TAC: A Topology-Aware Chord-based Peer-to-Peer Network

Javad Taheri, Mohammad Kazem Akbari

Advanced Information Technologies Lab.
Department of Computer Engineering and IT
Amirkabir University of Technology, Tehran, Iran
{j.taheri, akbarif}@aut.ac.ir

Abstract:

Among structured Peer-to-Peer systems, Chord has a general popularity due to its salient features like simplicity, high scalability, small path length with respect to network size, and flexibility on node join and departure. However, Chord doesn't take into account the topology of underlying physical network when a new node is being added to the system, thus resulting in high routing latency and low efficiency in data lookup. In this paper, we introduce the TAC, a novel topology-aware protocol which is based on Chord. TAC introduces the local ring concept by dividing the geographical space into smaller areas. Through binding each new node to a proper local ring concerning its physical location, TAC considers the physical network topology of the overlay network to demonstrate more efficient key lookup. Simulation results show that TAC performs better in terms of more efficient routing and less bandwidth usage.

Keywords: Peer-to-Peer Systems, Routing Efficiency, Topology Awareness, Chord

1. Introduction

PEER-TO-PEER (P2P) networks are types of distributed systems with neither centralized control nor hierarchical organization. These systems are overlay networks which are built on top of physical layer network. Every node in this systems is a self-organizing peer which is logically connected to the network through its neighbors. An important feature of P2P systems is that each node can operate either as a client or a server. Due to this feature, P2P systems are considered to be a good substrate for developing many applications in a variety of fields like data sharing, content distribution, distributed programming and so forth 00. However, the Initial P2P systems like Napster 0 and Gnutella 0 have the problem of weak scalability when the number of nodes grows. To solve this problem, a new type of P2P systems called structured P2P systems which is based on using Distributed Hash Tables (DHTs) was introduced. Notwithstanding acceptable scalability, many of DHT-based P2P systems suffer from topology mismatch between overlay and physical network which would lead to inefficient routing of messages.

Up to now, many structured DHT-based P2P systems have been proposed. Chord 0, CAN 0, Pastry 0 and Tapestry 0 are the most well known. Among these systems, Chord has achieved wide popularity for its noticeable features like simplicity, high scalability, and flexibility in frequent node arrivals and departures. However, it is known that this protocol has low efficiency in data lookup. This problem comes from the fact that Chord does not consider underlying physical topology to build the overlay network and results in mismatch between nodes' adjacency in the physical and overlay networks.

The main contribution of this paper is introduction of TAC (Topology Aware Chord) algorithm. TAC is a modified version of Chord which considers physical network topology through dividing the geographical space in which the nodes are distributed, into smaller zones and then binding the nodes within each zone to a local ring. The local ring of a zone lets the nodes inside that zone to become aware of each other's existence. At the cost of keeping more information, TAC allows every node to be familiar with other proximate nodes and do a better routing by selecting the next hop more appropriately. As a result, the average distance that must be traversed by each message is significantly reduced. Moreover, less bandwidth is required in average to route a query from its source to the destination. In other words, TAC exploits underlying network topology information to perform better message routing.

The rest of this paper is organized as follows: Section 2 briefly discusses Chord protocol and previous works related to topology awareness issue. In section 3, the proposed protocol is presented in detail and the experimental evaluation is given in section 4. Finally, concluding remarks are presented in section 5.

2. Background

A structured P2P network is constructed of some computer nodes and a set of data in form of {key, value} pairs. Each node is responsible for maintaining some of the pairs in a way that the following requirements are satisfied:

1) every node should keep almost the same number of pairs
2) Every node should be able to lookup the value of a given key by searching a relatively small and bounded number of other nodes

Therefore, there should be some mechanisms to solve following problems:

- Distribution of data evenly among the overlay nodes
- Finding the node which is responsible for a given key by traversing a bounded and small number of other nodes

2.1. The Chord Protocol

Chord 0 assigns the responsibility of each {key, value} pair to the proper node using the consistent hashing method. In this protocol, an *m*-bit identifier is assigned to each node and key by hashing its name or IP_address. The resulted set of identifiers forms a modulo 2^m one-dimensional circular space (Fig. 1).

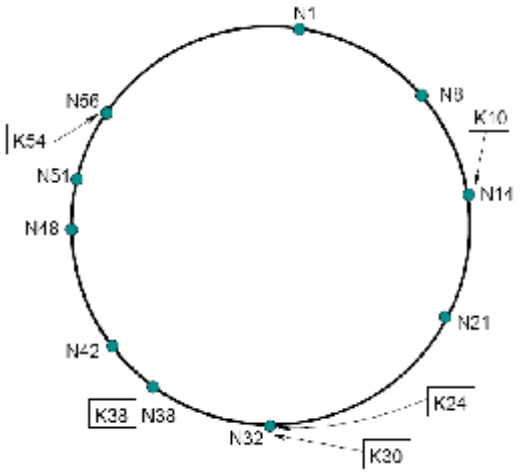


Fig. 1: Assigning keys to appropriate nodes in Chord0

According to global ring, every key is assigned to the first peer whose identifier is equal to or follows the key. This scheme tends to balance the load on the system, since each peer receives approximately the same number of keys 0. The other important property is that with high probability, when a new node joins or leaves the network, only a small number of the keys has to be moved.

In Chord, to lookup a key and retrieve it's value, every node keeps a *finger table* which is comprised of at most *O*(*Log N*) records(fingers), where *N* is the total number of overlay nodes. Every finger keeps the identifier and IP address of an overlay node. The j^{th} finger of node *i* is the first node that succeeds *i* by at least 2^j in the identifier space, where $0 < j < m$ and *m* is number of identifier bits. As a result, the finger table contains more nearby nodes rather than faraway nodes at a doubling distance.

Using the finger table, Chord uses *finger routing* to forward lookup messages. The node *i* looks up the key *k*, by forwarding a lookup message to the node whose identifier most immediately precedes the successor node of *k* in the finger table. Each node repeats this process and the message gets closer and closer to the destination and finally reaches the destination.

2.2. Topology Awareness in P2P Overlays

A P2P system is a logical network which is built on top of the underlying physical network. Each node in this system communicates with other nodes through its neighbors. Therefore, every node needs to be bound to its neighbors at its arrival time. The selection process of neighbor nodes is an important task. If the neighbors of a node are selected on the basis of its physical topology, the quality of message routing between overlay nodes can be improved. In other words, if overlay neighbors of a node are fairly its neighbors in physical layer, the distance between two communicating overlay nodes is less. Furthermore, as a result of coincidence of overlay and physical neighbors, the messages transmission is faster and consequently the network traffic will be reduced.

Consider a sample physical network with four nodes (*A*, *B*, *C* and *D*) which is shown in Fig. 2. The numbers shown next to each edge, indicate the physical distance between two nodes. Fig. 3 shows two overlay networks which are built using these four nodes. The difference between two overlays is that each is built using different neighbors. In the overlay of Fig. 3(a), if *A* send the message *m* to *B*, the logical path

A→C→B

must be traversed (suppose a clockwise unidirectional communication). Mapping this path to the physical topology would result in:

A→R1→R2→C→R2→R1→B

physical path. Therefore, the message *m*

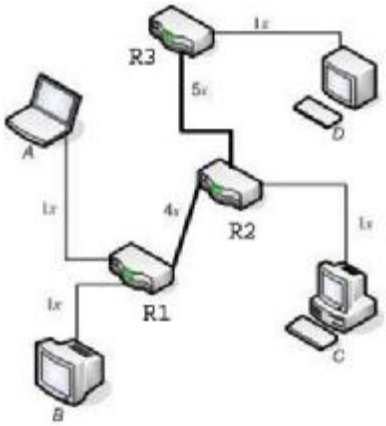
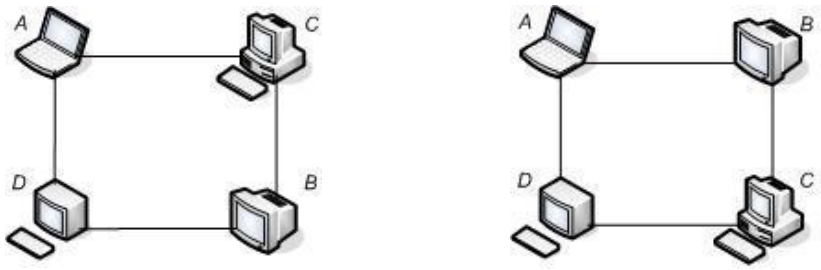


Fig. 2: A physical network which connects 4 computer nodes A, B, C and D through routers R1, R2 and R3. The edge between every two nodes depicts a physical link with its distance label.



(a) (b)

Fig. 3: Two different overlay networks from nodes of Fig. 2.

has to walk through a physical path with following length:

$$Dist_A = 1x + 4x + 1x + 1x + 4x + 1x = 12x$$

Similarly, the logical and physical path of the overlay network shown in Fig. 3(b), would be $A \rightarrow B$ and $A \rightarrow R1 \rightarrow B$ respectively and results in the following physical distance:

$$Dist_B = 1x + 1x = 2x$$

which is significantly shorter. The comparison of these two distances shows that the overlay network of Fig. 3(b) is more congruent with the underlying physical network than the overlay of Fig. 3(a)0. Of this, it can be concluded that in an overlay network, awareness of underlying physical network topology will decrease the length of traversed path and leads to more efficiency in routing and bandwidth usage.

2.3. Related Works

In general, the Chord protocol provides support for just one operation: given a key, it maps the key onto a node 0. Having this in mind, the topology of underlying physical network is not considered in Chord. As a result, this protocol suffers from inefficient routing and high latency in data lookup. To solve these problems, many techniques have been proposed recently to take account of physical network topology in Chord routing.

In this regard, PChord 0 presents a better awareness of physical topology than the original Chord by using *proximity lists*. The proximity list of a node is a list of geographically close nodes which the node discovers in its lifetime. The next hop is decided by the entries in both finger_table and the proximity list. Although this approach achieves better routing efficiency than Chord while keeping lightweight maintenance costs, it suffers from slow convergence and inefficiency in the case of churn, where the lifetime of a node in the overlay is relatively short [9].

Chord6 [10] is another protocol that deals with physical network topology in Chord. By utilizing hierarchical structure of IPv6 addresses, Chord6 assigns an identifier to each node in a way that the nodes in the same domain have close identifiers. However, the nodes in two close domains may have very different identifiers. Although this approach is simple and can reduce the average path length, however, because of large number of Internet domains and small number of overlay nodes in same domain, Chord6 seems to

be inefficient. Furthermore, the current IPv4 based physical networks limit the implementation of this protocol.
AChord 0 is a modified version of Chord which takes account of physical network topology in the overlay network using anycast mechanism. Anycast is one of the communication methods in IPv6 in which an address is given to a group of nodes. When a node from outside the group sends a message to this address, the message would be delivered to the nearest node in that anycast group. AChord considers all nodes in the overlay network as an anycast group. If a new node decides to join the overlay network, it sends a request to the anycast address of this group and then is connected to the nearest node in the overlay. However, besides the good routing efficiency, AChord assumes that the underlying network supports an ideal anycast i.e., the outside messages are always routed to the nearest node which in the current Internet, this is not always feasible. Moreover, similar to the Chord6 mechanism, AChord is designed on the basis of IPv6 protocol which is not widely operational yet.

3. Proposed Protocol

In this section, the proposed protocol will be discussed. First we give a brief overview of TAC.

3.1. Overview

The key idea behind TAC is to divide the geographical space in which the overlay nodes are located into smaller areas and then introduce the nodes inside each area to each other. TAC assumes that if geographically proximate nodes are aware of each other's existence, the lookup messages will traverse smaller paths and the efficiency of routing will be significantly increased.
Speaking in more detail, TAC improves the topology awareness of Chord by dividing the geographical space into smaller areas called zones. When a node joins the overlay, TAC introduces other nodes within same zone to this node by binding it to the zone's local ring. Moreover, each node is responsible for maintaining another finger table which is related to the local ring nodes. This table, which is called *zone finger table*, is identical to original finger table in terms of formatting and the completion procedure, except that this table deals only with local ring nodes. The aim of this table is to maintain the information of proximate nodes (the nodes within same zone) and therefore let the lookup process to be done more efficiently.
When a key lookup request is received in node *n*, it first looks for the entries in the zone finger table. If an appropriate node is found, it will be selected as the next hop. Otherwise, the next hop will be selected using the information of finger table using Chord's original lookup algorithm.

With this in mind, the detailed information of TAC, i.e. the process of geographical partitioning, local ring, zone finger table and routing algorithm, will come next.

3.2. Geographical Areas

In Chord, the locality of the overlay nodes does not influence the key location routing. As we mentioned before, this lack of topology information results in poor efficiency in routing. To solve this problem, TAC partitions the nodes' geographical space into smaller areas called *zones* and assigns every zone, the nodes which are enclosed in its boundaries. The size of each zone could be variable and we don't set any precise rule here to determine each zone's boundaries, because some non-engineering factors like politics or natural conditions may affect the size of each zone in the real world. However, we observed that there is a trade-off between the number of overlay nodes in a zone(which depends on zone's size) and the worst-case latency among them. It is desirable to have more nodes within a zone with minimum worse-case latency.

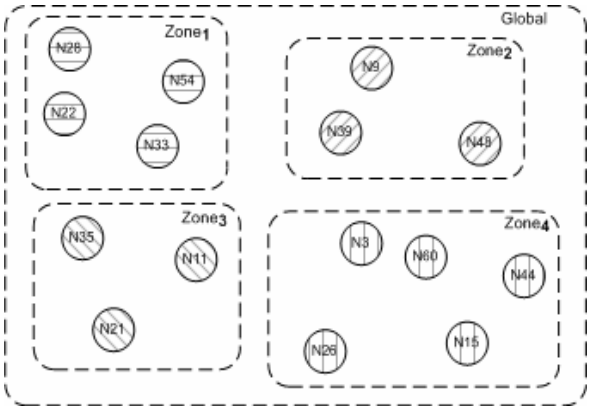


Fig. 4: Geographical space which is divided into 4 zones

A fine granularity in space partitioning, i.e. creating small zones, will reduce the worst-case latency but will reduce the number of nodes. In the other hand, increasing the size of each area guarantees having more overlay nodes inside each zone, but will increase the worst-case latency. In this paper, we divide the space into N_z same-size zones where N_z is a variable. Fig. 4 shows a hypothetical global geographical space which is divided into 4 zones (Zone 1-4). Every node is depicted as a circle and the pattern of all nodes inside a zone is the same.

3.3. Local Rings

In Chord, an *m*-bit identifier is assigned to every overlay node by hashing its IP address. The resulted set of identifiers, forms a modulo 2^m one-dimensional circular space. In this paper, we call this circular space the *global ring*, because all overlay nodes inside all zones are participated. In addition to the global ring, TAC forces the

overlay nodes of each zone to form another circular space called *local ring*. The main difference between the global and the local rings is that the former contains all of nodes but the latter is comprised of only the nodes of the related zone. Fig. 5 shows the global ring of the overlay nodes shown in Fig. 4. Moreover, the local ring of zone 4 is also depicted in dotted line.

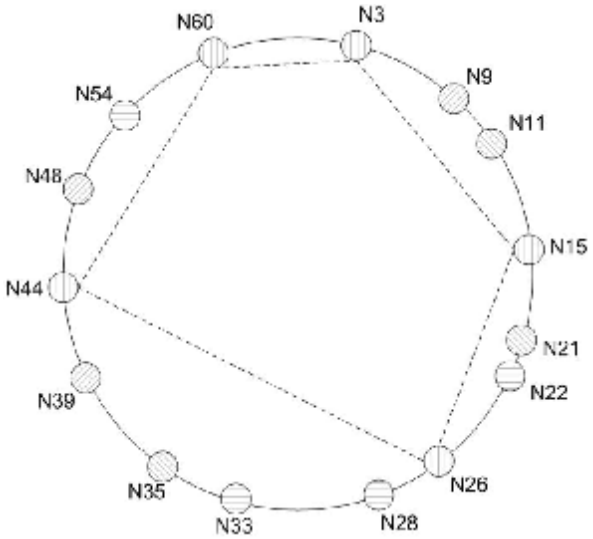


Fig. 5: Global and local rings

Chord keeps the information of global ring in a directory server. When a new node n_{new} decides to join the network, it sends a join request to the directory sever. Upon receiving the request, the server assigns an identifier to n_{new} by hashing its IP address and then registers it in the proper position in the global ring regarding its identifier. Using the global ring information, directory server also sends the IP addresses of successor and predecessor to n_{new}. Having these addresses, n_{new} will be able to gradually get familiar with other overlay nodes and build its finger table.

In TAC, the directory server has slightly more responsibility. Besides keeping the information of global ring, it also keeps the information of geographical space partitioning and connectivity information of all local rings.

In other words, the directory server is aware of all zones and their scopes. When n_{new} decides to join the overlay network, it sends its geographical coordinates to the server along with the join request. The directory server then registers n_{new} with the global ring using same process in Chord. Moreover, by using the n_{new} 's location coordinates, the server determines the zone in which n_{new} is located and registers n_{new} with the local ring of that zone. Like the global ring, the IP address of successor and predecessor nodes of n_{new} in the local ring will also be sent to n_{new}. This information will be used to build complete the zone finger table gradually.

3.4. Zone Finger Table

Each node in Chord keeps a hash table in order to determine the next hop during key location. This table is called *finger table* and maintains information of about $O(logN)$ other nodes where N is the number of overlay nodes in the global ring. To construct this table, every node first gets the IP address of its successor node in the global ring from the directory server and gradually finds other fingers and inserts their information in the table over the time.

In addition to the finger table, in our proposed protocol every node keeps another hash table called *zone finger table*. This table is dedicated to keep the information of some proximate nodes which are registered in local ring. Speaking more specifically, It maintains the information of about $O(LogM)$ overlay nodes, where M is the number of overlay nodes in the same zone. This table gets initialized by the IP address of successor node in the local ring. The other procedures are the same as those are used to build the finger table in original Chord.

Like the finger table, this table is used to find the proper next hop when there is a key location request. The detailed information is presented in the next subsection.

3.5. Key Location Procedure

When a key location request arrives at a node, the node first checks to see if the value of requested key is stored in its own storage space. If not, the request will be forwarded to the next hop. In Chord, the next hop is selected using the available information in finger table and is the finger that has the closest identifier preceding the data key.

In TAC protocol, the procedure for selecting next hop is slightly different. To find the proper next hop, every node first checks the identifier of requested key to see if it is smaller than the identifier of *zone_successor* (the node which is successor of current node in the finger table). If yes, the next hop is selected using the original procedure in Chord by selecting the closest preceding node from finger table. Otherwise, if the key's identifier is greater than or equal to the identifier of zone_successor, the next hop is selected by finding closest preceding node from zone finger table. The pseudocode for this procedure is shown in Fig. 6. The size of finger table and zone finger table are assumed to be l and m respectively.

```
// ask node n to find the successor of id
n.find_sucessor(id)
  if ( id ∈ (n, zone_ successor))
    n_next = closest_preceding_node(id);
  else
    n_next = closest_zone_preceding_node(id);
  return n_next.find_successor(id);

// search the finger_table for the highest predecessor of id
n.closest_zone_preceding_node(id)
  for i = m downto 1
    if (zone_finger[i] ∈ (n; id))
      return zone_finger[i];
  return n;

// search the zone_finger_table for the highest predecessor of id
n.closest_preceding_node(id)
  for i = l downto 1
    if (zone_finger[i] ∈ (n; id))
      return zone_finger[i];
  return n;
```

Fig. 6: Pseudocode for selecting next hop in TAC

Let's clarify the procedure by an example. Referring to the global and local rings shown in Fig. 5, assume that a key location request which the identifier of its key is 24 arrives in node *N3*. The node *N3* first compares the key with its zone_successor (here is *N15*) and finds that the key is larger. Therefore, it uses the *closest_zone_preceding_node*() procedure and finds that *N15* is the proper next hop (closest preceding node in the zone's finger table) and forwards the request to this node. In the other hand, when node *N15* receives this request, finds that the requested key is smaller than its zone_successor (*N26*) and uses *closest_preceding_node*() to find the proper next hop. This operation will be repeated as long as the responsible node for this key is not found.

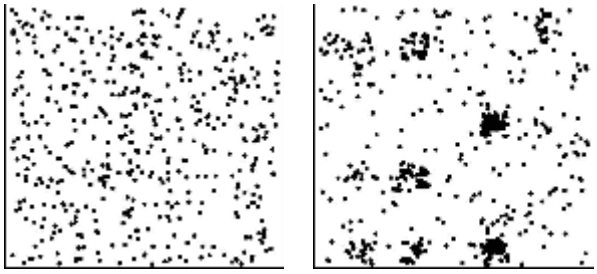
4. Experimental Results

In this section, we evaluate the routing efficiency of TAC by means of experimental results which we have obtained by simulation. First, we describe the simulation environment in which the experiments are done. Second, to evaluate the proposed protocol, we used three metrics and compared the experimental results with Chord. These metrics are Distance Ratio (DR), Bandwidth Usage, and Hop Number which are discussed respectively.

4.1. Simulation Environment

We implemented TAC protocol in Java, using SimJava 0 simulation library. To produce the network topology for our simulation, we used Brite 0 which is a well known representative internet topology generator. Brite gets some parameters about the topology from user and then gives the physical location of nodes in a square-shaped plane, under different placement models. These input parameters are the number of nodes *n*, the size of square side *A*, and the distribution model *M* under which the topology will be generated. In our experiments, we used two distribution models: *random* and *heavy-tailed*. In the random distribution model, every node is placed in a random location in the plane. However, in the heavy-tailed model, placement of nodes is done by focusing on some particular points. This model of distribution is more compatible with real world topologies since heavy-tailed distributions have been observed in the context of topological properties of Internet 0. Fig. 7 shows two sample network topologies generated by Brite using random and heavy-tailed distribution models.

We simulated TAC protocol using both random and heavy-tailed topologies with *n*=1000 and *A*=1000. Furthermore, we divided the geographical space into different number of zones. The distance metric is based on topological distance between every two nodes on the plane. We distributed 2000 key among nodes using consistent hashing and set every node to submit a lookup query in every 100 milliseconds for 100 times. Furthermore, we divided the geographical space (plane in the Brite) into different number of zones (form 1 to



a) random distribution **b) heavy-tailed distribution**

Fig. 7: Distribution of nodes by using two different models in Brite topology generator

1600) and evaluated TAC in all of them. In our experiments, the size of all zones is the same.

4.2. Distance Ratio

The first metric we used to evaluate TAC, is Distance Ratio (DR): the ratio of the distance which a lookup query traverses to reach its destination node to the distance between its source and destination 0. It is clear that the

more efficient a routing protocol is, the less DR its packets have. In an ideal case, the traversed path will be equal to the distance between source and destination and thus DR will be 1.

Fig. 8 shows the average DR of lookup queries in Chord and TAC when the geographical space is divided into different number of zones. The experimental results which are shown in Fig. 8(a) and (b) are obtained by using the topologies which are created with random and heavy-tailed distribution models respectively. The DR of Chord is the same in all zones and shown in the diagram. As it can be seen, DR of both Chord and TAC are equal when the number of zones is 1. This is predictable since TAC with just 1 zone presents Chord. Furthermore, as the number of zones increases, the DR of TAC decreases. By dividing the geographical space into an optimum number of zones (10 in random model and 16 in heavy-tailed model), the DR reaches its minimum value. After this optimum point, the number of zones has a reverse effect on DR. The reason is that when the number of nodes of a zone decreases; the knowledge of nodes about their proximate nodes becomes lesser and the DR begins to increase. The numerical values shown in Table 1 indicate that by dividing the geographical space into an optimum number of zones, TAC reduces the average DR of Chord by %29.1 and %31 in random and heavy-tailed topologies respectively.

4.3. Bandwidth Usage

Bandwidth usage is the second metric which is used to evaluate the proposed protocol. Since TAC is more efficient in key location, it is expected to occupy less bandwidth during key lookups. To prove this claim, we considered the Average number of lookup Queries which are in Transit in the network (AQT) during key location. Fig. 9 shows the experimental results about AQTs which were obtained by partitioning the geographical space into different number of zones and using both random (Fig. 9(a)) and heavy-tailed (Fig. 9(b)) topology models. Similar to DR metric, the diagrams indicate that there is an optimum number of zones in which the bandwidth usage has its minimum value. Referring to Table 1 we can find that in comparison to Chord, TAC saves the bandwidth by %23 at least.

4.4. Hop Number

The third metric we are going to talk about is the average number of hops which every lookup query traverses to reach its destination. Despite two previous metrics in which TAC surpasses the Chord, Fig. 10 shows that using TAC slightly increases average number of hops a query has to traverse. However this increase is negligible and according to Table 1 is less than %1.5.

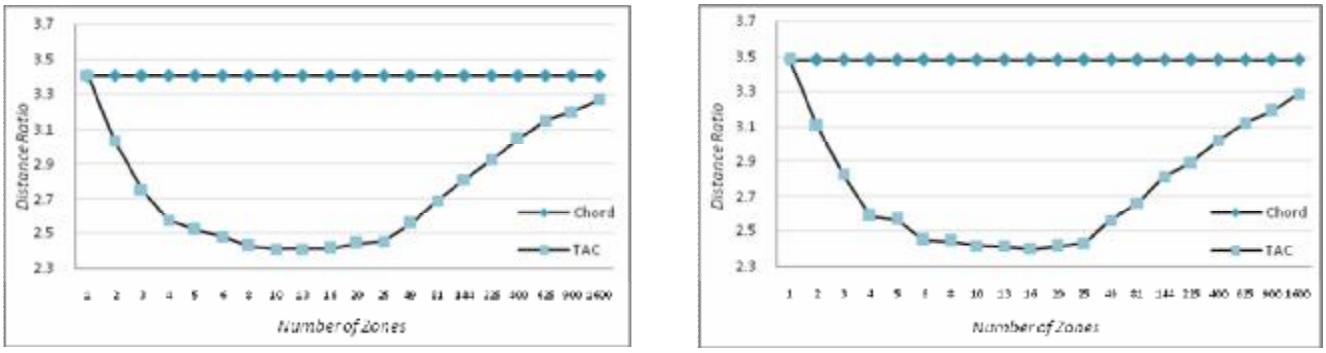
TABLE 1:
Comparing experimental results of Chord and TAC when the number of zones is optimum

Topology model	Optimum number of zones	Chord Distance Ratio	TAC Distance Ratio	Distance Ratio Change	Chord Bandwidth Usage (query/second)	TAC Bandwidth Usage (query/second)	Bandwidth Usage Change	Chord Average Hop Number	TAC Average Hop Number	Hop Number Change
Random	10	3.40	2.41	- %29.2	20976	16506	- %21.3	6.85	6.96	%1.5
Heavy-	16	3.48	2.4	- %31	21417	16307	- %23.8	6.85	6.95	%1.4

5. Conclusions

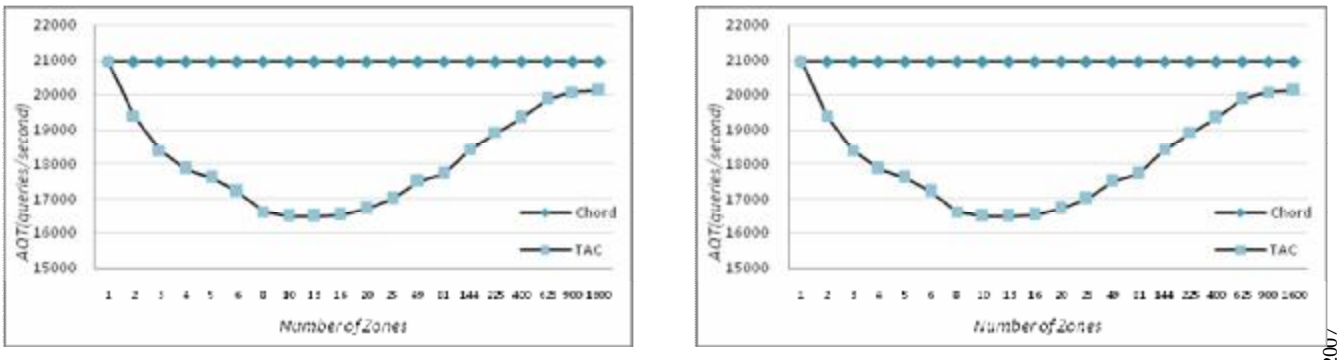
Due to its lack of physical network topology information, Chord suffers from low routing efficiency, and ineffective use of bandwidth in data lookup. To alleviate this problem, we proposed TAC, a topology-aware Peer-to-Peer system which is based on Chord. In TAC, the geographical space is divided into smaller zones and the nodes within each zone are logically connected to each other. At the expense of storing the information of additional nodes, TAC improves routing efficiency of lookup queries. Our simulation results were promising and showed more efficient routing and less bandwidth usage.

In our future work, we will focus on methods of calculating the optimum number of zones which would result in least average DR of queries. Furthermore, we are going to propose a mechanism to use TAC as an efficient local caching structure in P2P applications.



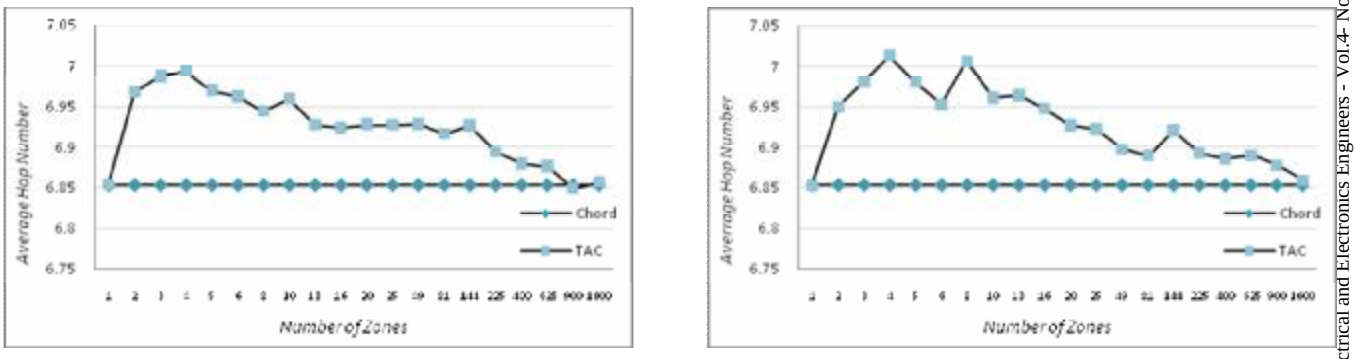
(a) (b)

Fig. 8: Average Distance Ratio of lookup queries in different number of zones;
a)using random topology b) using heavy-tailed topology



(a) (b)

Fig. 9: Average number of lookup Queries in Transit in the network(AQT) in different number of zones;
a)using random topology b)using heavy-tailed topology



(a) (b)

Fig. 10: Average number of hops traversed by lookup queries in different number of zones;
(a) using random topology (b)using heavy-tailed topology

References

[1] Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup protocol for internet applications", IEEE/ACM Transactions on Networking, vol. 11, no. 1, pp. 17–32, 2003.

[2] Lua E. K., Crowcroft J., Pias M., Sharma R. and Lim S.,"A Survey and Comparison of Peer-to-Peer Overlay Network Schemes", IEEE Communication survey and tutorial, March 2004.

[3] Napster. [Online]. Available: http://www.napster.com/

[4] Gnutella development forum, the gnutella v0.6 protocol. [Online]. Available: http://groups.yahoo.com/group/the gdf/files/

[5] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content addressable network", in Processings of the ACM SIGCOMM, 2001, pp. 161–172.

[6] Rowstron and P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems", in Proceedings of the Middleware, 2001.

[7] Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. D. Kubiatowicz, "Tapestry: A resilient global-scale overlay for service deployment", IEEE Journal on Selected Areas in Communications, vol. 22, no. 1, pp. 41–53, January 2004.

[8] Rostami H., Habibi J."Topology awareness of overlay P2P networks" Journal of Concurrency and Computation: Practice and Experience, InterScience, 2006

[9] F. Hong, M. Li, J. Yu, and Y. Wang, "PChord: Improvement on chord to achieve better routing efficiency by exploiting proximity," in Proceedings of the 25th IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW'05), June 2005.

[10] J. Xiong, Y. Zhang, P. Hong, and J. Li, "Chord6: IPv6 based topology-aware Chord," in Proceedings of the Joint International Conference on Autonomic and Autonomous Systems and International Conference on Networking and Services (ICAS/ICNS 2005), Aug 2005

[11] Dao L. H., Kim J.,"AChord: Topology-Aware Chord in Anycast-Enabled Networks", *IEEE International Conference on Hybrid Information Technology (ICHIT'06)*, 2006.

[12] F. Howell and R. McNab, "*SimJava: A Discrete Event Simulation Package For Java With Applications In Computer Systems Modelling*", First International Conference on Web-based Modelling and Simulation, San Diego, CA, Society for Computer Simulation, January 1998.

[13] Brite, 2003. http://www.cs.bu.edu/brite/ December 2005

[14] Mirrezaei S. I., Shahparian J., Ghodsi M. "RAQNet: A topology-Aware Overlay Network", *A.K. Bandara and M. Burgess (Eds.): AIMS 2007*. Springer 2007.