Capacity Constrained Broadcast and Multicast Protocols for Clusters in a Pyramid Tree-based Structured P2P Network

Indranil Roy*, Swathi Kaluvakuri*, Koushik Maddali*
Abdullah Aydeger*, Bidyut Gupta*
Southern Illinois University at Carbondale, Carbondale, Illinois, USA

Narayan Debnath†
Eastern International University, VIETNAM

Abstract

In this paper, we have considered an existing non-DHT based structured P2P network. It is an interest-based system. At the heart of the architecture, there exists a tree like structure, known as Pyramid Tree, even though it is not a conventional tree. A node on the tree represents a cluster of peers with common interest. There is no limit on the size of such clusters. Residue Class based on modular arithmetic has been used to realize the structure of a cluster. It has been shown that overlay diameter of each such cluster is just one (one overlay hop). Thus, each cluster is a completely connected network. Therefore, theoretically any peer in such a cluster is logically connected to every other peer in the cluster. However, since peers are heterogeneous in nature, therefore, in practice we have to consider their different capacities while designing any communication protocol inside the cluster. In this paper, we have addressed this issue and offered reasonably efficient solutions for broadcasting and multicasting considering peer heterogeneity.

Key Words: P2P network, structured, non-DHT based, pyramid tree, capacity constrained.

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 [2] 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) [10, 15, 18]. Such overlay architectures can offer efficient, flexible, and robust service [7, 10, 15, 16, 18]. 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 [4, 6, 9, 14, 17]; 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 [3, 11-13]. It offers advantages of DHT-based systems, while it attempts to reduce the complexity involved in churn handling. Authors in [11, 13] 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' [5]. The P2P architecture has two levels. At the heart of it, it is a tree structure (pyramid tree); it is not a conventional tree. This tree forms the first level of the system. Each node on the tree represents uniquely a cluster-head of a cluster of peers with common interest and the cluster head is the first peer to join the system among the peers in this cluster. Such clusters form the second level of the architecture. Residue class based on modular arithmetic has been used to realize the architecture. Some of the main advantages of the system are its low data lookup efficiency and ease of churn handling. In this paper, we have considered such architecture and have dealt with a practical issue related to the architecture as detailed in below.

Problem Statement. In our earlier proposed pyramid tree P2P architecture [11, 13], every cluster has an overlay diameter of 1. Each such cluster may consist of a very large number of peers with common interest. It means that every peer in any such cluster C_i has direct logical connection to all other peers inside the cluster. In reality, peers are capacity constrained and it is most likely that any cluster will have heterogeneous peers; therefore, peers can be differently capacity constrained. Hence, even though the overlay diameter is 1, in practice a peer can

*E-mail: [indranil.roy, swathi.Kaluvakuri, Koushik]@siu.edu, [aydeger, Bidyut]@cs.siu.edu,
† E-mail: Narayan.debnath@eiu.edu.vn

communicate only to few other peers at a given time depending on its capacity. In this paper, we address this issue and offer reasonably efficient solutions for broadcasting and multicasting considering peer heterogeneity.

Our Contribution. We have earlier designed an inter-cluster broadcast protocol [13] in which a participating node (cluster-head) may have to activate a maximum of only three of its links at a time for the propagation of a broadcast packet along the pyramid tree. So, the protocol appears to have followed implicitly an effective capacity constrained approach, even though that was not the objective at the time of designing the protocol. Consider the following reasonably efficient capacity constrained architecture consisting of the peers in any cluster C_i. We logically restructure the peers inside cluster C_i in the following way: we partition the peers in C_i in a number of pyramid trees of identical sizes (except possibly the last one, explained later) and implement the idea of our already designed broadcast protocol on these trees inside the cluster. Note that in the original version of the inter-cluster broadcast protocol, a node in the tree is a cluster-head, whereas when applied inside a cluster a node in the tree can be any peer in the cluster that includes the cluster-head of the cluster as well. Let us first state the capacity constrained broadcast protocol. Later, we shall consider multicasting.

This paper is organized as follows. In Section 2, we talk about some related preliminaries and in Section 3, we present the capacity constrained broadcast along with the proposed restructuring method of peers inside a cluster. In Section 4, we present the capacity constrained multicast protocol. Section 5 draws the conclusion.

2 Related Preliminaries

In this section, we present some relevant results from our recent work on the pyramid tree based P2P architecture [11, 13] 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}, $0 \le i \le n-1$, 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 [13].

1) The tree consists of n nodes. The ith node is the ith cluster head C_i^h.
2) Root of the tree is at level 1.
3) 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 [5].
4) 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.
5) The tree is a complete one if at each level j, there are j number of nodes (i.e., j number of cluster-heads).
6) 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.
7) Joining of a new cluster always takes place at the leaf level.
8) A node that does not reside either on the left branch or on the right branch of the root node is
   an internal node.
9) Degree of an internal non-leaf node is 4.
10) 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 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 [13].

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

2.3 Assignments of Overlay 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).

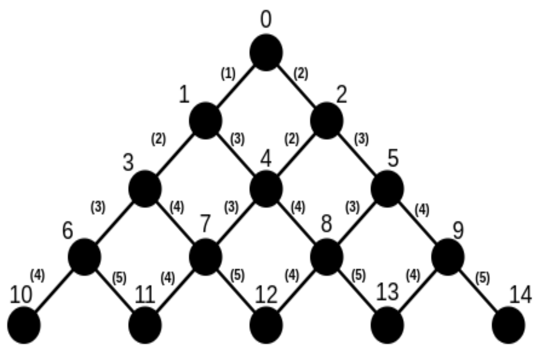


Figure 1: A complete pyramid tree with root 0

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 $i = 0, 1, 2, \ldots$

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 group G_r are logically linked together if their assigned logical addresses are mutually congruent.*

Lemma 2. *Each group C_i forms a complete graph.*

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

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

2.4. Virtual Neighbors [13]

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 (overlay) 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 *incremen*t 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 + increment) 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.

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 [13] 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 which 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 group 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.

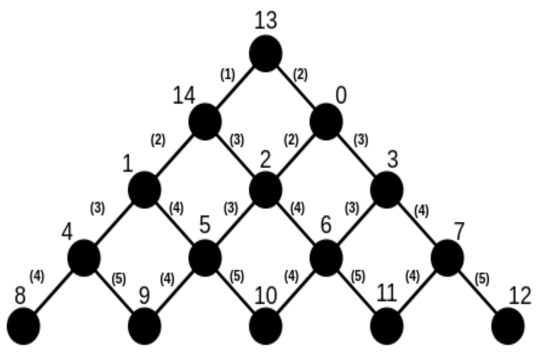


Figure 2: A complete pyramid tree with root 13

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

Diameters of both trees are the 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. Broadcasting from either root 0 in the tree of Figure 1 or from root 13 in the tree of Figure 2 can be completed in 4 hops. 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

The following note on broadcasting will help in understanding better the proposed approach on capacity constrained broadcast.

2.5 Broadcast in Complete and Incomplete Pyramid Trees

We now state an informal sketch of the broadcast protocol [13] for complete pyramid tree architecture. It has been shown that the protocol does not generate any duplicate packet. The protocol uses properties 1 to 4 as stated above. It works as follows: whenever a node X on the tree wishes to broadcast, it will assume itself as the root of the overlay tree during broadcasting.

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 it is a leaf node.*

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

For incomplete pyramid tree architecture, a broadcast source unicasts its packets to the root of the tree and the root then broadcasts the packets in the tree following the broadcast protocol designed for complete trees. Justification of the source itself not broadcasting its packets has been worked out in detail in one of our on-going projects [8]. It has been proven that only one duplicate packet will be generated per broadcast packet and it is independent of the total number of peers present in the P2P system [8].

3 Capacity Constrained Broadcast

Before we state the protocol formally, we first illustrate in detail the partitioning of the peers in a cluster.

3.1 Partitioning of Peers

We illustrate the partitioning process with an appropriate example. Let us consider a pyramid tree architecture consisting of cluster-heads of different distinct interests. Each cluster-head in turn is connected to peers of common interest belonging to its cluster. Let us call this tree the Global Pyramid Tree (GPT). Assume that a mod value of M has been used for the formation of the GPT. To illustrate the partitioning scheme, let us consider a cluster C_i with cluster-head C_i^h in the GPT. Let C_i consist of 40 peers. Let us assume that inside cluster C_i we use a mod value of 10 to build four complete pyramid trees T_1, T_2, T_3, and T_4 such that each one consists of 10 nodes. It is shown in Figure 3. Observe that the last tree T_4 not necessarily has to be a complete one, it depends on the number of peers inside a cluster; however, it has no effect on our explanation of the idea. We call each such tree inside cluster C_i a local pyramid tree (LPT). Note that since the mod value is 10 for building the LPTs, the peers in any such LPT will have secondary overlay addresses from 0 to 9, respectively (Figure 3). We shall use these secondary overlay addresses for restructuring the LPTs. Why are we considering pyramid trees inside a cluster? The reason is that we have already designed efficient communication protocols for pyramid tree architecture, and it is logical as well as reasonable to consider similar protocols inside a cluster; it offers uniformity in terms of implementation both at the levels of GPT and LPT.

Let the logical address of the cluster-head C_i^h be i based on the mod value of n used in the GPT formation. Then, the other 39 peers in C_i will have the respective overlay addresses as (i+n), (i+2n), …, (i+39n) based on their sequence of joining the cluster. These addresses are termed as primary overlay addresses. The roots of the four trees have the respective addresses as i, (i+10n), (i+20n), and (i+30n). Since C_i is completely connected, therefore, any two peers in the cluster are logically connected. However, to incorporate the idea of 'capacity constrained', we assume that among the roots of the four trees, only the links that connect the neighboring roots are present. That is, only links between T_1 and T_2, T_2 and T_3, and

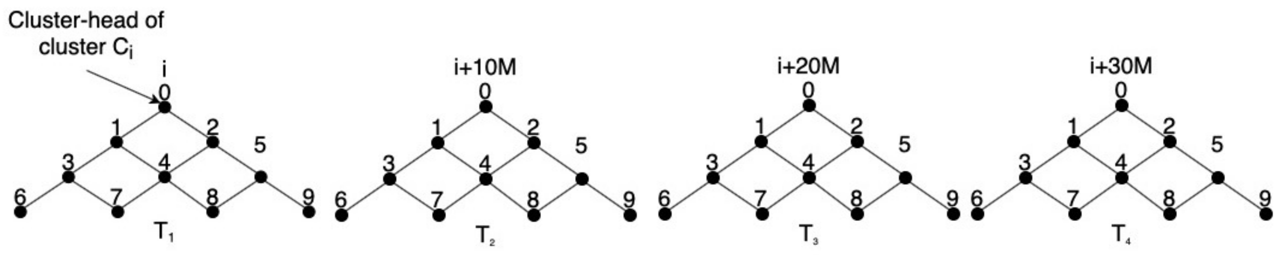


Figure 3: Cluster C_i has four component trees T_1, T_2, T_3, and T_4

In addition, we shall not use in our broadcast protocol any logical link that connects two peers in two different LPTs, none of which is a root in the corresponding tree.

Besides the above-mentioned primary overlay addresses based on the mod value of n, we also assign tertiary overlay addresses only to the roots of the LPTs inside a cluster. If a cluster C_i has n number of LPTs, the root of the first LPT formed will be assigned a tertiary overlay address 1, the root of the second LPT with address 2; in a similar way the root of the j^{th} LPT will have the address j. These tertiary overlay addresses are used in the proposed broadcast and multicast protocols inside a cluster.

Cluster-head C_i^h assigns the addresses (i+n) to (i+9n) to the first nine peers joining the cluster besides the cluster-head itself. It forms the tree T_1 with itself as the root. Note that the virtual links' information among the peers in T_1 define implicitly its pyramid tree structure. Cluster-head C_i^h assigns the next 10 arriving peers with addresses (i+10n) to (i+19n) and imparts the responsibility of becoming the root of the tree T_2 and forming the complete tree T_2 to the peer with address (i+10n). In this way, the other trees are also formed. As pointed out earlier, the last tree T_4 may not be a complete one; it depends on the number of the arriving peers joining the tree. For broadcasting (also for multicasting) inside a tree, we assume that the root as well as each pear in the tree will maintain a local list of the overlay and IP addresses of all peers in the tree. Thus, in this example, each peer in T_1 will maintain the addressing information of 10 peers including that of itself. In general, for an LPT T_j, we denote the table as $\mathcal{T}_j$. In addition, the link information of a peer in the tree (i.e., which other peers it is connected to) is present in the table as well. Thus, effectively, structure of the pyramid tree is actually embedded (implicitly) in the table in the form of the links' information that a peer in the tree is linked with some particular other peers.

Besides, cluster-head C_i^h and the roots of all other LPTS will maintain a separate table $\mathcal{T}_r$ containing the tertiary overlay and the IP addresses of the root of each LPT in the cluster. Observe that table size depends on the mod value used to create the LPTs. At one end it will be a single tree with all peers in the cluster in it; this is not a practical approach since a cluster may have tremendously large number of peers resulting in high broadcast/multicast latency; in addition, table size maintained by each peer will be as large as the number of peers in the cluster. *So, a practical approach will be to use reasonably small mod value to create the trees and this mod value can be the choice of the designers; it can also be dynamically changed because after all these trees are virtual.*

3.2 Restructuring of the LPTs

For the proposed protocols (broadcast and multicast) to work correctly, we need to consider the effect on an existing LPT caused by peers joining and leaving (churn handling). Let us start with the peers joining first.

When we consider new peer joining, only the last LPT may get affected structurally from the following viewpoint. According to our proposed way of forming the LPTs in a cluster, say C_i, process of any new peer joining the cluster will be taken care of by only the root of the last LPT. For example, in Figure 3, the last LPT is the tree T_4 and its root is the peer with overlay address (i+30n). If the tree T_4 is already a complete one, then a new tree T_5 will be formed with its root having the address (i+40n) and all subsequent joins will take place in T_5 unless it is full; and the process of new tree-formation will go on as needed as explained above.

Observation 3. *Any new peer joining a cluster C_i will not affect structurally any LPT other than the last one.*

Observation 4. *If the existing last LPT is an incomplete one, new peers joining may turn it into a complete one, or it may remain an incomplete one.*

We now consider the effect on the structures of the LPTs due to peers leaving. Unlike joins, leaving of peers can take place at any time in any LPT and therefore, it can affect structurally any LPT. Therefore, a complete LPT may become incomplete after some peer(s) leave it. In addition, a complete tree may remain a complete one as well if multiple peers leave the tree; however, the new one will have a smaller number of levels. Besides, based on the positions of the leaving peers in an LPT, the pyramid tree architecture of the concerned LPT may be destroyed completely. See Figures 4 and 5. In the trees shown in these figures a peer with secondary overlay address X appears as X(k); k is the IP address of peer X. A detailed explanation of the trees in these figures appears later in this section. The need for such a representation will be clear shortly. The structure of an LPT may be destroyed after a peer leaves; it depends on the position of the leaving peer on the tree. Therefore, some efficient restructuring process need to be executed after peers leave the LPTs so that the characteristics of pyramid tree architecture can be retained, be it a complete or an incomplete one after peers leave and hence, the existing broadcast / multicast protocols can be applied in the restructured trees with some possible graceful degradation. Below we have stated the restructuring method after peers leave.

3.2.1 Restructuring Method. As mentioned earlier, each peer in the jth LPT, viz. T_j maintains the addressing information of all peers in T_j including that of itself in a table T_j. This information includes a peer's overlay address and the IP address.

The following two situations are considered and we state the methods to handle these so that the characteristics of pyramid tree architecture can be retained.

Case 1: Any peer p with secondary overlay address X in an LPT T_j other than its root j leaves.

We assume graceful degradation; right before leaving the tree T_j, peer p unicasts a 'leave' message to the root j of T_j. Root j will delete the entry corresponding to the peer p with address X from its table T_j and assigns a virtual address X to the peer which had earlier the address (X+1) mod m where *mod m* is used to build the tree T_j. In this way, readdressing of the overlay addresses for all peers following the leaving one will take place. Root j forms a new table T_j' and unicasts this updated table to the rest of the peers of the tree T_j. Thus, the structure of a pyramid tree remains intact after the peer p leaves; of course, the tree T_j now can become an incomplete one. It may also remain as a complete one depending on the number of the peers leaving; however, in that case its level will be smaller than the original version of the tree T_j. Observe that structure of the pyramid tree is actually embedded in the table in the form of the links' information that a peer in the tree is linked with some particular other peers, along with the overlay addresses of the peers.

Case 2: Root j of T_j leaves

Step 1. Root j unicasts a 'leave' message to the peer p' with the next higher overlay address.

Step 2. Peer p' becomes the next root and assigns its new tertiary overlay address as j.

Step 3. Peer p' assigns new overlay addresses to the rest of the available peers, i.e., a previous address Y now becomes (Y-1).

Step 4. New root p' forms a new table T_j' and unicasts it to the other peers.

Step 5. A new tree T_j' is built. This new tree may be an incomplete or complete one depending on the total number of the leaving peers.

Observation 5. *Any combination of peers leaving an LPT can be handled as in either Case 1 or 2. If the combination involves the root, case 2 will be considered; otherwise it is Case 1.*

Example (Case 1)

Let us consider the jth LPT of some cluster as shown in Figure 4, peers' secondary overlay addresses are from 0 to 9 and the tree is a complete one. To explain the situation of Case 1 clearly, we assume that the peers' respective IP addresses are a, b, …, j. Assume that peer 4 leaves. The structure will be no more that of a pyramid tree after the leave (Figure 5). However, based on the proposed restructuring method, peer 5 in Figure 4 now has the logical address 4, similarly peer 6 in Figure 4 now has the logical address 5 and in that way peer 9 has 8. The new pyramid tree after restructuring is shown in Figure 6. Note that in Figure 6 peer 4 with IP address f is different from peer 4 with IP address e in Figure 4. That is, peer 4 in Figure 6 is actually peer 5 in Figure 4. Note that peers with logical addresses 0 to 3 have not gone through any change. Root j constructs a new table T_j' and unicasts to the other peers in this tree.

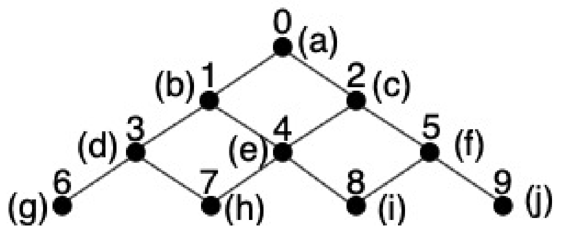
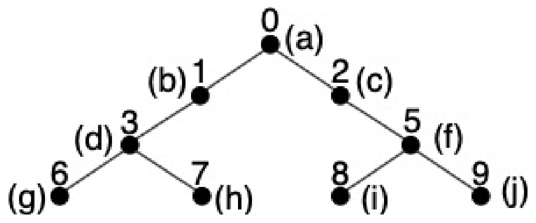
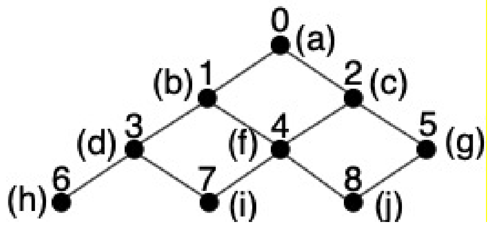


Figure 4: Before peer 4 leaves





Example (Case 2)

Consider the tree of Figure 4. Assume that the root peer with overlay address 0 and IP address a is leaving. Based on the restructuring method, right before leaving peer 0 unicasts a leave message to peer 1. Peer 1 (b) now becomes the new root and its overlay address becomes 0 and it will convert any other overlay address Y to (Y-1). It builds a new table T_j' and unicasts it to the rest of the peers. The new information about the links along with the changed overlay addresses produces the tree as shown in Figure 7. Thus, the structural properties remain intact except that the tree is now an incomplete one.

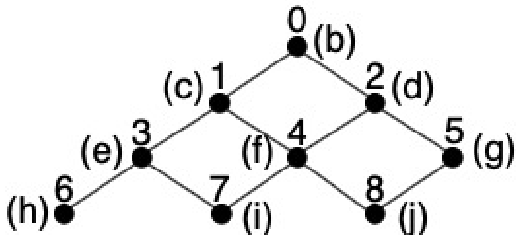


Figure 7: After restructuring, it remains a pyramid tree

3.3 Protocol Capacity-Constrained-Broadcast

Earlier we have discussed the effect of the leaving of peers on the existing structure of an LPT. Since after such leaving an LPT may become an incomplete pyramid tree even after restructuring, therefore, we shall consider only the protocol Broadcast-Incomplete in an LPT. We do it irrespective of the completeness of the tree because protocol Broadcast-Complete is applicable only to complete LPT, whereas Broadcast-Incomplete work both in complete and incomplete LPTs. It may be noted that in Broadcast-Incomplete in an LPT a source peer first unicasts its packets to the root of the tree (after restructuring wherever applicable); the root in turn broadcasts the packets to the rest of the tree. It generates only one extra packet per packet broadcast. This is the only bad effect of the protocol. Note that if the root itself is the source of broadcast, neither any extra packer is generated, nor there is any additional unicast.

In the proposed protocol (see Figure 8), we assume the following: a cluster C_j has k number of LPTs and the tertiary overlay address of the root of the jth LPT is j ($1 \leq j \leq k$). We denote the jth LPT as LPT_j and we call the root simply as root j (its tertiary address is r also). As mentioned earlier, cluster-head C_j^h maintains a separate table T_r containing the tertiary overlay and the IP addresses of the root of each LPT in the cluster; whereas every other root saves the tertiary overlay and the IP addresses of the root(s) of its neighboring LPT(s) only. In the following protocol, unicasting to an LPT means unicast to the root of the LPT.

Protocol

Based on the location of the broadcast source, we need to consider the following three possible situations:

Source peer is present in LPT_k; or is present in LPT_1; or is present in some LPT_r, $r \neq k$ and $r \neq 1$

```
if j = k          / Source peer is present in the last LPT (LPT_k)
    unicasts to LPT_{j-1}
    root j executes Broadcast-Incomplete protocol in its LPT_k
    j = j-1
    while 1< j
        root j unicasts to LPT_{j-1}
        root j executes Broadcast-Incomplete protocol in LPT_j
        j = j-1
    continue
    root 1 executes Broadcast-Incomplete in LPT_1
---------------------------------------------------------------
if j = 1               / Source is present in the first LPT_1
    while j < k
        root j unicasts to LPT_{j+1}
        root j executes Broadcast-Incomplete protocol in LPT_j
        j = j +1
    continue
    root k executes Broadcast-Incomplete protocol in LPT_k
---------------------------------------------------------------

if 1 < j < k
        / source is present in some other LPT_r, r ≠ k and r ≠1

step 1.  root j unicasts to LPT_{j-1} and LPT_{j+1}
         root j executes Broadcast-Incomplete protocol in
         LPT_j
step 2.  t = j-1
         while 1< t
             root t unicasts to LPT_{t-1}
             root t executes Broadcast-Incomplete protocol
             in LPT_t
             t = t-1
          continue
          root 1 executes Broadcast-Incomplete protocol in
          LPT_1
step 3.  t' = j+1
         while t' < k
             root t unicasts to LPT_{t'+1}
             root t' executes Broadcast-Incomplete protocol
             in LPT_{t'}
             t' = t'+1
          continue

         root k executes Broadcast-Incomplete protocol in
         LPT_k
```

Figure 8: The protocol

Theorem 1. Each peer in a cluster C_j receives a copy of each broadcast packet.

Proof. The protocol ensures that each root gets a copy of each broadcast packet. Each root then executes Broadcast-Incomplete' in its tree. Since 'Broadcast-Incomplete' guarantees that each peer in a tree receives a copy of each broadcast packet; hence is the proof. □

Performance

We shall discuss first broadcast latency followed by memory utilization by any root.

Let a cluster C_j be divided into n LPTs. Let us first consider the worst case in which the source of broadcast is either in the 1st LPT or in the last one, the kth LPT. Let it be in the 1st LPT. Let us assume that all LPTs have identical number of levels and it is l. It takes (k-1) hopes for a broadcast packet to reach the kth root (i.e., the root of the kth LPT). Observe that the concept of pipelining is implicitly present in the protocol; that is, by the time a broadcast packet arrives at the l^{th} level of the kth tree, broadcast is already complete in all other trees. Therefore, the number of hops to complete the broadcast in the worst case is [(k-1) + (l-1)]. In addition, it is clear that if the kth LPT has less than l number of levels, the total number of hops as mentioned above is sufficient to complete the broadcast.

If we consider the best case, i.e., when the broadcast source is present in a tree located at the middle. Total number of hops = [(k-1)/2 + (l-1)].

From the viewpoint of the memory used by a peer in any LPT_j to save its table T_j or by the cluster-head C_j^h (and any other root) to save its tables T_j and T_r, we have observed the following using clusters of different sizes. Broadcast latency does not vary much with m (m is the mod value used to create the LPTs inside the cluster); however, there is considerable increase in the memory requirement to save the tables as we increase the value of m. In fact, memory requirement varies linearly with cluster size. Therefore, it may be suggested that the designers select a reasonably small value of m for efficient use of the memory of the peers.

4 Capacity Constrained Multicast

We denote the LPT containing the source of multicasting as LPT_s. The source peer, say peer p, in LPT_s registers with the root s. In general, the corresponding root s can itself be the source of multicasting as well. Source peer p first registers with the root s and then during multicasting, it unicasts its packets to the root s and the root s in turn sends the multicast packets to the peers in its tree which have joined the multicast session; actually, these peers form a core-based tree (CBT) [1] with s as its root. In general, we denote a CBT with core j as CBT_j. In addition, root s is also responsible to send multicast packets to the roots of the other trees that are interested in receiving the packets. So effectively the root s acts as the source of multicast during multicasting. The proposed protocol will use the relevant information present in the tables of the roots as in the case of capacity constrained broadcast. A multicast session consists of three phases which are stated below.

Phase 1: Roots learning about the interested peers

Step 1. Source root s of the LPT_s broadcasts a 'query' message in all the component trees. This can be accomplished by executing the capacity constrained broadcast protocol.

Step 2. Source root s forms the CBT_s with s as its root if some peers in LPT_s join the core s for receiving multicast packets from the source peer p of multicasting.

/ CBT_s is formed with s as its root

Phase 2: Formation of core-based trees (CBTs) in other component trees and across the cluster

Step 1. Root j unicasts a join request to the source root s if it receives any join request(s) from peers in LPT_j

Step 2. Root j forms a CBT_j consisting of the interested peers in LPT_j

/ this is the CBT inside LPT_j

Step 3. A 2-level CBT with root s is formed with other joining roots as its leaves. The maximum diameter of this tree is 2.

/ this 2-level CBT with root s is formed across the cluster

Phase 3: Multicasting from source p

Step 1. Source peer p unicasts multicasts packets to source root s

Step 2. Root s sends the packets to the joining cores in the 2-level CBT.

Root s multicasts the packets to the peers in the CBT_s in LPT_s.

Step 3. Each joining core j multicasts the packets to the peers in the CBT_j in LPT_j.

Performance

We assume that for a given mod value of m, the cluster contains k number of LPTs. The trees have the same level l, except possibly the last one that may have fewer peers not enough to make the level l. However, for simplicity we assume that all LPTs have the same level. Therefore, k.m is the total number of peers in the cluster before partitioning. The protocol builds CBTs with maximum level l inside some LPTs containing multicast receivers and a 2-level CBT across the cluster with leaves as some roots of some LPTs interested in receiving multicast packets.

A multicast packet travels from a source peer p to the source root s in a maximum of (l-1) hops, then to all other roots in the 2-level CBT in 1 hop, and also to all group members in the receiving LPTs in a maximum of (l-1) hops.

Note that multicast in the source LPT goes on along with the multicasts in other LPTs; so, idea of pipelining is implicitly present. Hence, we can safely assume that approximately in (l-1) hops multicast in the related LPTs can be completed.

Therefore, multicast latency in hops is (l-1) +1+ (l-1), i.e., (2l-1); and it is independent of the size of the cluster. Hence, time complexity is $O(l)$. Since $l \approx 2^m$, therefore, it may be suggested that the designers select a reasonably small value of m for low latency multicast in the cluster.

5 Conclusions

In this paper, we have considered a recently reported non-DHT based structured P2P system. The main advantages of the architecture are its very low data lookup latency and ease of churn handling compared to most DHT-based P2P systems. In this architecture, a cluster consists of peers with common interest and its overlay diameter is only one hop. In reality, because of peer heterogeneity, peers are differently capacity constrained and therefore, lookup latency of $O(1)$ inside a cluster may not be achievable in reality. It has led us to propose practical approaches for both broadcasting and multicasting inside a cluster of peers considering peer heterogeneity.

We are now investigating how the proposed structure can be used/modified in order to reduce the traffic and operating costs of Internet Service Provider (ISP) and P2P Service Provider.

References

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

[2] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker, "Making Gnutella-Like P2P Systems Scalable," *Proc. ACM SIGCOMM*, Karlsruhe, Germany, pp. 407-418, August 25-29 2003.

[3] Shiping Chen, Baile Shi, Shigang Chen, and Ye 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.

[4] P. Ganesan, Q. Sun, and H. Garcia-Molina, "Yappers: A Peer-to-Peer Lookup Service Over Arbitrary Topology," *Proc. IEEE Infocom 2003*, San Francisco, USA, 2:1250-1260, March 30 - April 1, 2003.

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

[6] M. Kleis, E. K. Lua, and X. Zhou, "Hierarchical Peer-to-Peer Networks using Lightweight SuperPeer Topologies," *Proc. IEEE Symp. Computers and Communications*, pp. 143-148, 2005.

[7] 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.

[8] Koushik Maddali, Indranil Roy, Swathi Kaluvakuri, and Bidyut Gupta, "Efficient Broadcast Protocols for Complete and Incomplete Pyramid Tree P2P Architecture," under preparation.

[9] Z. Peng, Z. Duan, J. Jun Qi, Y. Cao, and E. Lv, "HP2P: A Hybrid Hierarchical P2P Network," *Proc. Intl. Conf. Digital Society*, pp. 86-90, 2007.

[10] 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.

[11] Indranil Roy, Bidyut Gupta, Banafsheh Rekabdar, and Henry 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*, Las Vegas, NV, 63:121-129, 2019.

[12] Indranil Roy, Koushik Maddali, Swathi Kaluvakuri, Banafsheh Rekabdar, Ziping Liu, Bidyut Gupta, and Narayan Debnath, "Efficient Any Source Overlay Multicast in CRT-Based P2P Networks ─ A Capacity - Constrained Approach," *Proc. IEEE 17th Int. Conf. Industrial Informatics (IEEE INDIN)*, Helsinki, Finland, pp. 1351-1357, July 2019.

[13] Indranil Roy, Nick Rahimi, Koushik Maddali, Swathi Kaluvakuri, Bidyut Gupta, and Narayan 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, CA, 63:182-188, 2020.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] B. Y. Zhao, L. Huang, S. C. Rhea, J. Stribling, A. Zoseph, and J. D. Kubiatowicz, "Tapestry: A Global-Scale Overlay for Rapid Service Deployment," *IEEE J-SAC*, 22(1):41-53, Jan. 2004.

Indranil Roy (photo not available) is currently a PhD student 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 degree in Computer Science from Southern Illinois University, Carbondale in 2018. 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 Net Cracker Pvt Ltd from 2012-2014.

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.

Abdullah Aydeger (photo not available) is an Assistant Professor in the Department of Computer Science at Southern Illinois University, Carbondale. He received his M.S. and a Ph.D. degree from the Department of Computer Engineering at Florida International University in 2016 and 2020 and his B.S. degree in Computer Engineering from Istanbul Technical University in 2013. His research interests include Software Defined Networking, Network Function Virtualization, Moving Target Defense, and their utilization for different network security and resiliency problems. He applies the ideas not only to traditional ISP networks but also to emerging network domains within cyber-physical systems and IoT. He has published papers in reputable journals and conferences. He has also contributed two book chapters. Dr. Aydeger has served as a reviewer for many conferences and journals.

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 Blockchain. 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.