On Generalization of Residue Class Based Pyramid Tree P2P Network Architecture

Indranil Roy*
Southeast Missouri State University, Cape Girardeau, MO

Nick Rahimi†,
[2] University of Southern Mississippi, Hattiesburg, MS

Ziping Liu*
Southeast Missouri State University, Cape Girardeau, MO

Bidyut Gupta‡
Southern Illinois University; Carbondale, IL

Narayan Debnath§
Eastern International University; VIETNAM

Abstract

In this paper, we have considered a recently reported 2-layer non-DHT-based structured P2P network. It is an interest-based system and consists of different clusters such that peers in a given cluster possess instances of a particular resource type. It offers efficient data look-up protocols with low latency. However, the architecture lacks in one very important aspect: it is assumed that no peer in any cluster can have more than one resource type and this could be a very hard restriction practically. Therefore, in the present work, we have addressed this issue of generalizing the architecture to overcome this restriction and have come up with effective solutions. We have modified appropriately our previously reported data look-up protocols wherever applicable in order to accommodate the idea of generalization while making sure that look-up latencies of these modified protocols remain the same.

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

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 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) [18, 20, 27]. Such overlay architectures can offer efficient, flexible, and robust service [14, 18, 20, 27, 29]. 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, 16, 26, 30]; 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 [4, 9, 17, 21, 24]. It offers advantages of DHT-based systems, while it attempts to reduce the complexity involved in churn handling. Authors in [21] 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, 19]. Our structured architecture is an interest-based peer-to-peer system [1, 5, 10, 11-12, 17, 21, 24-25, 28]. In such a system, 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,

* Department of Computer Science.
† School of Computing Sciences & Computer Engineering.
‡ School of Computing.
§ School of Computing and Information Technology.

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.

The overlay network considered in this paper is a 2-layer non DHT based architecture [21]. 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 possess instances of a particular resource 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.

Related works and our Contribution: Before we state our present contributions, we briefly state now some of our recent related contributions. The pyramid tree architecture was initially used in the area of Testable Fault Tolerant Arrays of Processors [8, 19]. Later we realized its potential as a probable network architecture for P2P communication systems especially for interest-based systems. In [21] authors studied extensively what the architecture could offer from the viewpoints of data look-up efficiency and they identified several interesting architectural properties (stated in the Preliminaries) that finally led to the design of various simple yet very efficient data look-up protocols [15, 22-23]. It is a non-DHT-based two-level structured architecture and experimental results [23] have shown the superiority of the proposed various data look-up protocols when compared with the protocols used in some noted DHT-based structured networks from the viewpoints of search latency and complexity involved. It offers as well several advantages when compared with some noted works on interest-based architectures [1, 5, 10, 11-12, 25, 28]. Authors have extensively studied the effect of churn on the architecture [23]; besides another important contribution was the design of intra-capacity constrained broadcast and multicast protocols which take into consideration the real situation where peers most likely will be heterogeneous [22].

However, we believe that all these above-mentioned recent contributions on interest-based architectures still lack in one very important aspect: in these architectures, the underlying assumption is that no peer can have more than one resource type and this could be a very hard restriction practically. Therefore, in the present work, we have addressed this issue of generalizing pyramid tree based P2P architecture and have come up with effective solutions that allow a peer to possess multiple different resource types.

The organization of the paper is as follows. In Section 2, we talk briefly about some related preliminaries. Our contributions in the present paper appear in Sections 3 and 4. In Section 3, generalization of the architecture has been considered. In Section 4, effect of the generalization on the existing communication protocols [15, 22-23] has been considered. Section 5 draws the conclusion.

2 Preliminaries

In this section, we present some relevant results from our recent works on the Pyramid tree based P2P architecture [15, 21-23]. for interest-based peer-to-peer system. Residue Class based on modular arithmetic has been used to realize the overlay topology.

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 i^{th} node is the i^{th} 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, $a \equiv r \pmod n$}.

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 [15, 21].

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 [15].

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 [15]. 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 accommodate a very large number of possible resource types (if needed in the 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 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 an 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.

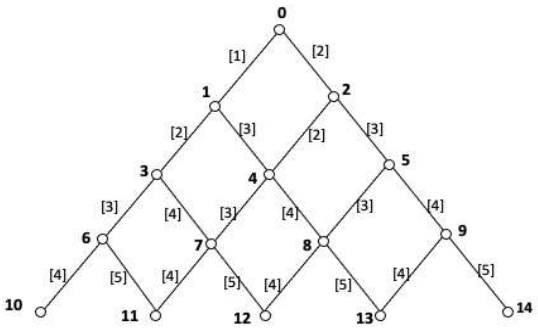

Figure 1: A complete pyramid tree with root 0

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 [21] 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 the root of this tree is node 13, whereas root of the tree of Figure 1 is 0.

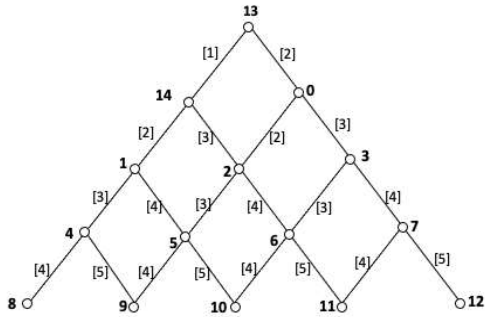

Figure 2: A complete pyramid tree with root 13

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. 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 Generalization of the Architecture

As mentioned earlier, in the architecture, it is assumed that no peer can have more than one resource type and this could be a very hard restriction practically. To overcome this restriction, we have come up with the concept of *Generalization* i.e., the architecture is generalized in such a way that a peer can have multiple resource types. Generalization of the Architecture needs to deal with two possible scenarios. Below we consider the two possible scenarios and state how to incorporate some necessary changes in the architecture in order to handle the two scenarios. Throughout our presentations, we shall interchangeably use the words 'node' and 'cluster-head'. So, a node on the tree is actually a cluster-head. These are all peers though. However, we strictly use the word 'peer' to represent members of a cluster only to avoid any possible confusion. In addition, we assume that 'resource with type k' and 'resource with code k' mean the same resource.

3.1 Peer with Multiple Existing Resource Types

Scenario 1: Let us consider a situation that in some cluster

C_i, its cluster-head C_i^h or a peer p in C_i wants data insertion of another existing resource type, say R_k in the network. Here data-insertion by a peer means the peer in question declares the possession of instances of another resource type that already exists in the system.

As mentioned earlier, peers in cluster C_k possess instances of the resource type R_k. Also, peer p in C_i already possesses some instances of the resource type R_i.

Solution*:* The solution for this scenario is as follows. The cluster-head C_i^h or peer p will now become a member of cluster C_k as well. So, it is understood that the IP address of C_i^h /p will be known to members of both the clusters C_i and C_k. It means that, in the overlay network, C_i^h /p will appear logically in both the clusters C_i and C_k, and will have direct logical connections to all member peers of clusters C_i and C_k. Therefore, it should be clear that our already reported intra- and inter-cluster data-lookup protocols [28] do not need any modification in this scenario. The same is true for broadcasting involving the cluster-heads in the tree. In addition, we have observed that the capacity-constrained broadcast and multicast protocols inside a cluster [20] in the tree need not be modified as well.

However, we observe that the existing inter-cluster data look-up protocol as well as the broadcast protocol involving all cluster-heads in the tree [15] will need some appropriate modifications to handle the second scenario. We shall present these in detail in the following sections. Before that we present the following solution to tackle the second scenario.

3.2 Existing Peers Declaring New Resource Types

Scenario 2: Consider a P2P interest-based pyramid tree system which has currently r distinct resource types, viz., R_0, R_1, R_2, … R_{r-1}. Assume that the respective resource codes are 0, 1, 2, …, (r-1). Without any loss of generality, let us assume a scenario where cluster-head C_i^h / a peer p in a cluster C_i wants a data insertion of a new resource type R_r currently not present in the network.

Solution*:* Solution lies in an appropriate modification of the table of information (*TOI*) maintained by each cluster-head. We know that in *TOI*, corresponding to each cluster-head there is an entry (tuple). For example, the tuple for some cluster-head C_i^h appears as <resource code (logical address) owned by peers in C_i^h, IP address of the cluster-head C_i^h >; note that in the architecture resource code and the logical address of a cluster-head are the same. That is, one denotes the other. It facilitates packet propagation in the tree. In short, we write the tuple as < Res. Code, IP (C_i^h) >. As new clusters are formed owing to peers joining with new resource types, the *TOI* grows dynamically and the newest joining cluster-head is assigned with the next largest logical address not yet used and hence its resource code also becomes the largest among all such existing codes. Therefore, this table remains sorted with respect to logical addresses of cluster-heads (i.e., with respect to the resource codes of the resources they possess).

Coming back to the second scenario, a new entry is made in the *TOI* corresponding to the new resource type R_r with resource code r. So currently this code r is the largest one present in the table. Based on if it is the cluster-head C_i^h / or a peer in cluster C_i that wants a data insertion of a new resource type R_r, in the newly entered tuple, the corresponding cluster-head will be either C_i^h or the peer p. That is, if it is C_i^h, it will now represent two different clusters corresponding to two different resource types i and r. So, it will have two different logical addresses i and r as well. Therefore, later any peer wishing to join with resource type r will join the cluster with logical address r. Effectively, C_i^h now will maintain two different clusters C_i and C_r, i.e., one with peers for resource code i and the other with peers with resource code r. It is clear that cluster-head in the second case with resource type r is now C_r^h which is actually C_i^h. In case it is the peer p in cluster C_i, peer p will maintain a cluster of peers with resource type r; thus, p will appear as a peer in Cluster C_i and will also appear as a cluster-head C_r^h with logical address r. Therefore, we have modified the *TOI* to include the relevant information of the new entry. Below we give an example to clear the idea further.

Observation 3. Generalization of the architecture may require some nodes of the tree represent multiple cluster-heads with the same IP address, but with different distinct resource types.

Example 1: Without any loss of generality let us consider a 3-level complete pyramid tree. Thus, the tree has six distinct resource types with respective resource codes as 0, 1, 2, 3, 4, 5. According to the structure of the tree node 0 is at level 1, nodes 1 and 2 at level 2, and nodes 3,4, and 5 are at level 3. Next, assume that cluster-head C_1^h declares that it has just possessed some instances of another new resource type with 6 as its code. Now, the tree becomes a 4-level incomplete tree with seven nodes (i.e., seven cluster-heads) with node 6 at level 4. Therefore, as explained above, *TOI* needs to be modified. Before and after the above declaration *TOI* appears as shown below in Figures 3a and 3b. We denote IP address of a node X as IP(X). 'Res. Code' is actually 'Resource Code'.

Note that in Figure 3b cluster-head C_1^h has appeared twice: once it represents a cluster-head with logical address 1 and next with logical address 6, appearing as C_6^h. That is, C_1^h now represents virtually two different clusters of peers C_1 and C_6, one with instances of resource type with code 1 and the other one with code 6. In effect, the 2nd appearance of C_1^h as C_6^h makes the tree incomplete with 7 nodes.

Note that if instead of the cluster-head C_1^h some peer, say p* in cluster C_i declares that it has just possessed another new resource type with 6 as its code, the entry for resource code 6 will become <6, IP(p*)> in Figure 3b. Hence, the new cluster-head p*(i.e., C_6^h) forms a cluster with peers willing to join with instances of resource type 6.

It may be noted that any inter-cluster query for some instance of the resource type 6 will be directed at either C_1^h or p* depending on the tuple corresponding to resource code 6 (Figure 3b).

Res. Code	IP address
0	IP(C_0^h)
1	IP(C_1^h)
2	IP(C_2^h)
3	IP(C_3^h)
4	IP(C_4^h)
5	IP(C_5^h)

Figure 3a: *TOI* before declaration

Res. Code	IP address
0	IP(C_0^h)
1	IP(C_1^h)
2	IP(C_2^h)
3	IP(C_3^h)
4	IP(C_4^h)
5	IP(C_5^h)
6	IP(C_6^h)

Figure 3b: *TOI* after declaration: IP(C_1^h) = IP(C_6^h)

4 Modification of Existing Inter-Cluster Data Look-Up and Broadcast Protocols

As pointed out earlier the existing inter-cluster data look-up and broadcast protocols (involving all cluster-heads in the tree) will need some appropriate modifications to handle only the second scenario. In this section we deal with this. We again emphasize that none of the two scenarios have any effect on the existing capacity-constrained broadcast and multicast protocols inside a cluster [22] i.e., as long as the communication is inside a cluster only, no related existing protocols need be modified.

4.1 Modified Inter-Cluster Data Look-Up Protocol

In the generalized protocol stated below, codes from line 2 to line 3 are added to handle the second scenario. This section of the total code deals with the situation when a peer represents multiple cluster-heads with each cluster having distinct resource types. In the architecture, any communication between a peer $p_i \in C_i$ and a peer $p_m \in C_m$ takes place only via the respective cluster-heads C_i^h and C_m^h. Without any loss of generality let a peer p_i^* ($\in C_i$) request for $<R_m, V^*>$. Note that peer p_i^* knows that $R_m \notin C_i$, because resource code used in cluster C_i is i. The protocol appears in Figure 4 below.

Protocol Generalized Inter-Data-Lookup

1. p_i^* sends a data lookup request for $<R_j, V^*>$ to its cluster-head C_i^h
2. *if* $IP(C_i^h) = IP(C_m^h)$ */ C_i^h checks in its TOI; same peer acts as cluster-heads for clusters C_i and C_m*
   if C_m^h *possesses* $<R_m, V^*>$
   C_m^h *unicasts* $<R_m, V^*>$ *to* p_i^*
   else
   C_m^h *broadcasts the request for* $<R_m, V^*>$ *in* C_m */ one hop communication*
   if $\exists p_m (\in C_m)$ *with* $<R_m, V^*>$
   p_m *unicasts* $<R_m, V^*>$ *to* p_i^*
   else
3. C_m^h *unicasts 'search fails' to* p_i^*
4. *else*
   C_i^h *determines the cluster-head* C_m^h *'s IP address from its TOI using* C_m^h *'s resource code / logical address of* C_m^h = *resource code of* R_m = m
   C_i^h *unicasts the request to* C_m^h
   if C_m^h *possesses* $<R_m, V^*>$
   C_m^h *unicasts* $<R_m, V^*>$ *to* p_i^*
   else
   C_m^h *broadcasts the request for* $<R_m, V^*>$ *in* C_m */ one hop communication*
   if $\exists p_m (\in C_m)$ *with* $<R_m, V^*>$
   p_m *unicasts* $<R_m, V^*>$ *to* p_i^*
   else
   C_m^h *unicasts 'search fails' to* p_i^*

Figure 4: Modified generalized inter-cluster data-lookup protocol

As in Observation 2 earlier, search latency for modified inter-cluster data look-up approach remains bounded by the diameter of the tree and is independent of the total number of peers present in the system.

4.2 Modified Broadcast Protocol

In the context of broadcasting, it may be noted that, in general, inter-cluster broadcast involves always intra-cluster broadcast as well, with the exception when a cluster-head wants to update some control information (ex. broadcasting of updated *TOI* by the root of tree) 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.

An interesting observation is that if the root is node 0 (logical address), even an incomplete tree always remains a connected one; on the other hand, for any other cluster-head as root, an incomplete tree may not remain connected. To explain the idea briefly and clearly, consider the complete tree of Figure 1. Its root is node 0. If root changes to some other node, say node 13, the tree still remains a complete one as is shown in Figure 2. This property of the architecture arising from 'virtual neighbors' has been discussed in detail earlier. Now assume that the tree in Figure 1 does not have node 14, i.e., cluster 14 is yet to be

formed. So the tree is incomplete, yet it is connected. In this situation, if node 0 broadcasts some packets, all other nodes will receive copies. Now assume that node 13 is the broadcaster and node 13 is assumed to be the root. It is seen that its immediate neighbor on its left branch should be node 14 (Remark 1); however node 14 does not exist in the tree as assumed above. So propagation of any broadcast packet along the left branch cannot take place based on the Broadcast-Complete protocol [15]. Therefore, broadcast fails. However, if node 13 unicasts its packets first to node 0 (root) which will then act as the pseudo broadcaster on behalf of node 13, all nodes get copies because the tree remains connected with node 0 as its root. This is actually the idea used in the Broadcast-Incomplete protocol [15]. This has led us to consider modifying only the existing Broadcast-Incomplete protocol to appropriately handle the second scenario.

An informal sketch of the modified incomplete broadcast protocol

Step 1: A broadcast source node X will unicast its packets to the root node 0.

Step 2: Root 0 sends packets to its neighbors on left and right branches.

Step 3: Each receiving node on the left branch sends packets to its neighbor on this branch till a receiving node is a leaf node.

Step 4a: 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 4b: 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 4c*:* Propagation along the i^{th} left branch continues as in Step 3.

In the above informal sketch, 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-incomplete protocol. That is, node 0 will act as the pseudo broadcast source (like in CBT multicast [2] the core is the pseudo multicast source). Hence, the tree will remain connected with node 0 as its root even if the original tree is an incomplete one. This justifies our consideration to modify only the existing broadcast-incomplete protocol.

Since node X is a part of the tree, 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 further 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 formal presentation of the protocol appears in Figure 4.

It may be noted that instead of using the left branches originating at nodes on the right branch (as in step 4 above), the protocol can use the right branch of the root and all right branches emanating from the nodes on the left branch of the root. In this way, it will also generate only one duplicate packet per packet broadcast as well. 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.
Address (X):	logical address of node X
IP address of X:	IP(X)

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 has already been reported in detail in [23]. Therefore, in the following algorithm by 'node 0' it means the current root.

We have modified the existing broadcast-incomplete protocol [15] to incorporate the solution for scenario 2 as discussed in the previous section. The modified portion appears on lines 12 to 14 (Figure 5) and it resolves the issue raised in scenario 2. This small piece of code is crucial in the modified protocol. Below, we have explained its importance considering again Example 1.

Initially peers in cluster C_1 have instances of resource type with code 1. Assume that later cluster-head C_1^h declares that it has just possessed another new resource type with 6 as its code. Therefore, now cluster-head C_1^h represents virtually two clusters, one consisting of peers possessing resource type with code 1 and the other with code 2 (refer to Figure 3b). Now without any loss of generality we shall consider the following three possible cases.

Case 1. Assume that node 2 (i.e., cluster-head C_2^h) is the source of broadcast (appeared as X in the protocol). It starts unicasting its broadcast packets to the root node 0 which then broadcasts the packets to all nodes (cluster-heads) following the

Protocol Generalized-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
    if IP(C_i^h) ≠ IP(X)
    C_i^h keeps a copy
13. **else** C_i^h does not keep a copy / C_i^h has multiple distinct resource types; already has copy of every packet since it is the source X
14. **else** does not keep a copy
15. **if** $N_h = 1$ / *it is a leaf level node*
16. stops forwarding
17. **else**
18. **if** flag = L in the received packet
19. **if** [(address (C_i^h) + (Inc + 1)) mod n] ≤ N_{max} / *N_{max} is the largest current logical address in the tree*
20. $N_h = N_h - 1$ / *build a new BP_L packet*
21. Inc = Inc+1 / *n = number of residue classes = number of distinct resource types*
22. new BP_L packet = < N_h, Inc, L, Info >
23. 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*
24. **else**
25. stops forwarding / no such address exists; tree is incomplete

26. **else** */ flag is R and C_i^h are on the right branch of the broadcast source*
27. **if** [(address (C_i^h) + (Inc)) mod n] ≤ N_{max}
28. Inc = Inc in the received BP_R packet */ build a new BP_L packet*
29. $N_h = N_h$-1
30. flag = L
31. new BP_L packet = < N_h, Inc, L, Info >
32. 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*
33. **else**
34. stops forwarding / no such address exists; tree is incomplete
35. **if** [(address (C_i^h) + (Inc + 1)) mod n] ≤ N_{max}
36. $N_h = N_h$-1 */ build a new BP_R packet*
37. Inc = Inc+1
38. flag = R
39. new BP_R packet = < N_h, Inc, R, Info > */ propagation along the right branch of C_i^h continues*
40. C_i^h forwards to the node with address, [(address (C_i^h) + Inc) mod n]
41. **else** / no such address exists; tree is incomplete
42. stops forwarding

Figure 5: Protocol Generalized- Broadcast-Incomplete

protocol. However, it is observed that every node receives exactly one copy of each packet except node 2, because node 2 will also receive copies of its already unicasted packets from the root node 0 because of broadcasting. So, it discards the received copies. This has appeared in the protocol as stated in *italics* below.

12. ***if*** $C_i^h \neq X$
    if IP(C_i^h) ≠ IP(X)
    C_i^h keeps a copy
13. **else** C_i^h does not keep a copy
14. ***else*** *does not keep a copy*

Case 2. Assume that node 1 is the source, i.e., C_1^h = X. Cluster-head C_1^h has appeared twice on the tree once with logical address 1 and another with logical address 6. So, C_1^h itself being the source of broadcast (as node 1), will discard the packets when it receives as node 6 from root node 0 because of broadcasting. This has appeared in the protocol as stated in *italics* below.

12. ***if*** $C_i^h \neq X$
    if IP(C_i^h) ≠ IP(X)
    C_i^h keeps a copy
13. ***else*** *C_i^h does not keep a copy*
14. **else** does not keep a copy

Case 3. If logical addresses of the broadcast node X and a receiving node are different, still there is a chance that both nodes are actually the same one (Case 2 above). However, if

not, their IP addresses will differ and the receiving node will keep copies of the received packets. This has appeared in the protocol as stated in *italics* below.

12. ***if*** $C_i^h \neq X$

if $IP(C_i^h) \neq IP(X)$

C_i^h *keeps a copy*

else C_i^h does not keep a copy

else does not keep a copy

Theorem 1. The protocol generates exactly one extra copy per packet broadcast.

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 the right branch. Propagation stops when a receiving node is a leaf node. Therefore, all nodes receive the broadcast information.

Besides, 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, 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-incomplete protocol [15] complexity is dependent only on the number of the distinct resource types n present in the system, which in turn determines the value of the number of levels d of the tree.

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.

Remark 6. The proposed method to generalize the architecture is remarkably simple and efficient, and it does not affect the existing intra-cluster data look-up protocol; it affects the existing inter-cluster data look-up and the broadcast protocol is arguably a minimal way (as is observed in the modified portion of the original pseudo code in case of the broadcast protocol).

5 Conclusion

Authors, in recent studies, have exploited the architectural properties of the Pyramid tree P2P network to design different communication protocols with reasonably low search latency. However, these recent contributions still lack in one very important aspect (like other existing interest-based architectures): in the architecture, it is assumed that no peer can have more than one resource type and this could be a very hard restriction practically. In the present work, we have addressed this issue of generalizing the architecture and have come up with effective solutions. Effect on the architecture due to generalization is just reflected in the modified 'Table of Information' which is actually consulted by the various protocols for data/query propagation. We have shown that generalization has no effect on the existing intra-cluster data look-up protocol. Only the existing inter-cluster data look up and the broadcast protocols need to be modified. The two modified protocols have the same low bandwidth requirements and look-up complexities as those of the already existing ones.

As a continuation of our research, we are now working on designing protocols for secured communication in the generalized architecture.

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] Tony 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. 407-418, August 25-29, 2003.

[4] 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 System*s, 18(9)1188-1201, Sep. 2007.

[5] Wen-Tsuen Chen, Chi-Hong Chao and Jeng-Long Chiang, "An Interested-based Architecture for Peer-to-Peer Network Systems," 20th International Conference on Advanced Information Networking and Applications - Volume 1 (AINA'06), Vienna, pp. 707-712, 2006, doi: 10.1109/AINA.2006.93.

[6] Jie Cheng and Ryder 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," Proc. IEEE Infocom 2003, San Francisco, USA, pp. 1250-1260, March 30-April 1 2003.

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

[9] Bidyut Gupta, Nick Rahimi, Shahram Rahimi, and Ashraf Alyanbaawi, "Efficient Data Lookup in Non- DHT Based Low Diameter Structured P2P Network," Proc. IEEE 15th Int. Conf. Industrial Informatics (IEEE INDIN), Emden, Germany, pp.143-148, July 2017.

[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, pp. 362-365, 2010, doi: 10.1109/ICMeCG.2010.80.

[11] Mujtaba Khambatti, Kyung Ryu, and Partha 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, E. K. Lua, and X. Zhou, " Hierarchical Peer-to-Peer Networks using Lightweight SuperPeer Topologies," Proc. IEEE Symp. Computers and Communications, pp. 944-950, 2005.

[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] Koushik Maddali, Indranil Roy, Swathi Kaluvakuri, Bidyut Gupta, Narayan Debnath, "Design of Broadcast Protocols for Non DHT-Based Pyramid Tree P2P Architecture," *IJCA*, 28(4)193-203, December 2021.

[16] Z. Peng, Z. Duan, J.Jun Qi, Y. Cao, and E. Lv, "HP2P: A Hybrid Hierarchical P2P Network," Proc. Intl. Conf. Digital Society, pp. 18-24, 2007.

[17] N. Rahimi, K. Sinha, B. Gupta, and S. Rahimi, "LDEPTH: A Low Diameter Hierarchical P2P Network Architecture," Proc. IEEE 14th Int. Conf. on Industrial Informatics (IEEE INDIN), Poitiers, France, pp. 832-837, July 2016.

[18] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A Scalable Content-Addressable Network, CAN," *ACM*, 31(4):161-172, 2001.

[19] Arnold L. Rosenberg, "The Diogenes Approach to Testable Fault-Tolerant Arrays of Processors," *IEEE Tran. Computers*, c-32(10):902-910, Oct. 1983.

[20] 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.

[21] Indranil Roy, Bidyut Gupta, Banafsheh Rekabdar, and Henry Hexmoor, "A Novel Approach Toward Designing A Non-DHT Based Structured P2P Network Architecture," (Proceedings of 32nd Int. Conf. Computer Applications in Industry and Engineering), EPiC Series in Computing, 63(2019):121-129, 2019.

[22] Indranil Roy, Swathi Kaluvakuri, Koushik Maddali, Abdullah Aydeger, Bidyut Gupta, and Narayan Debnath, "Capacity Constrained Broadcast and Multicast Protocols for Clusters in Pyramid Tree-Based Structured P2P Network," *IJCA*, 28(3):122-131, Sep. 2021.

[23] Indranil Roy, Swathi Kaluvakuri, Koushik Maddali, Ziping Liu, and Bidyut Gupta, "Efficient Communication Protocols for Non DHT-Based Pyramid Tree P2P Architecture," *WSEAS Transactions on Computers*, 20:108-125, July 2021 (Invited paper).

[24] 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.

[25] 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, 1 June 2015, doi: 10.1109/TPDS.2014. 2327033.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

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

Indranil Roy is an Assistant Professor in the Department of Computer Science at the Southeast Missouri State University. He received his MS and Ph.D. degrees in Computer Science from Southern Illinois University, Carbondale in 2018 and 2022, respectively. His current research interest includes the design of architecture and communication protocols for structured peer-to-peer overlay networks, security in overlay networks, and Blockchain.

Nick Rahimi (photo not available) is the Director of Cyber Innovation Lab and an Assistant Professor at the School of Computing Sciences & Computer Engineering of the University of Southern Mississippi (USM). Dr. Rahimi obtained two

Bachelor of Science degrees in Computer Software Engineering and Information Systems Technologies with a concentration in Cybersecurity and received his Master and Ph.D. degrees in Computer Science from Southern Illinois University (SIU). His research interest lies in the area of cybersecurity, blockchain, cryptography, internet anti-censorship, machine learning in cybersecurity, distributed systems, and decentralized networks. Prior to joining USM, Dr. Rahimi was a tenure track Assistant Professor at SIU for 2 years and Southeast Missouri State University (SEMO) for one year respectively. Moreover, he has over eight years of experience in industry as a software engineer and team leader.

Ziping Liu received her PhD in Engineering Science from Southern Illinois University at Carbondale in 1999 and began her computing career at Motorola, where she developed software for mobile phones. Currently, she is a Professor of Computer Science at Southeast Missouri State University, where she has been teaching since 2001. Her research interests encompass a wide range of topics, including machine learning, cloud computing, secured software design, wireless ad hoc networks and sensor networks, distributed computing and game development.

Bidyut Gupta (photo not available) is currently a Professor of Computer Science at the School of Computing, Southern Illinois University at Carbondale. His research interests include fault tolerant distributed computing, design of P2P network architectures with low latency communication protocols, fog computing and its applications, and high latency networks. He is a Senior member of IEEE and ISCA.

Narayan C. Debnath (photo not available) 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. Formerly, Dr. Debnath served as a Full Professor of Computer Science at Winona State University, Minnesota, USA for 28 years, and the elected Chairperson of the Computer Science Department at Winona State University for 7 years. Dr. Debnath has been the Director of the International Society for Computers and their Applications (ISCA), USA since 2014.

Professor Debnath made significant contributions in teaching, research, and services across the academic and professional communities. He has made original research contributions in software engineering, artificial intelligence and applications, and information science, technology and engineering. He is an author or co-author of over 500 research paper publications in numerous refereed journals and conference proceedings in Computer Science, Information Science, Information Technology, System Sciences, Mathematics, and Electrical Engineering. He is also an author of over 15 books published by well-known international publishers including Elsevier, CRC, Wiley, Bentham Science, River Publishing, and Springer.

Dr. Debnath has made numerous teaching, research and invited keynote presentations at various international conferences, industries, and teaching and research institutions in Africa, Asia, Australia, Europe, North America, and South America. He has been a visiting professor at universities in Argentina, China, India, Sudan, and Taiwan. He has been maintaining an active research and professional collaborations with many universities, faculty, scholars, professionals and practitioners across the globe. Dr. Debnath is an active member of the IEEE, IEEE Computer Society, and a Senior Member of the International Society for Computers and their Applications (ISCA), USA.