related Works

In our works on designing communication protocols [20] suitable for our already proposed non-DHT based architecture, we have used broadcasting (without its formal description). In the present work, we have presented the broadcast protocols formally. It may be noted that broadcasting has been considered only in very few works related to non-DHT based architecture. In a significant contribution in this area [5] authors have considered a non-DHT based P2P architecture and have proposed a method of broadcasting in order to achieve multicasting with hop complexity $O(\log_c^n)$ where n is the total number of peers in the architecture and c is the average capacity of a peer.

In our proposed work if we consider capacity constrained approach for intra-cluster broadcast, the number of hops required to broadcast in a cluster is $\log_c^Z$ instead of 1 logical hop, given that average capacity of a peer is c, and average cluster size is Z. Thus, a broadcast packet takes at most $\log_c^Z$ hops in the root-cluster (pseudo 'source cluster') to arrive at the root cluster-head, (d-1) hops to arrive at the cluster-head C_{n-1}^h from the root cluster-head, and $\log_c^Z$ hopes to complete broadcasting in the cluster C_{n-1}. Because of the implicit pipelining idea, by the time broadcasting finishes in cluster C_{n-1}, it is finished in all other clusters as well. Hence, the complexity depends on d and cluster size Z only; it is independent of the total number of peers in the network unlike in [5].

Besides, we have discussed in detail earlier if flooding should be used for broadcasting and we have concluded that (see Figure 5) flooding, controlled or uncontrolled, is not an option from the viewpoints of efficient bandwidth utilization and time to complete a broadcasting session.

Observation 3. *In capacity constrained approach search latency is a function of the number of distinct resource types and the number of peers present in a cluster; it is independent of the total number of peers in the overlay network.*

6 Conclusion

In this paper, we have considered a 2-level non DHT-based P2P architecture. This architecture has been the choice because (1) we have shown earlier [20] its superiority from the viewpoint of search latency of data lookup protocols compared to those in some very prominent DHT-based contributions [17-18, 26] and (2) its superiority over several existing interest-based architectures [1, 4, 10-12, 24, 27]. In this paper, we have presented an interesting way to design a broadcast protocol when the level-1 tree is not complete. We have used an architectural property, viz. 'virtual neighbors' in the design. The protocol is 100% bandwidth efficient. Besides, most noteworthy points of the proposed Broadcast-Incomplete protocol are (1) it is highly bandwidth efficient in that the number of duplicate packets is equal to the number of broadcast packets used in a broadcast session; (2) as in case of broadcast in complete tree, broadcast latency is independent of the number of peers in the system and depends only on the number of distinct resource types present in the system. Both protocols have the same complexity, viz. O(d).

It may be noted that we have earlier reported simulation results [20] comparing the performance of data lookup protocols suitable for our architecture with those of some noted DHT based architectures [17-18, 26]. In addition, a comparison of the proposed architecture with some existing interest-based ones [1, 4, 10-12, 24, 27] and some important results on churn

Algorithm 3. Protocol Broadcast-Incomplete

```
Executed by broadcast source X
1.  Node X unicasts packets to node 0 for broadcasting
Executed by root node 0    // node 0 acts as the pseudo broadcast source
2.  $N_h = N_h-1$                                   // node 0 builds a broadcast packet $BP_L$
3.  Inc = 1
4.  flag = L                                        // n = number of residue classes = number of distinct resource types
5.  $BP_L$ packet = < $N_h$, Inc, L, Info >         // propagation along the left branch of node 0 takes place
6.  Node 0 forwards the $BP_L$ packet to the node with address, [(address (X) + Inc) mod n]
7.  $N_h = N_h-1$                                   // node 0 builds a broadcast packet $BP_R$
8.  Inc = 2
9.  flag = R
10. $BP_R$ packet = < $N_h$, Inc, R, Info >         // propagation along the right branch of node 0 takes place
11. Node 0 forwards the $BP_R$ packet to the node with address, [(address (X) + Inc) mod n]
Executed by a receiving node $C_i^h$
12. if  $C_i^h \neq X$
13.         $C_i^h$ keeps a copy
14. else    does not keep a copy
15. if  $N_h = 1$                                   // it is a leaf level node
16.         $C_i^h$ keeps a copy
17.         stops forwarding
18. else
19.     if  flag = L in the received packet
20.         if [(address ($C_i^h$) + (Inc + 1)) mod n] $\leq N_{max}$    // $N_{max}$ is the largest current logical address in the tree
21.             $N_h = N_h-1$                       // build a new $BP_L$ packet
22.             Inc = Inc+1                         // n = number of residue classes = number of distinct resource types
23.             new $BP_L$ packet = < $N_h$, Inc, L, Info >
24.             $C_i^h$ forwards the $BP_L$ packet to the node with address, [(address ($C_i^h$) + Inc) mod n]    // propagation along the left branch of $C_i^h$ continues
25.         else
26.             stops forwarding                    // no such address exists; tree is incomplete
27.     else                                        // flag is R and $C_i^h$ is on the right branch of the broadcast source
28.         if [(address ($C_i^h$) + (Inc)) mod n] $\leq N_{max}$
29.             Inc = Inc in the received $BP_R$ packet    // build a new $BP_L$ packet
30.             $N_h = N_h-1$
31.             flag = L
32.             new $BP_L$ packet = < $N_h$, Inc, R, Info >
33.             $C_i^h$ forwards the new $BP_L$ packet to the node with address, [(address ($C_i^h$) + Inc) mod n]    // propagation along the left branch of $C_i^h$ continues
34.         else
35.                 stops forwarding                // no such address exists; tree is incomplete
36.         if [(address ($C_i^h$) + (Inc + 1)) mod n] $\leq N_{max}$
37.                 $N_h = N_h-1$                   // build a new $BP_R$ packet
38.                 Inc = Inc+1
39.                 flag = R
40.                 new $BP_R$ packet = < $N_h$, Inc, R, Info >
41.                 $C_i^h$ forwards to the node with address, [(address ($C_i^h$) + Inc) mod n]    // propagation along the right branch of $C_i^h$ continues
42.     else                                        // no such address exists; tree is incomplete
43.                 stops forwarding
```

Figure 8: Protocol broadcast-incomplete

handling have also been reported in [20].

Future work is directed at designing P2P Federation using our model architecture as the basic architectural component of the Federation.

References

[1] L. Badis, M. Amad, D. Aîssani, K. Bedjguelal, and A. Benkerrou, "ROUTIL: P2P Routing Protocol Based on Interest Links", 2016 International Conference on Advanced Aspects of Software Engineering (ICAASE), Constantine, pp. 1-5, 2016, doi: 10.1109/ICAASE.2016. 7843852.

[2] T. A. Ballardie, "Core Based Tree Multicast Routing Architecture", Internet Engineering Task Force (IETF), RFC 2201, September 1997.

[3] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker, "Making Gnutella-Like P2P Systems Scalable", *Proc. ACM SIGCOMM*, Karlsruhe, Germany, pp. 401-418, August 25-29, 2003.

[4] W. T. Chen, C. H. Chao, and J .L. Chiang, "An Interested-Based Architecture for Peer-to-Peer Network Systems", 20th International Conference on Advanced Information Networking and Applications (AINA'06), Vienna, 1:707-712, 2006, doi: 10.1109/AINA.2006.93.

[5] S. Chen, B. Shi, S. Chen, and Y.Xia, "ACOM: Any-Source Capacity-Constrained Overlay Multicast in Non-DHT P2P Networks", *IEEE Tran. Parallel and Distributed Systems*, 18(9):1188-1201, Sep. 2007.

[6] J. Cheng and R. Donahue, "The Pirate Bay Torrent Analysis and Visualization", *IJCSET*, 3(2):38-42, Feb. 2013.

[7] P. Ganesan, Q. Sun, and H. Garcia-Molina, "YAPPERS: A Peer-to-Peer Lookup Service Over Arbitrary Topology", IEEE Infocom 2003, Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies, (IEEE Cat. No.03CH37428), San Francisco, USA, 2:1250-1260, 2003, doi: 10.1109/INFCOM. 2003.1208961.

[8] B. Gupta and M. Mohsin, "Fault-Tolerance in Pyramid Tree Network Architecture," *J. Computer Systems Science and Engineering*, 10(3):164-172, July,1995.

[9] B. Gupta, N. Rahimi, S.Rahimi, and A. Alyanbaawi, "Efficient Data Lookup in Non-DHT Based Low Diameter Structured P2P Network," 2017 IEEE 15th International Conference on Industrial Informatics (INDIN), Emden, Germany, pp. 944-950, 2017, doi: 10.1109/INDIN.2017. 8104899.

[10] M. Hai and Y. Tu, "A P2P E-Commerce Model Based on Interest Community", 2010 International Conference on Management of e-Commerce and e-Government, Chengdu, 2010, pp. 362-365, doi: 10.1109/ICMeCG.2010.80.

[11] M. Khambatti, K. D. Ryu, and P. Dasgupta, "Structuring Peer-to-Peer Networks Using Interest-Based Communities", Lecture Notes in Computer Science, 1st International Workshop, DBISP2P 2003, Berlin, September 2003.

[12] S. K. A. Khan and L. N. Tokarchuk, "Interest-Based Self Organization in Group-Structured P2P Networks", 2009 6th IEEE Consumer Communications and Networking Conference, Las Vegas, NV, pp. 1-5, 2009, doi: 10.1109/CCNC.2009.4784959.

[13] M. Kleis, Eng KKeong Lua,, and Xiaoming Zhou, "Hierarchical Peer-to-Peer Networks using Lightweight Superpeer Topologies", 10th IEEE Symposium on Computers and Communications (ISCC'05), pp. 143-148, 2005, doi: 10.1109/ISCC.2005.78.

[14] D. Korzun and A. Gurtov, "Hierarchical Architectures in Structured Peer-to-Peer Overlay Networks", *Peer-to-Peer Networking and Applications*, Springer, pp. 1-37, March 2013.

[15] Z. Peng, Z. Duan, J. Qi, Y. Cao, and E. Lv, "HP2P: A Hybrid Hierarchical P2P Network", First International Conference on the Digital Society, (ICDS'07), pp. 18-18, 2007, doi: 10.1109/ICDS.2007.20.

[16] N. Rahimi, K. Sinha, B. Gupta, S. Rahimi, and N. C. Debnath, "LDEPTH: A Low Diameter Hierarchical P2P Network Architecture", 2016 IEEE 14th International Conference on Industrial Informatics (INDIN), Poitiers, France, pp. 832-837, July 2016, doi: 10.1109/INDIN.2016.7819275.

[17] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content-addressable network, CAN" , *ACM*, 31(4):161-172, 2001.

[18] A. Rowstron and P. Druschel, "Pastry: Scalable, Distributed Object Location and Routing for Large Scale Peer-to-Peer Systems", *Proc. FIP/ACM Intl. Conf. Distributed Systems Platforms (Middleware)*, pp. 329-350, 2001.

[19] I. Roy, S. Kaluvakuri, K. Maddali, A. Aydeger, B. Gupta, and N. Debnath, "Capacity Constrained Broadcast and Multicast Protocols for Clusters in Pyramid Tree-Based Structured P2P Network", *IJCA*, 28(3):122-131, Sep. 2021.

[20] I. Roy, S. Kaluvakuri, K. Maddali, Z. Liu, and B. Gupta, "Efficient Communication Protocols for Non DHT-based Pyramid Tree P2P Architecture", *WSEAS Transactions on Computers*, (Invited paper), 20:108-125, July 2021.

[21] I. Roy, B. Gupta, B. Rekabdar, and H. Hexmoor, "A Novel Approach Toward Designing a Non-DHT Based Structured P2P Network Architecture", EPiC Series in Computing, *Proceedings of 32nd Int. Conf. Computer Applications in Industry and Engineering*, 63:121-129, 2019.

[22] I. Roy, K. Maddali, S. Kaluvakuri, B. Rekabdar, Z. Liu, B. Gupta, and N. Debnath, "Efficient Any Source Overlay Multicast in CRT-Based P2P Networks ─ A Capacity - Constrained Approach", 2019 IEEE 17th International Conference on Industrial Informatics (INDIN), Helsinki, Finland, pp. 1351-1357, 2019, doi: 10.1109/INDIN41052.2019. 8972151.

[23] I. Roy, N. Rahimi, K. Maddali, S. Kaluvakuri, B. Gupta, and N. Debnath, "Design of Efficient Broadcast Protocol for Pyramid Tree-based P2P Network Architecture", EPiC Series in Computing, *Proceedings of 33rd Int. Conf. Computer Applications in Industry and Engineering, San Diego*, 63:182-188, 2020.

[24] H. Shen, G. Liu and L. Ward, "A Proximity-Aware Interest-Clustered P2P File Sharing System", *IEEE Transactions on Parallel and Distributed Systems*, 26(6):1509-1523, doi: 10.1109/TPDS.2014.2327033, 1 June 2015.

[25] K. Shuang, P. Zhang, and S. Su, "Comb: A Resilient and Efficient Two-Hop Lookup Service for Distributed Communication System", *Security and Communication Networks*, 8(10):1890-1903, 2015.

[26] I.Stocia, R. Morris, D. Liben-Nowell, D. R. Karger, M. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications", *IEEE/ACM Tran. Networking*, 11(1):17-32, Feb. 2003.

[27] Z. Tu, W. Jiang, and J. Jia, "Hierarchical Hybrid DVE-P2P Networking Based on Interests Clustering", 2017 International Conference on Virtual Reality and Visualization (ICVRV), Zhengzhou, China, pp. 378-381, 2017, doi: 10.1109/ICVRV.2017.00087.

[28] M. Xu, S. Zhou, and J. Guan, "A New and Effective Hierarchical Overlay Structure for Peer-to-Peer Networks", *Computer Communications*, 34:862-874, 2011.

[29] M. Yang and Y. Yang, "An Efficient Hybrid Peer-to-Peer System for Distributed Data Sharing", *IEEE Trans. Computers*, 59(9):1158-1171, Sep. 2010.

Koushik Maddali (photo not available) is a Ph.D. candidate in Department of Computer Science at Southern Illinois University Carbondale. He received his MS from the same university and his BS from Jawaharlal Nehru Technological University, India. His research interests include Peer to Peer Networking, BlockChain and worked on a Virtual Terminal project of Cisco from 2017-2018.

Indranil Roy (photo not available) is currently a PhD candidate in Computer Science Department of Southern Illinois University, Carbondale. He has completed his B.E in Electronics & Communication from RCCIIT, Kolkata, in the year 2016. He received his M.S in Computer Science from Southern Illinois University, Carbondale. His main research interests include blockchain along with interest-based P2P architecture.

Swathi Kaluvakuri (photo not available) is a Ph.D. candidate from Southern Illinois University Carbondale – School of Computing. She graduated from Jawaharlal Nehru Technological University with a Bachelor of Technology degree in Computer Science major. She holds a keen interest in the areas of Peer to Peer Networking and BlockChain and worked as a Software Engineer, Technical Product Support and IBM AS400 developer for NetCracker Pvt Ltd from 2012-2014.

Bidyut Gupta (photo not available) received his M. Tech. degree in Electronics Engineering and Ph.D. degree in Computer Science from Calcutta University, Calcutta, India. At present, he is a professor at the School of Computing (formerly Computer Science Department), Southern Illinois University, Carbondale, Illinois, USA. His current research interest includes design of architecture and communication protocols for structured peer-to-peer overlay networks, security in overlay networks, and block chain. He is a senior member of IEEE and ISCA.

Narayan Debnath (photo not available) earned a Doctor of Science (D.Sc.) degree in Computer Science and also a Doctor of Philosophy (Ph.D.) degree in Physics. Narayan C. Debnath is currently the Founding Dean of the School of Computing and Information Technology at Eastern International University, Vietnam. He is also serving as the Head of the Department of Software Engineering at Eastern International University, Vietnam. Dr. Debnath has been the Director of the International Society for Computers and their Applications (ISCA) since 2014. Formerly, Dr. Debnath served as a Full Professor of Computer Science at Winona State University, Minnesota, USA for 28 years (1989-2017). Dr. Debnath has been an active member of the ACM, IEEE Computer Society, Arab Computer Society, and a senior member of the ISCA.