Secured Communication in Generalized Non DHT-based Pyramid Tree P2P Architecture

Nick Rahimi*
University of Southern Mississippi, Hattiesburg, MS

Indranil Roy†, Ziping Liu†
South East 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. When first reported, peers in each cluster in the architecture used to possess instances of a particular resource type only. It was definitely a very hard restriction practically. Recently, to overcome this restriction a generalized form of the architecture has been reported in which any peer in any cluster can have more than one resource type. This generalized architecture is a little bit more complex than the original one and yet efficiency of each data look-up protocol in it remains the same as in the simpler initial version of the architecture. In this paper, as a continuation of our work in this direction, we have considered security in communication in the generalized architecture. To achieve it, mainly public key-based approach has been used for the different look-up protocols (except for multicasting) because the required number of public-private key pairs is small. However, for multicasting among the cluster-heads, we have followed a hybrid approach, because number of symmetric keys required is just one independent of the size of the multicast group and only the public-private key pair of the root cluster-head is needed.

Keywords: Structured P2P network, residue class, interest-based, non-DHT, secured communication.

1 Introduction

Recent trend in designing structured P2P architectures is the use of distributed hash tables (DHTs) [19, 21, 29]. Such overlay architectures can offer efficient, flexible, and robust service [4, 19, 21, 29, 31]. 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 [13, 17, 28, 32]; 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, 8, 18, 22, 25]. It offers advantages of DHT-based systems, while it attempts to reduce the complexity involved in churn handling. Authors in [22] 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' [7, 20]. It is an interest-based peer-to-peer system [1, 5, 19-12, 18, 22, 25, 27, 30] with peers of common interest clustered together. Its main focus is to improve the efficiency of data lookup protocols in that a query for an instance of a particular resource type is always directed to the cluster of peers which possess different instances of this resource type. So, success or failure to get an answer for the query involves a search in that cluster only instead of searching the whole overlay network as in the case of unstructured networks. However, that a peer can have only one resource type is a hard restriction practically. To overcome this problem, recently, a generalized form of the architecture [9] has been reported in which any peer in any cluster can possess more than one resource type.

1.1 Our Contribution

In the present work, as a continuation of our research in Pyramid tree p2p network area, we have considered security in communication in the generalized architecture and for this we have preferred public key-based cryptographic approach to redesign/edit our already reported intra- and inter-cluster data

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

look-up protocols, and the broadcast protocol without compromising with the efficiencies of the earlier reported protocols. However, a combination of symmetric and public key-based approach has been considered in designing a very efficient multicast protocol.

The organization of the paper is as follows. In Section 2, we talk briefly about some related preliminaries. For a better understanding of the architecture, refer to our recent publications in this direction. In Section 3, the newly designed secured intra- and inter-cluster data look-up protocols have been presented. In Section 4 we have presented the secured broadcast protocol and Section 5 contains the multicast protocol. Section 6 draws the conclusion.

2 Preliminaries

In this section, we present some relevant findings from our recent works on the Pyramid tree based P2P architecture [15, 22-24] 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 $S = \{C_i\}$, $0 \le i \le n-1$, 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.*

The overlay network considered is a 2-layer non DHT based architecture [22]. At layer-1, there exists a tree like structure, known as a 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.

2.1 Characteristics of Pyramid Tree

The following overlay architecture has been proposed in [15, 22-24].

- 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 [7].
- 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 $[r] = \{a: a \text{ is an integer}, a \equiv r \pmod n\}$.

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, each integer representing a residue class is the logical (overlay) address of the cluster-head of a cluster. For example, logical address of the first cluster-head is 0, for the second one it is 1, and so on. We use the integers 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 only the positive integer values are used for addressing. It becomes clear that mathematically any class consists of an infinite number of integers; it means that we do not put any limit on the size of a cluster. In general, number of peers can be too large compared to the number of distinct resource types.

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.*

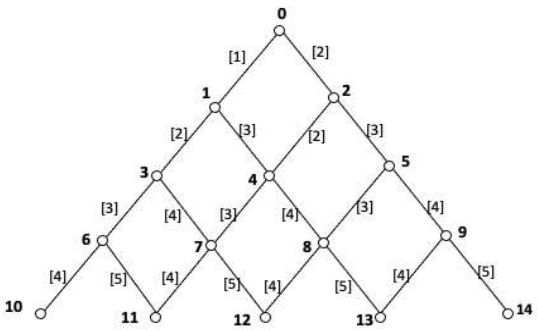

Figure 1: A complete pyramid tree with root 0

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.

3 Public Key and Symmetric Key-Based Secured Communication

Before we present the secured protocols the following information from the generalized architecture needs to be recalled. In the generalized architecture each cluster-head maintains a table of information *(TOI)* consisting of tuples corresponding to different cluster-heads. For example, the tuple for cluster-head C_i^h appears as < Res. Code i, IP (C_i^h) >. If a peer p in cluster C_i possesses another distinct resource type, say j, it will appear as a cluster-head as C_j^h in the *TOI*. Same is true if an existing cluster-head C_i^h possesses another distinct resource type, say k, it will appear as a different cluster-head Ck^h However, both Ck^h and Ci^h will have the same IP address but with different overlay addresses k and i respectively. We shall use *TOI* in some of our presented protocols here. Besides, we shall use the broadcast protocol designed for the generalized architecture in our protocols wherever needed. This protocol is known as generalized-broadcast-incomplete protocol. 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.

We have considered public key-based approach for most of the presented protocols in this work. However, symmetric key-based approach can also be used. As an example, we have shown how symmetric key-based approach can be used for intra-cluster data look-up. In addition, we have used a combination of both public and symmetric keys in designing the proposed secured multicast protocol.

In the rest of this section, we shall use the following notations. Public key and private key of the root cluster-head C_0^h are denoted as PU_0 and PR_0 respectively. For any other cluster-head C_i^h, these are PU_i and PR_i respectively. Request for an instance of a resource with code *i* is denoted as *Req-i*; and of course, the IP packet containing *Req-i* will have the requesting peer's identity, that is, its IP address. The corresponding response to *Req-i* is denoted as *Res-i*.

The process of encrypting a message M with a ***key*** is denoted as *E (**key**, M)* and decrypting as *D (**Key**, M)*. We have mentioned earlier that resource code is the same as the cluster-head's logical address; for example, if the logical address of a cluster-head C_i^h is i, resource code of the resource owned by the cluster-head as well as by peers in the cluster C_i is also i.

3.1 Public Key Distribution Among Cluster-Heads [16]

We start with a simple method for cluster-heads to know the public keys of all other cluster-heads. Assume that so far there are k number of clusters present in the network, with logical addresses of the existing cluster-heads being 0 to (k-1). That is, the largest resource code currently present in *TOI* is (k-1). In this context note that in the generalized P2P network, it is possible that $IP(C_i^h) = IP(C_j^h)$, i.e., the same peer represents two cluster-heads of two different clusters C_i and C_j containing respectively peers with resource type i and resource type j. Therefore, this peer will have two different logical addresses and two pairs of public key-private key; and the peer will use both the public keys and the private keys of the cluster-heads whenever needed. Same is true for any number of cluster-heads that the peer may represent. When a peer p with instance(s) of some resource type wants to join, at first cluster-head C_0^h checks its *TOI* if the resource type of the joining peer is already present in it. Now one of the following two different situations may arise.

Situation 1: Resource type of peer p does not exist in *TOI*

1. *New peer p contacts C_0^h for joining the network.*
2. *Cluster-head C_0^h assigns the joining peer with the next largest available number for the code; so, the code for p becomes k because TOI contains resource codes from 0 up to (k-1) before peer p joins.*
3. *C_0^h makes entry in the TOI for the new resource code k (which is now the logical address of the newly joining peer p) and the IP address of peer p, because now peer p becomes the cluster-head C_k^h.*
4. *C_0^h and C_k^h exchange their public keys, namely PU_0 and PU_k. Their respective private keys are PR_0 and PR_k. Cluster-head C_0^h updates a list L by including PU_k in it. List L now contains (k+1) different public keys*

corresponding to (k +1) cluster-heads. / List L initially contains only PU_0 of C_0^h / The same peer representing multiple cluster-heads with identical IP addresses but with different resource types, will have unique public key-private key pairs for each such cluster-head

5. *C_0^h performs E (PR_0, L) and executes the generalized-broadcast-incomplete protocol so that each node (cluster-head) on the tree receives a copy of the encrypted list L; each receiving node performs D (PU_0, E (PR_0, L)) to get the recent copy of the list L. / Now each node has the knowledge of the public keys of all existing nodes.*
6. *The above five steps are executed each time a peer with a new resource type contacts C_0^h to join the network.*

Situation 2: Resource type of peer p exists in *TOI*

When a new peer p with some instance(s) of an existing resource type with code, say, *m,* wants to join the system, the following steps are executed to include the peer in the network.

1. *New peer p contacts C_0^h for joining the network*
2. *C_0^h checks with the $(m+1)^{th}$ entry in TOI to get the IP address of C_m^h*
3. *C_0^h sends the IP address of C_m^h to p*
4. *Peer p contacts C_m^h*
5. *C_m^h gives its public key PU_m to peer p / Peer p will use this public key for secured communication inside cluster C_m as well as for inter-cluster data-look-up*

3.2 Symmetric Key Distribution Among Peers in a Cluster

When a new peer with some instance(s) of an existing resource type *m* wants to join the system, the following steps are executed to include the peer in the network.

1. *New peer p contacts C_0^h for joining the network*
2. *C_0^h checks with the $(m+1)^{th}$ entry in TOI to get the IP address of C_m^h*
3. *C_0^h sends the IP address of C_m^h to p*
4. *Peer p contacts C_m^h*
5. *C_m^h and peer p exchange their symmetric keys. / Peer p may use this symmetric key for secured intra-cluster communication in cluster C_m*

Note that if the number of peers in a cluster C_i is N, there will be N distinct symmetric keys, one for each peer; the list containing these corresponding N symmetric keys will be present with the cluster-head C_i^h.

Observation 1. Considering both Situations 1 and 2, total number of required Public-Private key pairs (N') is the number of cluster-heads present in TOI. Some peer may appear as cluster-head multiple times in a generalized situation; thus, depending on the situation this peer may possess more than one public-private key pair.

Observation 2. Number of required symmetric keys (N") for secured communication inside a cluster is the number of peers present in the cluster not counting the cluster-head. This number N" is much larger than N' because number of peers in a cluster is supposed to be very large compared to the number of distinct resource types [6].

Observation 3. Public key-based approach is preferred to symmetric based approach because of much smaller number of distinct keys required in the former.

In all protocols stated in this section, it is assumed that each cluster head has a copy of the list *L* and *TOI.* Besides, it is assumed that peers in a cluster are trustworthy and they are not intruders. An intruder peer is an outsider and does not belong to the P2P network.

3.3 Secured Intra-cluster Data Look-up Protocol

Assume that a peer p' in cluster C_i issues a data look-up request *Req-i* in C_i. Either public key or symmetric key cryptography can be used for secured communication. In the presented protocols, we denote IP address of a peer X as IP(X). We first state secured intra-cluster protocol using public key based cryptographic approach (Figure 2), followed by symmetric key based approach (Figure 3). After that we shall consider secured communication with anonymity.

3.3.1 Use of Public Key Cryptography

1. *if $p' = C_i^h$*
   C_i^h broadcasts the request Req-i in C_i/One hop communication
   if a peer p in C_i has the requested response Res-i*
   peer p unicasts E (PU_i, Res-i) to C_i^h*
   C_i^h performs D (PR_i, E (PU_i, Res-i)) to receive the response Res-i
   else
   Search for Req-i fails
2. *else*
   p' unicasts Req-i to C_i^h
   if C_i^h has the requested response Res-i
   C_i^h unicasts E (PR_i, Res-i) to peer p'
   peer p' performs D (PU_i, E (PR_i, Res-i)) to receive the response Res-i
   else
   C_i^h broadcasts the request Req-i in C_i
3. *if a peer p* in C_i has the requested response Res-i*
   peer p unicasts E (PU_i, Res-i) to cluster-head C_i^h*
   C_i^h performs D (PR_i, E (PU_i, Res-i)) to get Res-i
   C_i^h unicasts E (PR_i, Res-i) to the peer p'
   peer p' performs D (PU_i, E (PR_i, Res-i)) to receive the response Res-i
4. *else*
   Search for Req-i fails

Figure 2: Public key based secured intra-cluster data lookup

Remark 1. Only the public-private key pair of the cluster-head is needed to ensure security.

3.3.2 Use of Symmetric Key Cryptography

1. if $p' = C_i^h$
   C_i^h broadcasts the request Req-i in C_i / One hop communication
   . if a peer p* in C_i has the requested response Res-i
   P* unicasts E (SyKey (p*, C_i^h), Req-i) to cluster-head C_i^h
   / SyKey (p*, C_i^h) is the common symmetric key of p* and C_i^h
   C_i^h performs D (SyKey (p*, C_i^h), E (SyKey (p*, C_i^h), Res-i)) to get Req-i
   else
   Search for Req-i fails
2. else
   p' unicasts its Req-i to cluster-head C_i^h
   if C_i^h has the requested response Res-i
   C_i^h unicasts E (SyKey (p', C_i^h), Res-i) to peer p'
   peer p' performs D (SyKey (p', C_i^h), E (SyKey (p', C_i^h), Res-i)) to get Req-i
3. else
   C_i^h broadcasts the request Req-i in C_i
   if a peer p* in C_i has the requested answer
   peer p* unicasts E (SyKey (p*, C_i^h), Res-i) to cluster-head C_i^h;
   C_i^h performs D (SyKey (p*, C_i^h), E (SyKey (p*, C_i^h), Res-i)) to check the response Res-i
   C_i^h performs E (SyKey (p',C_i^h), Res-i)
   C_i^h unicasts the encrypted response to peer p'
   peer p' performs D (SyKey (p',C_i^h), E (SyKey (p',C_i^h), Res-i)) to receive the response Res-i
   else
4. search for Res-i fails

Figure 3: Symmetric key based secured intra-cluster data lookup

We mentioned earlier that number of required symmetric keys for secured communication inside a cluster is the number of peers present in the cluster not counting the cluster-head. This number may be very large. However, for public key-based approach it is only the public key and the private key of a cluster-head C_i^h that are required for secured intra-cluster communication in that cluster C_i. Because of this very small number of keys required, we have considered public key-based approach for designing secured communication protocols with anonymity.

3.4 Secured Intra-Cluster Communication with Anonymity

Let us first state a general idea to achieve anonymity irrespective of if the protocol used is an intra-cluster or an inter-cluster one. Achieving anonymity in the architecture is a bit tricky because diameter of each cluster is just one. It means that for intra-cluster communication, a requester and a replier are always one hop away from each other. So, logically there is no other peer present between them; this means they cannot hide their respective identities from each other while communicating with each other through request and response. It has led us to present the following simple yet effective solution to achieve anonymity between a requester and a responder inside a cluster. It can be applied to inter-cluster communication as well. The idea is somewhat similar to the idea used in *Mixes*. In general, the idea works as follows.

Let the requesting peer be p' and also let p" be a randomly chosen peer by p' toward the destination. Peer p' sends a request packet directly to peer p". Thus, p" acts as an intermediate forwarding peer of the request packet issued by peer p'. Before unicasting further, peer p" replaces the IP address in the source field in the received packet by its own identity. Thus, p" appears now as if it is the source of the requester to the next peer along a randomly chosen path to the destination.

In the present work, the path of query propagation is termed here as *query-path*. Furthermore, we assume that these peers will remember, wherever applicable, their respective immediate senders' identities and immediate followers' identities along the query propagation path so that a reply can follow *the reverse-query-path* all the way to the requesting peer. Similar approach of replacing addresses is followed along the reverse path while forwarding the corresponding response toward the requesting peer. Therefore, the replier will have no clue about who the original requesting peer is. Similarly, the requester will not know the true identity of the replier.

It may appear that too many address replacements may be needed, but we observe that it is not the case in both intra- and inter-cluster lookup protocols; it is at most two. Recall that diameter of a cluster is just one hop. Therefore, in the present work to achieve anonymity, a request is sent by a requesting peer p_i to its cluster-head C_i^h always via a randomly chosen peer (i.e., using only two hops); hence, source identity is replaced only once in a request packet while it travels to the cluster-head except in the case when the cluster-head itself issues the request; in this later case, only the response packet may go through the replacement process once if at all needed. Similarly, a response packet is always forwarded by the cluster-head C_i^h to the requesting peer p_i using two hops only; thereby replacement occurs only once in the response (reply) packet, irrespective of if the proposed protocol is intra- or inter-cluster protocol. Thus, we observe that only two replacements are sufficient to hide the identities of the requesting peer and the responder from each other. In other words, replacements occur only in one segment (from requesting peer to its cluster-head) of the query-path. Similarly, it occurs only in one segment (from the cluster-head to the requesting peer) of the reverse-query-path. Therefore, the look up latency increases only by two hops.

The public key-based secured intra-cluster data lookup protocol with anonymity appears in Figure 4 below.

1. if $p' = C_i^h$
C_i^h broadcasts the request Req-i in C_i / One hop communication
. / C_i^h always broadcasts a request irrespective of if it is the actual requester or not; so, no receiving peer has any clue about the true requester
if a peer p in C_i has the requested response*
peer unicasts E (PU_i, Res-i) to a randomly chosen peer p^r in C_i*
peer p^r unicasts Req-i to cluster-head C_i^h /Intruder cannot get Res-i as it does not know PR_i

C_i^h performs D (PR_i, E (PU_i, Res-i)) to get the response Res-i / C_i^h has no clue about the identity of the true responder
else
Search for Req-i fails
2. else
p' unicasts Req-i to a randomly chosen peer p^r in C_i
peer p^r unicasts Req-i to cluster-head C_i^h /Req-i is sent along the query-path to C_i^h

/ C_i^h does not know the identity of the actual requester
3. if C_i^h has the requested response Res-i
C_i^h unicasts E (PR_i, Res-i) to the random peer p^r along the reverse-query-path
peer p^r unicasts E (PR_i, Res-i) to peer p' / reverse-query-path is followed to reach the true requester
peer p' performs D (PU_i, E (PR_i, Res-i)) to receive the response Res-i

/Anonymity of the responder is preserved
else
C_i^h broadcasts the request Req-i in C_i
4. if a peer p in C_i has the requested response Res-i*
peer p unicasts E (PU_i, Res-i) to cluster-head C_i^h*

/Intruder cannot get Res-i as it does not know PR_i
C_i^h decrypts the information to get Res-i
C_i^h unicasts E (PR_i, Res-i) to the random peer p^r along the reverse-query-path
peer p^r unicasts E (PR_i, Res-i) to peer p' / reverse-query-path is followed to arrive at the true requester
peer p' performs D (PU_i, E (PR_i, Res-i)) to receive the response Res-i

/ p' does not know the identity of the true responder
else
Search for Req-i fails

Figure 4: Public key based secured intra-cluster data lookup with anonymity

Theorem 1. Anonymity between a requesting peer and a responding peer is preserved.

Proof: In the protocol, if cluster-head is the requester, any response arrives at the cluster-head via a randomly chosen peer in the cluster. Hence, cluster-head does not know the identity of the actual responder. On the other hand, if cluster-head is not the requester, any response is always unicasted finally by the cluster-head to the requesting peer via a randomly chosen peer in the cluster. So, such a requesting peer will not have any clue about the true responder. Similarly, any request from a peer other than the cluster-head always arrives at the cluster-head via a randomly chosen peer. So, the cluster-head has no clue about the true requester. Besides, whenever needed whether the requesting peer itself is the cluster-head or not, the Req-i is always broadcasted in the cluster by the cluster-head. So, no responder will have any clue about the true requester. Hence anonymity between a requesting peer and a responding peer is always preserved. □

The protocol offers secured communication because of two reasons: first, any intruder cannot know the private key of any cluster-head; therefore, when a responding peer unicasts to the corresponding cluster-head its response encrypted with the public key of the cluster-head, no intruder can know the response. Second, even if the intruder comes to learn about the public key of the cluster-head, it will be almost impossible to guess the random peer on the reverse-query-path along which the cluster-head unicasts any response encrypted with its private key: meaning that an intruder cannot identify the path of the reply to the requester making it impossible to look for the reply.

Observation 4. Security with anonymity with symmetric key based approach in intra-cluster data look-up can be achieved using query-path and reverse-query-path in a similar way as in the public key-based approach.

3.5 Secured Inter-Cluster Data Look-up Protocol with Anonymity

We shall illustrate the steps of the protocol using the following inter-cluster communication scenario. Let a peer p* in cluster C_i get an instance of a resource with code m. So, the request *Req-m* should be sent to cluster C_m to get any response related to the query. Note that the cluster-head C_i^h itself may also be the requesting node. Besides, the generalized architecture, the same peer may appear as multiple cluster-heads [26]; in such a case $IP(C_i^h)$ and $IP(C_m^h)$ will be identical even though the two cluster-heads will have different logical addresses in the *TOI*. Therefore, we require to consider all four possible cases involving the two cluster-heads and the requesting peer. The protocol is stated below in Figure 5.

if Peer p ≠ C_i^h and $IP(C_i^h) = IP(C_m^h)$ / Case 1*
Peer p unicasts Req-m to cluster-head C_i^h via a query-path in C_i / path is chosen as in intra-cluster protocol*
if C_m^h has the answer to Req-m

C_i^h unicasts E (PR_i, Res-m) to the requesting peer p* via the reverse-query-path

/ C_m^h and C_i^h are the same peer with different private keys

and different logical addresses
peer p* decrypts with PU_i to get the Res-m / anonymity of requester and responder is preserved
else
C_m^h broadcasts the request in C_m
if ∃ p' with the answer Res-m
p' unicasts E (PU_m, Res-m) to its cluster-head C_m^h
C_m^h performs D (PR_m, E (PU_m, Res-m) to get Res-m
C_i^h unicasts E (PR_i, Res-m) to the requesting peer p* via the reverse-query-path in C_i

/ C_m^h and C_i^h are the same peer with different private keys

different logical addresses
peer p* decrypts with PU_i to get the Res-m / Anonymity of requester and responder is preserved

else
C_i^h unicasts 'fail' information to peer p* via the reverse-query-path in C_i / Search for Req-m fails
else
if Peer p* ≠ C_i^h and IP(C_i^h) ≠ IP(C_m^h) / Case 2
C_i^h finds from list L the public key PU_m of cluster-head C_m^h / Identifies the cluster with resource type m
C_i^h unicasts E (PU_m, Req-m) to cluster-head C_m^h / Inter-cluster communication
C_m^h performs D (PR_m, E (PU_m, Req-m)) to get Req-m
if C_m^h has the answer to Req-m
It unicasts E (PU_i, Res-m) to cluster-head C_i^h / Inter-cluster communication
C_i^h performs D (PR_i, E (PU_i, Res-m)) to get Res-m / C_i^h receives the response
C_i^h unicasts E (PR_i, Res-m) to the requesting peer p* via the reverse-query-path in C_i
peer p* decrypts with PU_i to get the Res-m / Anonymity of requester and responder is preserved
else
C_m^h broadcasts the request in C_m
if ∃ p'' with the answer Res-m
p'' unicasts E (PU_m, Res-m) to its cluster-head C_m^h
C_m^h performs D (PR_m, E (PU_m, Res-m) to get Res-m
C_m^h unicasts E (PU_i, Res-m) to cluster-head C_i^h / Inter-cluster communication
C_i^h performs D (PR_i, E (PU_i, Res-m)) to get Res-m / C_i^h receives the response
C_i^h unicasts E (PR_i, Res-m) to the requesting peer p* via the reverse-query-path in C_i
peer p* decrypts with PU_i to get the Res-m / Anonymity of requester and responder is preserved
else
C_m^h unicasts 'fail' information to C_i^h / Search for Req-m fails
C_i^h unicasts it to peer p* via the reverse-query-path in C_i
else
if Peer p* = C_i^h and IP(C_i^h) ≠ IP(C_m^h) / Case 3
C_i^h finds from list L the public key PU_m of cluster-head C_m^h / Identifies the cluster with resource type m
C_i^h unicasts E (PU_m, Req-m) to cluster-head C_m^h / Inter-cluster communication
C_m^h performs D (PR_m, E (PU_m, Req-m)) to get Req-m
if C_m^h has the answer to Req-m
It unicasts E (PU_i, Res-m) to cluster-head C_i^h / Inter-cluster communication
C_i^h performs D (PR_i, E (PU_i, Res-m)) to get Res-m / C_i^h receives the answer to its query

/ C_i^h has no clue about the identity of the original responder;

that is, C_i^h is not sure if a peer in C_m or C_m^h itself has the

response
else
C_m^h broadcasts the request in C_m
if ∃ p'' with the answer Res-m
p'' unicasts E (PU_m, Res-m) to its cluster-head C_m^h
C_m^h performs D (PR_m, E (PU_m, Res-m) to get Res-m
C_m^h unicasts E (PU_i, Res-m) to cluster-head C_i^h / Inter-cluster communication
C_i^h performs D (PR_i, E (PU_i, Res-m)) to get Res-m / C_i^h receives the response

/ C_i^h has no clue about the identity of the original responder

/ C_i^h is not sure if a peer in C_m or C_m^h itself has the response

else
C_m^h unicasts 'fail' information to C_i^h
/ Search for Req-m fails

else
if Peer p = C_i^h and IP(C_i^h) = IP(C_m^h) / Case 4*

if C_m^h has the answer to Req-m
C_i^h receives the answer Res-m to its query
/ C_m^h and C_i^h are the same peer with different private keys

and different logical addresses

/ C_i^h is not sure if a peer in C_m or C_m^h itself has the response
else
C_m^h broadcasts the request in C_m
if ∃ p' with the answer Res-m
p' unicasts E (PU_m, Res-m) to its cluster-head C_m^h
C_m^h performs D (PR_m, E (PU_m, Res-m) to get Res-m
C_i^h receives the answer Res-m to its query / C_i^h has no clue about the identity of the original responder;

/ C_i^h is not sure if a peer in C_m or C_m^h itself has the response
Else
Query fails

Figure 5: Public key based secured inter-cluster data lookup with anonymity

Theorem 2. In inter-cluster communication, anonymity between a requesting peer and a responding peer is preserved.

Justification similar to the one used in Theorem 1 can be used here to prove that anonymity of both requesting and responding peers is preserved.

Observation 5. Only the public and private keys of two cluster-heads are necessary for secured inter-cluster data look-up with anonymity.

Remark 2. For both intra- and inter-cluster data look-ups with anonymity, the source address in a packet needs to be replaced at most twice considering both propagation of a request and propagation of a response.

Remark 3. Data look up latency increases at most by two hops only.

It may be noted that in none of the above two protocols there is a need for any cluster-head to replace the source address in a received packet by its own identity.

4 Secured Broadcast in Pyramid Tree

Let node X be the broadcast source for a message M. The source may be a peer p in some cluster C_i or itself is the cluster-head C_i^h. In the former case, X will first unicast its broadcast message M to its cluster-head C_i^h. which will then unicast M to the root cluster-head C_0^h. This root will actually execute broadcasting in the tree.

In the latter case, C_i^h will unicast its broadcast message to the root which will then broadcast in the tree. Existence of virtual neighborhood [24] is the reason behind the root broadcasting instead of C_i^h. The *Generalized-Broadcast-Incomplete protocol* has appeared in [IJCA June-23]. In this work, we shall incorporate security in this protocol. The following three possibilities need be considered in designing the secured protocol: (1) X is in C_i, but $X \neq C_i^h$, (2) $X = C_i^h$, and (3) $X = C_0^h$

4.1 Protocol Secured Generalized-Broadcast

1. if X is in C_i, but $X \neq C_i^h$
X unicasts the message encrypted with public key PU_i of C_i^h to cluster-head C_i^h
C_i^h decrypts the message M with its private key PR_i
C_i^h encrypts the message M with its private key PR_i
C_i^h unicasts the encrypted message to C_0^h

C_0^h decrypts the encrypted message with C_i^h's public key PU_i
/ authenticates the broadcast source
C_0^h encrypts the message M with its private key PR_0
else
2. if $X = C_i^h$
C_i^h encrypts the message M with its private key PR_i
C_i^h unicasts the encrypted message to C_0^h
C_0^h decrypts the encrypted message with C_i^h's public key PU_i
/ authenticates the broadcast source
C_0^h encrypts the message M with its private key PR_0
else
3. if $X = C_0^h$
C_0^h encrypts the message M with its private key PR_0 / $X = C_0^h$
4. C_0^h executes Generalized- Broadcast-Incomplete protocol
/ A receiving cluster-head C_m^h decrypts the received encrypted message with the public key PU_0 of the root to get the message M

5. if broadcasting is needed in any cluster C_j
C_j^h encrypts the message M with its private key PR_j
C_j^h broadcasts in cluster C_j / one hop communication
each receiving peer in C_j decrypts the encrypted message with PU_j to get M

Figure 6: Secured generalized-broadcast protocol

In this protocol step 5 will not be executed if broadcasting involves only the cluster-heads, for example when the root cluster-head broadcasts a copy of the updated *TOI* to all other cluster-heads. In this case, in the protocol, only the public-private key pair of the source cluster-head (if $X \neq C_0^h$) and that of the root are needed. If broadcasting is done in the clusters as well as in step 3 above, the total number of public-private key pairs is the number of the cluster-heads present in the tree.

5 Secured Multicast to a Group of Cluster-Heads

We will use some idea from Protocol Independent Multicasting – Sparse Mode (PIM-SM) for multicasting [2] in WANs. The reason for this is that in a Pyramid tree P2P network, number of cluster-heads is very small compared to the total number of peers in the network because the total number of distinct resources is very limited [6]. Therefore, any multicast group of cluster-heads will definitely have a smaller size. According to PIM-SM a core is needed and obviously the root cluster-head C_0^h becomes the logical choice as the core irrespective of its membership for a given multicast group G.

We shall use a hybrid approach based on symmetric and public keys as stated below. Multicasting has two phases: in the first phase a group symmetric key SyKey(G) is selected by the core and the core distributes the key to all member cluster-heads of the group using core's private key; in the second phase, multicast takes place using the symmetric key. We denote a cluster-head C_i^h wishing to join a group G, as C_i^h *(G),* also called a cluster member, a multicast source (also a cluster-head) as X, and X may or may not belong to the group; we also denote a multicast message as M.

Phase 1: Distribution of group-symmetric key

1. *Each cluster member C_i^h (G) unicasts a join request Req to the core C_0^h*
/ a branch from C_i^h (G) to core (root) is built as in PIM/SM, thereby forming a multicast tree with core as its root
2. *Core C_0^h selects the cluster-symmetric key SyKey(G) and encrypts it with its private key PR_0*
3. *Core C_0^h unicasts the encrypted key SyKey(G) to each C_i^h (G),*
4. *Each C_i^h (G) decrypts with core's public key PU_0 to get the key SyKey(G)*
/ Number of symmetric keys required is only one
/ Only the public-private key pair of the root cluster-head is needed
/ Encryption of SyKey(G) only happens once

Phase 2: Multicast session

if X is a group member and not the core
it unicasts M encrypted with the key SyKey(G) to the core
Core multicasts the encrypted message to all cluster members
Each receiving cluster member decrypts with the key SyKey(G) to get M
else
if X is a group member and is the core
Core multicasts M encrypted with the key SyKey(G) to all cluster members
Each receiving cluster member decrypts with the key SyKey(G) to get M
else
if X is not a group member
X unicasts to the core the message M encrypted with core's public key PU_0
/ X does not have the group key
Core decrypts it with its private key PR_0 to get the message M
Core multicasts M encrypted with the key SyKey(G) to all cluster members
Each receiving cluster member decrypts with the key SyKey(G) to get M

Figure 7: Secured multicasting

The above multicast scheme uses only one group symmetric key (independent of the size of the group) and the public-private key pair of the root. This is an advantage, no doubt. However, problem like *man-in-the -middle-attack* may result if an intruder gets hold of the public key PU_0 of the root. In that case, it can easily decrypt E [PR_0, SyKey(G)] using PU_0 to get the group symmetric key. However, if the following group key distribution method as stated below is used in Phase 1, this problem due to intrusion can be avoided.

Phase 1: Distribution of group-symmetric key

1. *Each cluster member C_i^h (G) unicasts a join request Req to the core C_0^h*
2. *Core C_0^h selects the cluster-symmetric key SyKey(G) and encrypts it with the public key PU_i of C_i^h*
/ each cluster-head knows the public keys of all other cluster-heads from TOI
3. *Core C_0^h unicasts the encrypted key SyKey(G) to each C_i^h (G),*
4. *Each C_i^h (G) decrypts with its private key PR_i to get the key SyKey(G)*
/ Number of symmetric keys required is only one
/ Number of public-private key pairs is the number of group members
/ Number of encryptions of SyKey(G) is the number of group members

In the above method of distribution, no intruder will have the knowledge of the private key of any joining node. So, it cannot get the group symmetric key anyway. However, this advantage comes at a cost; viz. now the required number of encryptions of SyKey(G) by the core is the number of group members whereas it is just one in the previous method, meaning thereby that multicasting based on this distribution will take more time to finish. In this context, it may be noted that multicasting inside a cluster [20] can be made secure in a similar way as above.

A question may arise like why not only public key-based approach is used for multicasting. It can be; however, it will be unnecessarily time consuming and it will not be truly multicasting. For example, consider a group of only five members, say m_1 to m_5. A source member say, m_1 wants to multicast a message M to other group members m_2 to m_5. So, first m_1 has to encrypt its message M separately with the respective public keys of the other group members and then it separately unicasts the four differently encrypted messages to the core to be sent to the four receivers m_2 to m_5; and then only the core can perform multiple-unicast to the receivers. First of all, there are too many encryptions and unicasts on behalf of the source causing it to be time consuming; secondly, it is not actually the idea of core-based multicasting. Whereas with symmetric key, source m_1 can just encrypt the message M once irrespective of the size of the group and unicasts it to the core. Only symmetric key distribution needs to be considered before multicast starts. Hence, the use of only a public key-based approach cannot be considered.

6 Conclusion

In this paper, we have considered the generalized form of a recently reported 2-layer non-DHT-based structured P2P network. In the generalized architecture efficiency of each data look-up, protocol remains the same as in the simpler initial version of the architecture. This has prompted us to consider the security aspect of the different communication protocols in the generalized architecture. To achieve it, public key-based approach has been used for the different look-up protocols (except for multicasting) because the required number of public-private key pairs is small. However, for multicasting among the cluster-heads, we have presented a hybrid approach using both symmetric key and public-private key ideas. It is shown that the number of symmetric keys required is just one independent of the size of the multicast group and only the public-private key pair of the root cluster-head is needed. To enhance security further while multicasting, we have shown that required number of public-private key pairs is just the number of cluster-heads.

As a continuation of our research, we are now working on secured communication in a federation of P2P architectures that consist of multiple Pyramid tree networks.

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*, September 1997, 2201.

[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 Systems*, 18(9):1188-1201, Sept. 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 - (AINA'06), Vienna, 1:707-712, 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] Bidyut Gupta and Mohammad Mohsin, "Fault-Tolerance in Pyramid Tree Network Architecture," *J. Computer Systems Science and Engineering*, 10(3):164-172, July,1995.

[8] 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. 944-950, July 2017.

[9] B. Gupta, I. Roy, N. Rahimi, Z. Liu, and N. Debnath, "On Generalization of Non DHT-Based Pyramid Tree P2P Network Architecture," *IJCA,* 30(1):116-123, March 2023.

[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, pp. 48-63, 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.143-148, 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] Anjila Neupane, Reshmi Mitra, Indranil Roy, Bidyut Gupta, Narayan Debnath, "Efficient and Secured Data Lookup Protocol using Public-Key and Digital Signature Authentication in RC-Based Hierarchical Structured P2P Network," *IJCA,* 30(1):140-150, June 2023.

[17] Z. Peng, Z. Duan, J.Jun Qi, Y. Cao, and E. Lv, "HP2P: A Hybrid Hierarchical P2P Network," *Proc. Intl. Conf. Digital Society*, pp. 18-28, 2007.

[18] 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.

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

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

[21] 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.

[22] 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*, 63:121-129, 2019.

[23] 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):12-18, Sept. 2021.

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

[25] Indranil Roy, Koushik Maddali, Swathi Kaluvakuri, Banafsheh Rekabdar, Ziping Liu, Bidyut Gupta, 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.

[26] Indranil Roy, Nick Rahimi, Ziping Liu, Bidyut Gupta, and Narayan Debnath, "On Generalization of Residue Class Based Pyramid Tree P2P Network Architecture," *IJCA,* 30(1):54-65, March 2023 Secured.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

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

Nick Rahimi (photo not available) is the Director of Cyber Innovations 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 interests lie 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.

Indranil Roy (photo not available) 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 interests include the design of architecture and communication protocols for structured peer-to-peer overlay networks, security in overlay networks, and Blockchain.

Ziping Liu (photo not available) 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.