Efficient Data Lookup in Non-DHT Based Low Diameter Structured P2P Network

Bidyut Gupta, Nick Rahimi, Shahram Rahimi, and Ashraf Alyanbaawi
Department of Computer Science
Southern Illinois University
Carbondale, IL, USA
{bidyut, nick ,rahimi}@cs.siu.edu and ashraf@siu.edu

***Abstract* — In this paper, we have considered a recently reported non-DHT based structured P2P system. The architecture is based on Linear Diophantine Equation (LDE) and it is an interest-based system; it offers very efficient data lookup. However, the architecture is restricted in that a peer cannot possess more than one distinct resource type. This may reduce the scope of its application. In this paper, we have extended the work by considering a generalization of the architecture, that is, a peer can possess multiple distinct resource types. We have proposed an efficient data lookup algorithm with time complexity bounded by (2+r/2); r is the number of distinct resource types. We have discussed about an alternative lookup scheme that needs constant number of hops and constant number of message exchanges. Besides, churn handling and ring maintenance have been shown to be very efficient.**

Keywords — P2P network; structured; Linear Diophantine equation; network diameter; data lookup; churn

I. INTRODUCTION

Peer-to-Peer (P2P) overlay networks are widely used in distributed systems. There are two classes of such networks: unstructured and structured ones. In unstructured systems [2] peers are organized into arbitrary topology. Flooding is usually used 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. Unstructured networks have excessive lookup costs 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) [4], [5], [9]. Such overlay architectures can offer efficient, flexible, and robust service [3] - [5], [7], [8],

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 hybrid systems [1], [6], [10] - [12]; their objective being incorporation of the advantages of both structured and unstructured architectures. However, these works have their own pros and cons.

We have earlier proposed a new hierarchical architecture [13], [14] in which at each level of the hierarchy existing networks are all structured. We have used Linear Diophantine Equation (LDE) as the mathematical base to realize the architecture. Note that most structured approaches use DHTs to realize their architectures. Use of Linear Diophantine Equation in designing P2P architecture is a completely new idea. We have explored the many different possible advantages that can be fetched using LDEs; some of these advantages include efficient handling of data look-up, node (peer) join/leave, anonymity, load balancing among peers, to name a few; besides achieving fault-tolerance is reasonably simple. We have shown that the complexity involved in maintaining different data structures is much less than that involved in the maintenance of DHTs. On several points, LDE-based overlay architecture can outperform DHT-based ones. The proposed architecture has considered interest-based P2P systems [6], [15], [16]. The rationale behind this choice is that users sharing common interests are likely to share similar contents, and therefore searches for a particular type of content is more efficient if peers likely to store that content type are neighbors [17].

II. PRELIMINARIES AND PROBLEM FORMULATION

Some of the preliminary ideas of the hierarchical P2P architecture proposed in [13], [14] have been considered in this paper. For the sake of completeness, we reproduce here from [13] some of the notations and the basic idea of using Linear Diophantine equations for the purpose of generating the logical addresses of the nodes (peers) of the overlay network.

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. 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'. In the model for interest-based P2P systems [13], we assume that no two peers with the same resource type R_i can have the same tuple [13]; that is, two peers with the same resource type R_i must have tuples $<R_i, V'>$ and $<R_i, V''>$ such that $V' \neq V''$. In [13] and [14], the assumption is that no peer can have more than one resource type.

We define the following. Let S be the set of all peers in a peer-to-peer system. Then $S = \{P^{Ri}\}$, $0 \leq i \leq r-1$. Here P^{Ri}

denotes the subset consisting of all peers with the same resource type R_i and no two peers in P^{Ri} have the same value for R_i and the number of distinct resource types present in the system is r. Also for each subset P^{Ri}, P_i is the first peer among the peers in P^{Ri} to join the system. We now describe the P2P architecture [13] suitable for interest-based peer-to-peer system

A. Two Level Hierarchy

In [13] we have proposed a two level overlay architecture and at each level, networks of peers are all structured. It is explained in detail below.

1) At level-1, we have a ring network consisting of only the peers P_i ($0 \le i \le r-1$). Therefore, number of peers on the ring is r, the number of distinct resource types. This ring network is used for efficient data lookup and so it is called as transit network.
2) At level-2, there are r numbers of completely connected networks of peers. Each such network, say N_i is formed by the peers of the subset P^{Ri}, ($0 \le i \le r-1$), such that all peers (ϵP^{Ri}) are directly connected (logically) to each other, resulting in the network diameter of 1. Each such N_i is connected to the transit ring network via the peer P_i. Peer P_i acts as the group-head of network N_i. From now on network N_i will be referred to as group$_i$ (in short as G_i) with P_i as its group-head. The architecture is shown in Fig. 1.
3) Each node in the transit ring network maintains a *global resource table* (GRT) that consists of tuples of the form *<Resource Type, Resource Code, Group Head Logical Address, Group Head IP address>*, where *Group Head Logical Address* refers to the logical address assigned to a node by our proposed architecture.

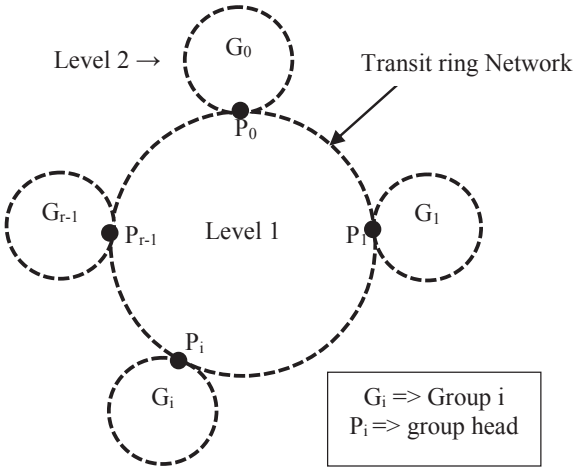

Fig. 1 A two-level structured architecture with distinct resource types

B. Linear Diophantine Equation (LDE) and Its Solutions

Let us consider the LDE as stated below.

$$an \equiv b \pmod{c}, \quad a, b, \text{ and } c \text{ are integers.} \qquad (1)$$

Let $d \mid b$, where $d = \gcd(a,c)$. It means that (1) has d mutually incongruent solutions.

The above equation can also be stated as

$$an + (-c)k = b, \quad k \text{ is an integer.} \qquad (2)$$

Each solution of Equ. (1) (& hence of (2) as well) has the form: $n = n_0 + ct/d$, $k = k_0 + at/d$

where n_0 and k_0 constitute one specific solution and t is any integer.

Among the different values of n described by $n = n_0 + ct/d$, we note that the d values

n_o, $n = n_0 + c/d$, $n = n_0 + 2c/d$, --- , $n = n_0 + (d-1)c/d$

are all mutually incongruent modulo c, because the absolute difference between any two of them is less than c.

Also the values of a, b, and c can be so chosen as to make d very large whenever needed. Observe that there are infinite other solutions which are congruent to each of the d solutions. For example, all solutions of the form

$(n_o + mc)$, m is an integer, are mutually congruent. Similarly all solutions of the form $[(n_0 + c/d) + mc]$ are mutually congruent.

C. Implementation of the Architecture

Assume that in an interest-based P2P system there are r distinct resource types ($r \le d$). That is, a maximum of d resource types can be present. Note that this is not a restriction, because d can be set to an extremely large value a priori by choosing an appropriate LDE. Consider the set of all peers in the system given as

$S = \{P^{Ri}\}$, $0 \le i \le r-1$.

As mentioned earlier, for each subset P^{Ri} (i.e. group Gi) peer Pi is the first peer with resource type Ri to join the system. Now we use the mutually incongruent solutions of a given LDE to define the architecture as follows.

The ring network (Fig. 1) at level-1 will consist of all such P_i's, for $0 \le i \le r-1$, and $r \le d$, such that

a) Each P_i will be assigned the logical address (n_0 + i.c/d). Note that (n_0 + i.c/d) is the i^{th} mutually incongruent solution where $0 \le i \le d-1$.
b) The transit network is a ring by default, because of modulo operation. Two peers in the ring network are neighbors if their assigned addresses differ by c/d, with the exception that the first peer P_0 and the last peer P_{r-1} will be considered as neighbors even though their addresses differ by (r-1).c/d. This structure has made the joining of new peers with new resource types very simple.
c) Resource type R_i possessed by peers in G_i is assigned the code (n_0 + i.c/d) which is also the logical address of the group-head P_i of group G_i.
d) Diameter of the ring network can be at most d/2.

At level-2 all peers having the same resource type R_i will form the group G_i (i.e. the subset P^{Ri}). Only the group-head P_i is connected to the transit ring network. Observe that any communication between any two groups G_i and G_j takes place via the respective group-heads P_i and P_j. Peers in G_i will be assigned with the addresses

$$[(n_0 + i.c/d)+ m.c], \text{ for } m = 0, 1, 2, \ldots \quad (3)$$

Note that m = 0 corresponds to the address of group-head P_i of G_i.

Observation 1. All addresses in G_i are mutually congruent solutions for a given i.

Observation 2. *Congruence Relation* is reflexive, symmetric, and transitive. Therefore it can be concluded that all peers in a group G_i are *directly connected (logically*) to each other forming a network of *diameter 1* only.

D. Problem Formulation

In the above architecture, we have assumed that no peer can have more than one resource type. It may become a hard restriction in practice. Therefore, to overcome this restriction, in the present work, we have considered *Generalization* of the architecture; that is, a peer can have multiple different resource types. To implement the idea we have redesigned the peer joining processes (Section III), the inter-group data lookup algorithm (Section IV). In Section V, we have considered concurrent joins and leaves, followed by the maintenance of the transit ring network in Section VI.

III. GENERALIZATION OF THE ARCHITECTURE

To describe the situation, let us consider that in $group_i$ the group-head P_i or a peer p (ϵ G_i) wants data insertion in the system of another existing resource type R_k; note that R_k exists in $group_k$ and P_i /p already possesses R_i.

The solution works as follows. Peer P_i /p will become a member of $group_k$ as well. That is, the IP address of P_i /p will be known to the members of both $group_i$ and $group_k$. Logically it means that in the overlay network, P_i /p will be directly connected to all members of both $group_i$ and $group_k$. Below we state its implementation.

A. Peer with Multiple Existing Resource Types

To describe the situation, let us consider that in $group_i$ the group-head P_i or a peer p (ϵ G_i) wants data insertion in the system of another existing resource type R_k; note that R_k exists in $group_k$ and P_i /p already possesses R_i.

The solution works as follows. Peer P_i /p will become a member of $group_k$ as well. That is, the members of both $group_i$ and $group_k$ will know the IP address of P_i /p. Logically, it means that in the overlay network, P_i /p will be directly connected to all members of both $group_i$ and $group_k$. Algorithm1 states its implementation. It is shown in Fig. 2.

Time complexity of Algorithm 1 is bounded by (1+ r/2), r being the number of distinct resource types. Data insertion for more existing resource types can be done similarly.

B. Existing Peers Declaring New Resource Types

To start with, let us assume that the P2P system has S number of distinct resource types, viz., R_0, R_1, R_2, … R_{s-1} . Without any loss of generality, let peer P_i /p in $group_i$ wants a data insertion for a new resource type R_s. Then following the way the transit ring is constructed, peer P_i /p will become the group-head of the newly created $group_s$ possessing resource type R_s. As the recent group-head P_s, location of P_i /p on the ring is now between P_{s-1} and P_0. So, if it is P_i, peer P_i will appear (logically) twice as group-heads on the ring for $group_i$ and $group_s$. If it is peer p, it will appear once as the group-head of groups and once as a member of $group_i$. Note that R_s will have the code (n_0 + sc/d) and it will also be another logical address for P_i /p. Now, P_i /p will ask the group-heads to update their global resource tables by including R_s and its code (n_0 + sc/d), along with the IP address of P_i /p. For implementation, P_i /p will now have another set of pointers pointing to its new neighbors, P_{s-1} and P_0. Group-heads P_{s-1} now changes its right neighbor from P_0 to P_s and group-head P_0 changes its left neighbor from P_{s-1} to P_s; they adjust their pointers accordingly.

To guard against group-head crash or leave, later when more peers join this group, P_i /p will store the IP addresses of P_{s-1} and P_0 in the peer with the next address [(n_0 + sc/d) + c]. We will elaborate further on fault-tolerance in Section VI.

Next, we consider data look up in the generalized structure.

IV. DATA LOOK-UP

We consider that a peer P_i is also the group-head P_s of $group_s$. Our proposed approach works as well if P_i possesses any number of distinct resource types. We assume that the system has r distinct resource types. In Fig. 3 we state the algorithm for data lookup inside a group.

1 Data insertion request for R_k from P_i /p is forwarded along the transit ring from group-head P_i to P_k *// maximum r/2 hops*
2 P_k assigns to P_i /p the next available address, not yet assigned in $group_k$.
// the address is of the form [(n_0 + kc/d) + yc], y is an integer

3 a. P_k broadcasts the address of P_i /p in $group_k$ *// P_i /p is the new member of $group_k$; 1-hop communication*
b. each $group_k$ member updates its list of neighbors
4 P_k unicasts a copy of neighbor list to P_i /p *// P_i /p is now a member of G_k*

Fig. 2 Algorithm 1: Data insertion for multiple existing resource types

A. Intra Group Lookup

1 node p_a (ϵ G_i) broadcasts in G_i for < R_i, V_b >
// one-hop communication since G_i is a complete graph

2 **if** p_b with < R_i, V_b > **then**
3 node p_b unicasts < R_i, V_b > to node p_a
4 **else**
5 search for <R_i, V_b > fails
6 **end**

Fig. 3 Algorithm 2: Intra-Group-Lookup

B. Inter Group Lookup

In our proposed architecture, any inter group communication involves travelling along the transit ring. Without any loss of generality let a peer p_a in G_i request for a resource < R_j, V* >. The following algorithm answers the query. In order to locate resource R_j, a search along the transit ring network is required. The algorithm for inter group lookup is presented in Fig. 4.

However, in our proposed architecture, number of peers on the ring is the number of distinct resource types r and it has been observed that the number of peers in most P2P networks is too large compared to the number of distinct resource types. Therefore, such search on the ring in our proposed architecture appears to be quite practical.

Another point to note is that use of the same logical address to denote a resource type and the corresponding group-head has not only made the search process simple and efficient, it also makes it feasible for every group-head to maintain the address of every other group-head in the transit network. This has two significant advantages:

1. Following Algorithm 3 (Fig. 4), the time complexity is bounded by (2 + r/2), because maximum number of hops required per any resource search is (2 + r/2), where r is the number of distinct resource types. Note that r « n, where n is the total number of peers in the system.
2. As an alternative resource lookup process, using the GRT a group-head P_i can directly unicast a message to any other group-head P_j avoiding any communication along the transit ring network. In this way, the lookup process will need a constant number of hops and a constant number of message exchanges.

In the following table, we have presented the complexity of our data lookup approach along with those of some other noteworthy structured approaches.

Table 1: Data Lookup Complexity Comparison

	CAN	**Chord**	**Pastry**	**Our Work**
Architecture	Structured P2P Overlay	Structured P2P Overlay	Structured P2P Overlay	Interest based, Two-level Structured Hierarchical
Lookup Protocol	{key, value} pairs to map a point P in the coordinate space using uniform hash function.	Matching key and NodeID.	Matching key and prefix in NodeID.	Inter-Group: Routing through Group-heads Intra-group: Complete Graph
Parameters	*N*-number of peers in network *d*-number of dimensions.	*N*-number of peers in network.	*N*-number of peers in network b-number of bits ($B = 2^b$) used for the base of the chosen identifier.	*r* - Number of distinct resource types. *N*-number of peers in network. r << N
Lookup Performance	$O(d\ N^{1/d})$	$O(\log N)$	$O(\log B^N)$	Inter-Group: $O(\log r)$ Intra-group: *One hop*

```
1 p_a (ϵ G_i) unicasts request for < R_j, V*> to group-head P_i
2 if P_i is also the group-head (P_j) for resource type R_j
3       if P_i (as P_j) possesses < R_j, V*> then
4             P_i (as P_j) unicasts < R_j, V*> to p_a
5       else
6             P_i (as P_j) executes Algorithm 2 in G_j
7 else
8       P_i determines resource <R_j, V*> group-head P_j's address code from GRT
          // address code of P_j = resource code of R_j = n_0 + j (c/d)
9       P_i computes h ← | (n_0 + i (c/d)) – (n_0 + j (c/d)) |
          // looking for minimum no. of hops along the transit ring
10      if h > r/2 then
11            P_i forwards the request along with the IP address of p_a to its predecessor P_i-1
12      else
13            P_i forwards the request along with the IP address of p_a to its successor P_i+1
14 end
15 if an intermediate group-head P_k is also the group-head for resource type R_j then
16      if P_k (as P_j) possesses < R_j, V*> then
17            P_k (as P_j) unicasts < R_j, V*> to p_a
18      else
19            P_k (as P_j) executes Algorithm 2 in G_j as its group-head P_j
20 else
21      each intermediate group-head P_k forwards the request until the request arrives at P_j
22      if P_j possesses <R_j, V*> then
23            P_j unicasts <R_j, V*> to p_a
24      else
25            P_j executes Algorithm 2 in G_j
26 end
```

Fig. 4 Algorithm 3: Inter-Group-Lookup

V. JOINS AND LEAVES

We assume that a well-known server keeps a copy of the GRT. When a new node (peer) wishes to join the system, it contacts the server. If the request to join is for an existing resource type, say R_i, the server sends the IP address of the group-head P_i to the node. If the request is for a new resource type, the server sends the IP address of the group-head P_0. Therefore, in our design the server plays a small but very important role related to load sharing by group-heads. All that is needed is when the GRT is updated by the group-heads, a copy is sent to the server. By virtue of its construction, the GRT remains sorted by default and in an ascending order of the Group-heads' logical addresses; so determining the exact group-head is $O(\log r)$. Note that in [14] server always sends the address of P_0 irrespective of the nature of a request, be it an existing or a new resource type; it would always increase the load on P_0 as all the requests are directed at it.

We now briefly discuss now the different possible situations of joining and leaving of peers.

A. Concurrent Joins

As pointed out earlier, a peer p either can join an existing group, or can form a new group with the group-head being the peer itself. In the former case, since nodes in a group are directly connected to each other, hence joining a group means forming a logical link between the peer p and each node in the group. We have presented the procedure in [14]. If multiple peers join the same group, say G_i, the join requests are queued at the group head P_i and are served on FCFS basis. Observe that joining multiple groups can take place concurrently, because joining one group is unrelated to joining other groups.

In case it is a new resource type R_s, the joining peer contacts P_0 that is the group-head of the very first group formed in the system [14]. Multiple such requests eventually arrive at P_0 and P_0 serves the requests on FCFS basis. We can handle insertion of multiple new resource types by the same peer in a similar way. Note that in the proposed architecture joining of any new resource type always takes place between the recent and the first groups [14]; this feature makes such joining localized to a single position on the ring; thereby making the joining process much simpler compared to existing related approaches [5] – [8]. It is obvious that the above-

mentioned two kinds of joins can take place simultaneously, because one involves existing groups and the other is about the formation of new groups.

B. Concurrent Leaves

We assume that any two directly connected peers in a group or along the transit ring exchange periodic hello packets. Whether it is a graceful leaving or abrupt leaving (crash) absence of a hello packet from a neighboring peer is interpreted as the peer being unreachable (not alive). That is, we do not differentiate between the above-mentioned two types of leaving. In effect, the logical link information about the leaving peer is deleted from the routing table of each peer not receiving the hello packet. Therefore, concurrent such leavings whether taking place in the same group or in multiple groups amounts to the deletion of the corresponding link information in the routing tables of the concerned non-leaving peers only. Of course, a non-leaving peer sequentially deletes multiple link information in case multiple peers leave the same group. Note that handling of single group-head crash has been discussed in [14]. Multiple group heads' leaving is considered in the following section.

C. Concurrent Joins and Leaves

Observe that 'concurrent joins and leaves' means that addition and deletion of logical links taking place concurrently. If a peer is involved in both actions, it will do so sequentially on FCFS basis; otherwise, different peers can execute these two operations concurrently in the system.

VI. RING MAINTENANCE

In our earlier work [14] we presented an approach to handle single group-head crash or leave. In the present work, we will consider multiple group-heads leaving simultaneously and we will show that the ring will remain connected in such situations. The approach works as follows.

Let us consider the peer P_r, the group-head of group G_r. The logical address of P_r is $(n_0 + r.c/d)$. Assume that peers $p^r{}_1$ and $p^r{}_2$ in G_r have the next two addresses followed by the group-head's address and these are $[(n_0 + rc/d) + c]$ and $[(n_0 + rc/d) + 2c]$ respectively. In [14], $p^r{}_1$ acts as the secondary group-head for group G_r to guard against the primary group-head leaving and we have considered that during the formation of this group, P_r stores in $p^r{}_1$ the addresses of its neighboring group-heads P_{r-1} and P_{r+1} along with a copy of the GRT. In the event of P_r leaving, $p^r{}_1$ becomes the new primary group-head and its communication connectivity with P_{r-1} and P_{r+1} remains intact. It also means that $p^r{}_2$ will now act as the new secondary group-head for group G_r. The new primary group-head $p^r{}_1$ will save the neighbors addresses, i.e. the addresses of P_{r-1} and P_{r+1} in $p^r{}_2$ and broadcasts to other group-heads to update their GRTs to reflect that $p^r{}_1$ is now the group-head of G_r. One noteworthy point is that peer $p^r{}_1$ does not need to inform the other peers in G_r about itself being the new group-head. The reason is simple and interesting. The way of construction of the routing table of a peer as a new peer joins group G_r ensures that the routing-table remains sorted by default and in an ascending order of the peers' logical addresses in the group. Therefore, the entries are same in each routing table and each peer knows that the peer with the lowest logical address is the current group-head.

However how can the connectivity along the ring be maintained if multiple group-heads leave simultaneously? In this paper, we have presented a simple solution for this. We propose that each group-head P_r and its secondary one, $p^r{}_1$ store the tuple, $[P_{r-1}, p^{r-1}{}_1, P_{r+1}, p^{r+1}{}_1]$. The following example explains the idea.

Let P_{i-1}, P_i, and P_{i+1} be the group-heads of three consecutive groups on the ring. We also call them as primary group-heads. The resource types corresponding to the group-heads are R_{i-1}, R_i, and R_{i+1} respectively. Let in P_i, the secondary group-head be $p^i{}_1$; similarly, the respective secondary group-heads in P_{i-1} and P_{i+1} are $p^{i-1}{}_1$ and $p^{i+1}{}_1$.

Therefore, both P_{i-1}, $p^{i-1}{}_1$ have the tuple $[P_{i-2}, p^{i-2}{}_1, P_i, p^i{}_1]$; similarly, both P_i and $p^i{}_1$ have the tuple $[P_{i-1}, p^{i-1}{}_1, P_{i+1}, p^{i+1}{}_1]$; and both P_{i+1} and $p^{i+1}{}_1$ have the tuple $[P_i, p^i{}_1, P_{i+2}, p^{i+2}{}_1]$.

Now, let us consider the worst-case scenario of all three primary group-heads, i.e. P_{i-1}, P_i, and P_{i+1} leaving at the same time. We observe that in G_i the new primary group-head $p^i{}_1$ has the IP addresses of the new primary group-heads $p^{i-1}{}_1$ and $p^{i+1}{}_1$ of the groups G_{i-1} and G_{i+1}. Therefore, group G_i remains connected to its neighboring groups G_{i-1} and G_{i+1}. Also, in group G_{i-1}, its new primary group-head $p^{i-1}{}_1$ uses the IP address of the group-head P_{i-2} to communicate with this group along the ring network; if P_{i-2} leaves, new primary group-head $p^{i-1}{}_1$ can communicate with this group along the ring network via the IP address of $p^{i-2}{}_1$. Similarly, we observe that group G_{i+1} can communicate with its neighboring groups as well. Observe that to enhance the degree of fault-tolerance, tuple-size can be increased to include more members of a group. The above discussion leads to the following observation.

Observation 3. Transit ring network remains connected even if consecutive primary group-heads leave the system.

VII. CONCLUSION

We have extended our earlier work to incorporate the general idea that a peer can possess multiple resource types. It enhances the scope of application of the architecture. The time complexity of the proposed inter-group data lookup algorithm is bounded by $(2 + r/2)$, where r is the number of distinct resource types and $r \ll n$, where n is the total number of peers in the system. For intra-group data look up, it needs two hops only [14]. Handling churn is remarkably efficient because of some noteworthy features of the architecture, viz., diameter of every group is one and location of a peer with new resource type is always between the existing last and the first peers on the ring unlike in any DHT-based architecture. We have also shown that keeping track of few IP addresses by the group-heads and the corresponding secondary group heads helps in efficient ring maintenance. This work is a part of an ongoing research project with the goal of designing P2P federation

consisting of small P2P systems so that bandwidth cannot be an issue.

REFERENCES

[1] P. Ganesan, Q.Sun, and H. Garcia-Molina, "Yappers: A peer-to-peer lookup service over arbitrary topology," Proc. IEEE Infocom, pp. 1250-1260, 2003, San Francisco, USA, March 30 - April 1 2003.

[2] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker, "Making gnutella-like p2p systems scalable," Proc. ACM SIGCOMM, Karlsruhe, Germany, August 25-29, 2003.

[3] 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, vol. 22, no. 1, pp. 41-53, Jan. 2004.

[4] A. Rowstron and P. Druschel, "Pastry: scalable, distributed object location and routing for large scale peer-to-peer systems," Proc. IFIP/ACM Intl. Conf. Distributed Systems Platforms (Middleware), pp. 329-350, 2001.

[5] 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, vol. 11, No. 1, pp. 17-32, Feb. 2003.

[6] M. Yang and Y. Yang, "An efficient hybrid peer-to-peer system for distributed data sharing," IEEE Trans. Computers, vol. 59, no. 9, pp. 1158-1171, Sep. 2010.

[7] M. Xu, S. Zhou, and J. Guan, "A new and effective hierarchical overlay structure for peer-to-peer networks," Computer Communications, vol. 34, pp. 862-874, 2011.

[8] 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.

[9] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content-addressable network," Proc. SIGCOMM Annual Conference on Data Communication, Aug. 2001.

[10] B.T. Loo, R. Huebsch, I. Stoica, and J.M. Hellerstein, "The case for a hybrid p2p search infrastructure," Proc.Workshop Peer-to-Peer Sysyems (IPTPS'04), pp. 141-150, Feb. 2004.

[11] Z. Peng, Z. Duan, J.Jun Qi, Y. Cao, and E. Lv, "HP2P: a hybrid hierarchical p2p network," Proc. Intl. Conf. Digital Society, 2007.

[12] K. Shuang, P Zhang, and S. Su, "Comb: a resilient and efficient two-hop lookup service for distributed communication system," Security and Communication Networks, vol. 8(10), pp. 1890-1903, 2015.

[13] Bidyut Gupta, Shahram Rahimi, Ziping Liu, and Sindoora Koneru, "Design of structured peer-to-peer networks using linear diophantine equation," Proc. CAINE, pp. 147-151, New Orleans, Oct., 2014.

[14] N. Rahimi, K. Sinha, B. Gupta, and S. Rahimi, "LDEPTH: A low diameter hierarchical p2p network architecture," Proc. 2016 IEEE Int. Conf. on Industrial Informatics (INDIN 2016), Poitiers, France, July, 2016.

[15] R. Zhang and Y.C. Hu, "Assisted peer–to-peer search with partial indexing," IEEE Trans. Parallel and Distributed Systems, vol. 18(8), pp. 1146-1158, 2007.

[16] E. Cohen, A. Fiat, H. Kaplan, "Associative search in peer-to-peer networks: harnessing latent semantics," vol. 2, pp. 1261-1271, 2003.

[17] Andrea Passarella, "A survey on content-centric technologies for the current internet: cdn and p2p Solutions," Computer Communications, vol. 35, pp. 1-32, 2012.