

# A Peer-to-Peer Network Based on Multi-Mesh Architecture

Sudheendra Murthy and Arunabha Sen  
{asen, sudhi}@asu.edu  
Department of Computer Science and Engineering  
Arizona State University  
Tempe, Arizona 85281

**Abstract**—This paper presents the design and evaluation of a highly scalable, decentralized and self-organizing peer-to-peer network architecture based on the multi-mesh topology. Our network automatically adapts to dynamic node arrivals, departures and failures. Each node maintains a fixed set of neighbor connections, regardless of the size of the network. This demonstrates the scalability of the network. Our network is close in spirit to the Content-Addressable Network. While the Content-Addressable Network uses torus as the underlying network topology, our network uses multi-mesh. Multi-mesh has some unique advantages over torus and this is reflected in the evaluations of our network against the Content-Addressable Network.

## I. INTRODUCTION

Peer-to-peer(P2P) Overlay Networks have gained a lot of popularity over the last few years for their applicability in highly scalable, decentralized distributed applications. Among the most recent P2P systems are Pastry [1], Tapestry [2], Chord [3] and Content-Addressable Network (CAN) [4]. These systems are based on the distributed resource discovery idea presented in [5] and fall into the category of distributed hash table systems. The issue of resource discovery in a P2P environment is to locate the distributed objects in an efficient way. Principal to the resource discovery scheme is a scalable distributed hash table, in which the objects are mapped on to different nodes in the system. The entire hash table itself is distributed among all the participants of the system.

In a P2P system, a lookup request for a particular object travels from node to node making application-level jumps and reaches the destination node containing the object. These application level hops are expensive and a P2P system should try to minimize the number of hops involved. Closely related is the issue of state space and routing efficiency tradeoff. State space is the space required to store the routing tables in each node to keep track of its neighbors. One of the main objectives in a P2P network is to achieve better routing efficiency, while maintaining a fixed state space. In this paper, we present a new P2P network that displays good routing efficiency, still maintaining a very small and fixed number of neighbors. Our network is based on the Multi-Mesh architecture [6]. A multi-mesh network consists of nodes with uniform degree four. The multi-mesh network is similar to a mesh and a torus network in terms of its simplicity of interconnections and routing. However, it possesses better topological properties compared

to both of them. For example, if  $N$  is the total number of nodes in the network, a two dimensional torus has a diameter of  $\Theta(N^{1/2})$ , whereas a multi-mesh has a diameter of  $\Theta(N^{1/4})$ . In both the graphs however, the degree of each node is four.

The rest of this paper is organized into three parts. In the first part, we introduce some necessary background and related work in P2P systems and multi-mesh network. In the second part, we present the design of our P2P system based on the multi-mesh architecture and the simulation results. The third part discusses some future extensions of the multi-mesh architecture to three dimensions.

## II. BACKGROUND AND RELATED WORK

Each node in a P2P network has a node identifier and stores all those objects whose key closely matches with the node identifier. This node is referred to as the *home node* for those objects. The most basic operation in a P2P system is `lookup(key)`, which locates the node that stores the object with the key. When some node in the network issues a `lookup` request for an object, the request is routed in the overlay network towards the home node of that object. The existing distributed hash table P2P systems namely, Pastry, Tapestry, Chord and CAN form a different overlay network structure and has a different routing algorithm.

In Pastry, the node identifiers are chosen from a 128-bit circular key space. The routing table has  $\log_b N$  rows ( $b$  is some integer), with each row having nodes whose identifiers match one prefix more than those of the previous row. Routing is done by matching the identifier in the local routing table for the longest shared prefix with the key. Pastry routes a message within  $O(\log_b N)$  hops.

Tapestry is a variant of the resource discovery idea presented in [5]. The modifications support dynamic node insertions and departures that were not supported earlier. As in Pastry, Tapestry nodes maintain links to a set of neighbors that share common prefixes with its identifier. Tapestry can route a message in  $O(\log_b N)$  hops.

In Chord, the home node of an object is the node whose identifier most closely succeeds the key of that object. The routing table in each node has a list of  $k$  nodes that immediately follow it in the key space. The graph formed by Chord can be visualized as a circle with a number of chords. Each node maintains  $O(\log_2 N)$  chords spaced exponentially around

the key space and these chords improve the efficiency of the routing. Chord can route a message within  $O(\log_2 N)$  hops.

CAN places nodes into a virtual, multi-dimensional key space. The entire key space is divided among all the existing live nodes. The borders of this virtual space are wrapped around to form a d-dimensional torus topology. Each node owns a region of this space called the zone and its neighbors are the nodes that own the contiguous zones. Routing queries are passed along the axes in this virtual space until they reach the destination. CAN nodes maintain  $O(d)$  neighbors and the path-lengths are  $O(dn^{1/d})$  hops.

Though Pastry, Tapestry and Chord have shorter path-lengths than CAN, it comes at the expense of having to maintain more state information in the routing tables. The routing state information in Pastry, Tapestry and Chord grows with the number of nodes in the network. This overhead involves not just the extra storage required to store the routing tables but also the number of updates that have to be done when nodes join or leave the network. We would like to achieve shorter path-lengths, while maintaining a fixed number of neighbors and this is achieved by our Multi-Mesh based P2P system.

These systems are often evaluated with respect to three parameters.

1) Overlay path-length - This is the diameter of the P2P overlay network measured in terms of number of hops.

2) Neighbor state - This is the size of the routing table in each participating node in the P2P overlay network.

3) Cost of node insertions - An insertion of a new node into the overlay network incurs some overhead in terms of the messages exchanged in order to affect the routing tables in the neighboring nodes.

Table I shows a comparison of Pastry, Tapestry, CAN, Chord with respect to these parameters.

TABLE I  
METRICS COMPARISON

Parameter	Pastry	Tapestry	Chord	CAN
Overlay Path-Length	$O(\log_b N)$	$O(\log_b N)$	$O(\log_2 N)$	$O(N^{1/d})$
Neighbor-State	$O(b \log_b N)$	$O(b \log_b N)$	$O(\log_2 N)$	$O(d)$
Messages to insert	$O(\log_b N)$	$O(\log_2^2 N)$	$O(\log_2^2 N)$	$O(dN^{1/d})$

CAN is based on torus network and the above measures of diameter and path-lengths are directly derived from the properties of torus. The multi-mesh network has the same fixed neighbor state as that of two-dimensional torus but has a better path-length than torus, which makes it an ideal candidate for adaptation to P2P networks.

### III. THE MULTI-MESH NETWORK

The Multi-Mesh Network [6] was originally intended for use in parallel processor environments. Since its inception, it has also been proposed as an architecture for use in Optical

Networks [7]. A multi-mesh network is a uniform graph, in which the degree of each node is four. A Complete Multi-Mesh (CMM) network consists of exactly  $n^4$  nodes, where  $n$  is some integer ( $n \geq 3$ ). The basic unit of a multi-mesh network is a block, which is a complete mesh consisting of  $n^2$  nodes arranged in  $n$  rows and  $n$  columns.  $n^2$  such blocks arranged in  $n$  rows and  $n$  columns make up a CMM. Each node in a multi-mesh is identified by a four tuple  $(\alpha, \beta, x, y)$ , where  $1 \leq \alpha, \beta, x, y \leq n$ . The first two literals  $(\alpha, \beta)$  identify the block and the last two  $(x, y)$  identify the node in that block.

The  $n^2$  nodes in a block are connected as a regular two-dimensional mesh. The boundary nodes in a two-dimensional mesh have degree less than four. In a multi-mesh, these nodes are connected to other blocks using the following rules.

Rule 1)  $\forall \beta, 1 \leq \beta \leq n$ , and node  $(\alpha, \beta, 1, y)$  is connected to the node  $(y, \beta, n, \alpha)$  where  $1 \leq y, \alpha \leq n$ ,

Rule 2)  $\forall \alpha, 1 \leq \alpha \leq n$ , node  $(\alpha, \beta, x, 1)$  is connected to the node  $(\alpha, x, \beta, n)$  where  $1 \leq y, \alpha \leq n$

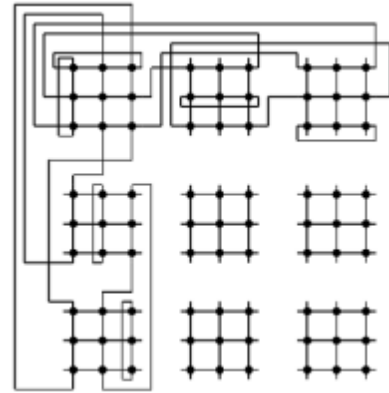


Fig. 1. A partially completed CMM with 81 nodes,  $n=3$  (Not all links are shown for clarity)

One limitation of a CMM is its requirement for exactly  $n^4$  nodes. This is a serious restriction on some networks that have less than  $n^4$  nodes or dynamically changing set of nodes, like the P2P systems. In [7], the authors have presented a generalization of the multi-mesh interconnection topology, which relaxes this requirement. This modification termed as Generalized Multi-Mesh (GM) supports any number of nodes. In an GM comprising of  $N$  nodes with a block size of  $n, 1 \leq N \leq n^4$ , there are  $m = \lceil N/n^2 \rceil$  number of blocks, which are arranged in rows and columns. There are  $p = \lceil m/n \rceil$  rows, in which only the last row is partially complete with  $q = \text{mod}(m, n)$  blocks. All the other rows have  $n$  complete blocks, each block having  $n^2$  nodes. Only the last block in the last row may be incomplete. Four functions are used to describe the inter-block connections:

1)  $top(\alpha, \beta, i)$ : If the  $i^{th}$  column in the block has at least one node, then  $top(\alpha, \beta, i)$  is the node in row 1, column  $i$ . Otherwise, it is not defined.

2)  $bottom(\alpha, \beta, i)$ : If the  $i^{th}$  column in the block has at least one node, then  $bottom(\alpha, \beta, i)$  is the  $i^{th}$  column node in the highest indexed row that has at least  $i$  nodes. Otherwise,

it is not defined.

3)  $left(\alpha, \beta, i)$ : If the  $i^{th}$  row in the block has at least one node, then  $left(\alpha, \beta, i)$  is the node in row  $i$ , column 1. Otherwise, it is not defined.

4)  $right(\alpha, \beta, i)$ : If the  $i^{th}$  column in the block has at least one node, then  $right(\alpha, \beta, i)$  is the highest indexed column node in row  $i$ . Otherwise, it is not defined.

The inter-block connections are given by the following rules.

Rule 1.1) If the node  $top(x, \beta, \alpha)$  exists, then  $bottom(\alpha, \beta, x)$  is connected to the node  $top(x, \beta, \alpha)$ .

Rule 1.2) If the node  $top(x, \beta, \alpha)$  does not exist and the node  $top(\alpha+1, \beta, x)$  exists, then  $bottom(\alpha, \beta, x)$  is connected to the node  $top(\alpha+1, \beta, x)$ .

Rule 1.3) If the node  $top(x, \beta, \alpha)$  as well as the node  $top(\alpha+1, \beta, x)$  do not exist, then  $bottom(\alpha, \beta, x)$  is connected to the node  $top(i, \beta, x)$ , where  $i$  is the lowest indexed row for which  $top(i, \beta, x)$  is not connected using Rule 1.1).

Rule 2.1) If the node  $left(\alpha, x, \beta)$  exists, then  $right(\alpha, \beta, x)$  is connected to the node  $left(\alpha, x, \beta)$ .

Rule 2.2) If the node  $left(\alpha, x, \beta)$  does not exist and the node  $left(\alpha, \beta+1, x)$  exists, then  $right(\alpha, \beta, x)$  is connected to the node  $left(\alpha, \beta+1, x)$ .

Rule 2.3) If  $left(\alpha, x, \beta)$  as well as  $left(\alpha, \beta+1, x)$  do not exist, then  $right(\alpha, \beta, x)$  is connected to the node  $left(\alpha, i, x)$ , where  $i$  is the lowest indexed column for which  $left(\alpha, i, x)$  is not connected using Rule 2.1).

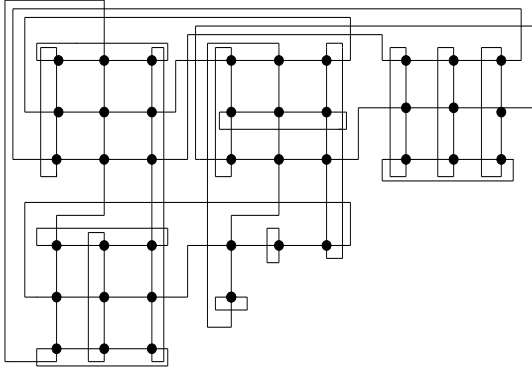


Fig. 2. A GM Network with 40 nodes,  $n=3$

A special case of GM is when  $N = n^4$ , in which case, it reduces to a CMM. Note that in this case, only rules 1.1) and 2.1) apply as  $top(x, \beta, \alpha)$  and  $left(\alpha, x, \beta)$  always exist.

Both GM and CMM have diameter of  $\Theta(N^{1/4})$ , whilst the per-node degree is four. This feature of the multi-mesh makes it very attractive for use in P2P systems.

#### IV. P2P NETWORK BASED ON MULTI-MESH

Our P2P Network design is centered around the generalized multi-mesh. Each node in the peer-to-peer network at a given instant of time assumes a particular position in this generalized multi-mesh and the network grows incrementally. Initially, the system designer sets a suitable value for the block size ( $n = N^{1/4}$ ).

#### A. Routing in a Multi-Mesh Network

The process of routing in a multi-mesh is described in [6]. In a multi-mesh, a block has connections to all the blocks in the same row or same column. Routing to a destination node in a block that is in the same row or same column as the source block (block containing the source node) is trivial. If the destination block is not in the same row or same column as that of the source block then, there is one intermediate block in the path between the source block and the destination block. We will scrutinize in some more detail how routing is done in this case.

To route a message from a source node  $(\alpha_s, \beta_s, x_s, y_s)$  to a destination node  $(\alpha_d, \beta_d, x_d, y_d)$ , first imaginary lines are drawn in the source block connecting the nodes  $(\alpha_s, \beta_s, 1, \alpha_d)$  &  $(\alpha_s, \beta_s, n, \alpha_d)$  and  $(\alpha_s, \beta_s, \beta_d, 1)$  &  $(\alpha_s, \beta_s, \beta_d, n)$ . Similar imaginary lines are drawn in the destination block connecting the nodes  $(\alpha_d, \beta_d, 1, \alpha_s)$  &  $(\alpha_d, \beta_d, n, \alpha_s)$  and  $(\alpha_d, \beta_d, \beta_s, 1)$  &  $(\alpha_d, \beta_d, \beta_s, n)$ . These lines divide the source block and the destination block into four quadrants each. The four boundary nodes through which these imaginary lines are drawn in the source block are called the source block exit points and the four boundary nodes in the destination block are called the destination block entry points. Now, the quadrants into which the source node and the destination node fall in their respective blocks are located. There are two exit points in the source block closest to the source node and two entry points in the destination block closest to the destination node. The two possible paths passing through these two pairs of entry points and exit points are enumerated and the path with the lower number of hops is selected. The intermediate block has an entry point and an exit point. A list consisting of the source block exit node, intermediate block exit node and the destination node is made and it is included in the routing message for routing purposes. Each node that receives this message tries to route the message to the node at the top of the list. Further, when the message reaches the exit nodes, the nodes remove their entry at the top of the list, so that the message travels towards the next item in the list and so on, until it reaches the destination. The P2P network construction is described in the next section.

#### B. P2P Network Construction

Four main algorithms are principal to the construction of all P2P systems in general. The first two algorithms describe the manner in which new nodes join and leave the P2P network. The next two describe the procedures for inserting a new object into the system and retrieving an object.

1) *Join algorithm*: The *key* in our case is the four tuple  $(\alpha, \beta, x, y)$ ,  $1 \leq \alpha, \beta, x, y \leq n$ , where  $(\alpha, \beta)$  represents the block position and  $(x, y)$  represents the node position in the block. Each node in the overlay network is associated with a pair of keys. The first one is called the Real Identifier (*RIId*), which is the position in the multi-mesh network that this node occupied when it initially joined the network. Recall that the GM grows incrementally. So, nodes joining successively in the network have consecutive *RIId*s. The second identifier termed

as the Virtual Identifier ( $VID$ ) is actually the key of the objects for which this node acts as the *home node*. Both  $RID$  and  $VID$  are of the form  $(\alpha, \beta, x, y)$  as mentioned earlier. Note that  $RID$  of a node depends on the state of network when the node joined the network. But,  $VID$  is constant for a particular node, obtained by hashing its unique identifier (for example, the machine IP address). Each node maintains a routing table, which has the neighbor  $RID$ s and their corresponding IP addresses. In addition, every node maintains a small 'key-table', which has the IP of the home node that has its  $VID$  same as the  $RID$  of this node. This key-table can be regarded as a pointer to the home node.

It is assumed that a new node wishing to join the P2P network knows at least one of the existing nodes in the network. This existing node in the network acts on behalf of the new node and issues the join request in the network. The join operation is presented in the following algorithm.

Step 1) The new node sends a  $join(VID, IP)$  request to one of the existing nodes in the network.

Step 2) The existing node finds the position of the new node in the network, which is the new node's  $RID$ . Recall that the multi-mesh network grows incrementally row-by-row in a block and, then block-by-block. If the current last block in the network is an incomplete block, then this position is the next position after the last node's position in this block, otherwise, it is the first node's position in a new block.

Step 3) Next, a message is routed towards the node in the network that has its  $RID$  same as the  $VID$  of the new node, informing it about the position of the new node. This step is required since all requests for objects published at the new node go to this node.

Step 4) Lastly, the routing table of the new node and its neighbors are updated.

2) *Leave algorithm*: When a node leaves the network, it leaves behind a void in the space it occupied in the multi-mesh network. This void is filled up by the current last node (with the highest  $RID$ ) in the network. The departing node inserts a special  $leave(RID, key-table)$  message into the network and exits. This special message is routed towards the final node in the network. The last node upon receiving this message, updates its state, informs existing and new neighbors about the changes and changes its  $RID$  to the  $RID$  in the message.

Node failures are handled in a similar way, except for the fact that the node failures have to be recognized by some node that tried routing a message to the failed node. This node then takes care of informing the last node in the network to fill up the void left behind by the failed node.

3) *Insert algorithm*: The operation of publishing an object in the system is the operation of inserting the object into its home node. To insert an object, an  $insert(key, object)$  message is routed towards the node that has its  $RID$  equal to key. The destination node upon receiving this message checks its key-table. If the key-table has an  $RID$  entry then, the object is sent to the node in the key-table, otherwise, the object is published in the destination node itself.

4) *Retrieve algorithm*: The node that wants to retrieve an object with a key, routes a  $retrieve(key)$  message towards the node whose  $RID$  is equal to the key. The destination node upon receiving this request checks its key-table to see if the home node for this object exists. If the home node exists, then the request is forwarded to the home node. If the home node does not exist, then the destination node checks its local repository for the object. If the object is found, the object is sent back to the requester, otherwise, an error message informing that the object does not exist in the system is sent to the requester.

### C. Simulation Environment and Results

Our P2P network is comparable to 2-dimensional CAN, since both the networks maintain a fixed per-node degree of four and given  $N$  nodes, both have the same number of overlay edges. While the CAN is based on the torus, our network is based on multi-mesh. In all our simulation experiments, we compare the performance of our network with CAN. Our simulation environment consists of a physical network of 5500 nodes modelled as a Transit-Stub topology using the GT-ITM topology generator [8]. Transit-Stub topologies have 2-level hierarchy of routing domains with transit domains that interconnect lower level stub domains.

1) *Performance Metrics*: Our evaluations focus on two main performance aspects.

- Average stretch
- Average hop-length

Stretch is defined as the ratio of the path-length on the overlay network to the path-length on the physical network. Stretch gives a notion of how *close* the topology of the overlay network is to the physical topology. This can be improved by a variety of techniques including *landmark ordering of nodes* [4]. In *landmark ordering of nodes*, every node is ordered in the overlay network according to its relative distance measured to a set of fixed nodes in the network called landmark nodes. Landmark ordering technique can be applied to a number of P2P systems including all those that we have discussed in this paper. We currently have not implemented landmark ordering in our P2P network, hence, all our measurements with CAN are without the landmark ordering technique.

2) *Simulation Procedure*: Given the physical network and the number of nodes in the overlay network ( $N$ ), we first construct two P2P networks, one based on multi-mesh and the other based on CAN. The same set of physical nodes participate in both the P2P networks. Next  $k$  number of random sender-receiver pairs are selected and for each pair, a message is routed from the source to the destination in both the networks. The stretch and the number of hops for each sender-receiver pair are measured in both the networks. The experiment is repeated for different values of  $N$ .

3) *Simulation Results and Discussion*: In all our experiments, we vary the number of overlay nodes from 81 to 4096 nodes. While it is possible to have a generalized multi-mesh of any number of nodes, we had a complete multi-mesh in all our experiments. CMMs are easier to construct than GMs and further, both have approximately the same performance

metrics. All the results that we obtained with CMMs would hold true even for GMs.

The first graph shows the average stretch as a function of the number of overlay nodes in the network.

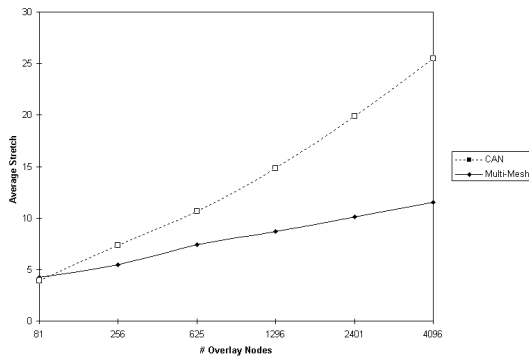


Fig. 3. Average Stretch

The graph of Fig. 4. shows the number of routing hops as a function of the size of the overlay network.

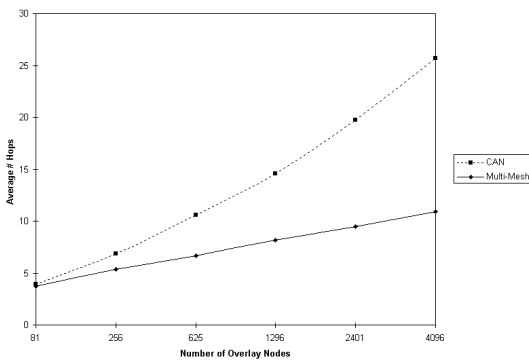


Fig. 4. Average Number of hops

The graph of Fig. 5. shows the variation of the maximum number of hops for the selected source-destination pairs in both CAN and multi-mesh P2P network.

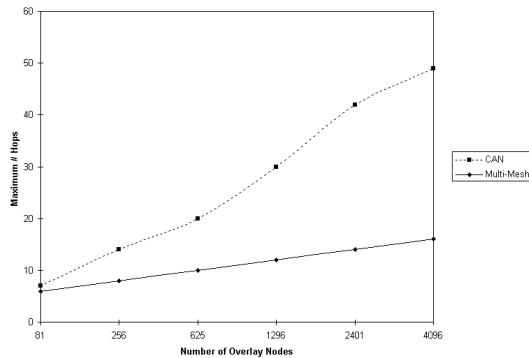


Fig. 5. Maximum Number of hops

It is evident from the results that the multi-mesh P2P networks provide some advantages over CAN networks. The

overlay topology of our P2P network is in a sense more *strongly connected* than in CAN. In large P2P networks typically consisting of several tens of thousands of nodes, multi-mesh P2P network would provide far better performance statistics than CAN.

## V. FUTURE WORK

Currently, the multi-mesh architecture exists only in two dimensions. However, it is possible to extend it to 3 dimensions and in general to  $d$  dimensions. A two-dimensional multi-mesh is superior to a two-dimensional torus in many aspects and a  $d$ -dimensional multi-mesh would be superior compared with its counterpart. A  $d$ -dimensional multi-mesh has  $2d$  number of neighbors. This gives us a clue that increasing the per-node state space, i. e., going to higher dimensions, would result in better routing performances. We have at present, extended the multi-mesh to 3-dimensions. Due to the space limitation in this paper, we will not be discussing the details of it here.

The effect of concurrent node failures in a multi-mesh P2P network is yet to be determined. The number of simultaneous node failures that would seriously impact the performance of the network would help us understand the resilience of the multi-mesh P2P network.

## VI. CONCLUSION

We have outlined the design and implementation of a new P2P network based on the multi-mesh architecture. The system supports dynamic node arrivals and departures. The evaluations demonstrate the advantages of this network over CAN. Analysis shows us that while still maintaining the same number of nodes and edges as CAN, this network provides better connectivity. This is indeed a unique property.

## ACKNOWLEDGMENT

The authors would like to thank Dr. Sylvia Ratnasamy for providing us with the source code for the CAN simulations.

## REFERENCES

- [1] A. Rowstron and P. Druschel, "Pastry: Scalable, Decentralized Object Location and Routing for Large-scale Peer-to-Peer Systems", *Proceedings of IFIP/ACM International Conference on Distributed Systems Platforms (Middleware 2001), Heidelberg, Germany*, November, 2001.
- [2] B. Y. Zhao and J. Kubiatowicz, "Tapestry: An infrastructure for fault-tolerant wide-area location and routing", *Technical Report*, UCB CSD 01-1141, University of California at Berkeley, Computer Science Department, 2001.
- [3] I. Stoica, R. Morris, D. Karger, F. Kashoek and H. Balakrishnan, "Chord: A scalable Peer-To-Peer lookup service for internet applications", *Proceedings of ACM SIGCOMM '01, San Diego, USA*, August, 2001.
- [4] S. Ratnasamy, P. Francis, M. Handley, R. Karp and S. Shenker, "A Scalable Content-Addressable Network", *Proceedings of ACM SIGCOMM '01, San Diego, CA*, August, 2001.
- [5] G. Plaxton, R. Rajaram and A. Richa, "Accessing Nearby Copies of Replicated Objects in a Distributed Environment", *Proceedings of ACM SPAA*, June, 1997.
- [6] D. Das, M. De and B. P. Sinha, "A New Network Topology with Multiple Meshes", *IEEE transactions on Computers*, vol. 48, no. 5, May 1999.
- [7] A. Sen, S. Bandyopadhyay and B. P. Sinha, "A New Architecture and a New Metric for Lightwave Networks", *Journal of Lightwave Technology*, vol. 19, no. 7, July 2001.
- [8] E. Zegura, K. Calvert and S. Bhattacharjee, "How to Model an Inter-network", *Proceedings of IEEE Infocom '96, San Francisco, CA*, May, 1996.