International Journal of Advance Engineering and Research Development (IJAERD)
Volume 1,Issue 5,May 2014, e-ISSN: 2348 - 4470 , print-ISSN:2348-6406

Distributed Network Routing Algorithms Table for Small World Networks

Mudit Dholakia[1]

[1]*Department of Computer Engineering, VVP Engineering College, Rajkot, 360005, India,*
Email:muditdholakia@gmail.com

Abstract- In this paper we present more efficient routing algorithm for small world network using the distributed table. In the scheme proposed, use of the network information will make the path finding between the nodes in the small world network efficient. In this scheme the network also shares the network information amongst the nearby nodes in a way that the memory space used to manage routing table for the network at each router will be less.

Keywords- Small World Network, Routing Algorithm, Distributed Network Table,

I. INTRODUCTION

The word 'Small World Phenomena' first came to existence from Milgram's famous experiment on how people are connected to each other and how there exist surprisingly short paths between peoples [3]. Milgram in his experiment showed that any person in the USA is only six relationships away from any other person. This phenomenon is known as 'six degrees of separation'. Milgram also showed that people are remarkably good at finding short routes without the knowledge of the global situation. After the growth of dense computer networks and social networks, this phenomenon also attracted lots of researchers to work on that. Small world networks exhibit properties different from simple networks and there for routing in these networks are also very different. There are many algorithms designed for routing in small world network, almost all of them uses greedy approach to find the target nodes [1]. Watts and Strogatz proposed a very refined model based on a class of random networks with edges of network divided into 'local' and 'long range' contacts [6]. The famous Kleinberg model showed that a greedy algorithm, which uses only local information to construct a path with average $(log\ n)^2$ steps [2]. However Kleinberg assumed in his model that, individual node has information about its own coordinates as well as coordinates of the target [2]. In practical the nodes may be unaware of anything but their immediate neighbor [5]. Oskar Sendberg showed in his article that it is even possible to implement the Kleinberg's model with only information about immediate neighbors [4].

A. MOTIVATION

However a lot work has been done on routing in small world network. We saw that most of them are based on the greedy approach in which nodes only have local information. Others approaches with node having network table are quite unexplored. In the real world or social networks, it is mandatory to use the greedy approach as an individual cannot know which persons another person might know, but in router we can use a network table to find a more efficient path. Motivated from this approach we have designed a new strategy. In which node entries are shared among nearby nodes.

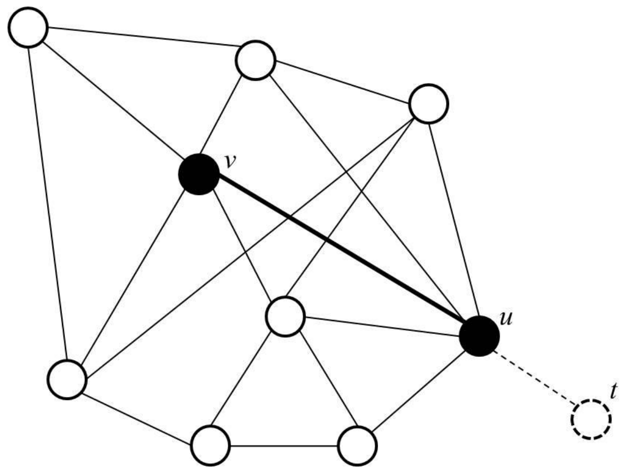
B. CONTRIBUTION

Our contribution will be an efficient algorithm with for routing in small world network with use of distributed network table. When working with large scale small world networks, distributing the table will not only reduce the memory consumed in each router, but will also reduce the routing time that we will see in upcoming sections. Our model is designed for small world networks but can also be applied in other large scale networks with minor modification.

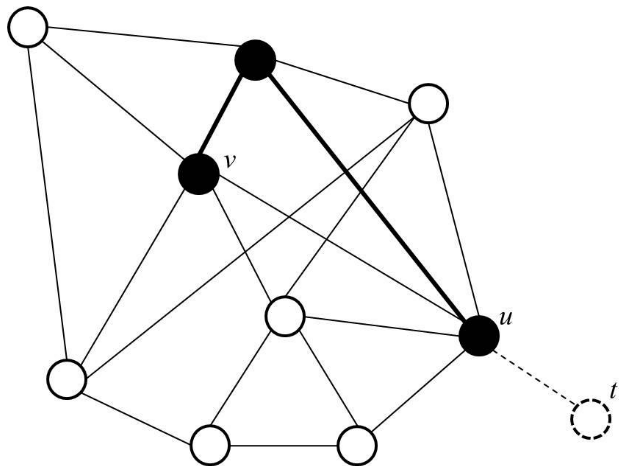
II. ALGORITHM WITH DISTRIBUTED TABLE

Instead of applying greedy strategy for small world network we propose to maintain a distributed network table with entries for every node in the network. The table is itself distributed among nearby nodes thus not requiring much memory in a single router.

In each table shared among nearby nodes there is an entry for each and every node in the network which contains the address of the next node in the path. When packet with *t* as target node arrives on any node *v* it searches the entry for *t* in the table stored in itself and stored in nearby routers. If the entry is found in its own table, it forwards the packet to the address found in table. If the entry is found in table of any nearby router *u* it forwards the packet to *u*, which then forwards it to the address found in table. Figure 1 shows this process. We will see the details of this process later in this paper.



(a) Entry Found in router itself. Router directly



(b) Entry found in neighbor

Figure.1. Response on entry found

II. ROUTING IN THE NETWORK

Now we will see how routing takes place in network with distributed table. We know that with the best greedy algorithm can only archive the path of order $log^2 n$ [1]. Here we will see that how we can archive better path with use of distributed table.

A. NEIGHBORHOOD OF A NODE

Here we will use term neighborhood of a node as the group of node which is sharing the same network table with that particular node. The size of the neighborhood of a node can be defined as the number of hops between that node and the farthest member of the neighborhood

B. ASSUMPTIONS MADE FOR THE IDEAL MODEL

1. The distribution table is fully updated with the change in the network.
2. The distribution table is shared in such a way that for every node (router) the union of all the entries of its neighbors should be perfect disjoint of its own entries.
3. Every node only contains the entries of nodes which should be nearest to it. In other word, if node v contains the table entry for a target node t and is away from t by distance d then no other node in neighborhood should have distance to t less than d.

These assumptions may seem like somewhat overdemanding, but they are made for ideal model. In practical, partial violation of these assumptions will also lead to a very efficient model.

Theorem 1: If any node v is at minimum distance d from target node t, then any node with distance r from v. If the minimum distance of that node from t is x then

$$d - r \leq x \leq d + r$$

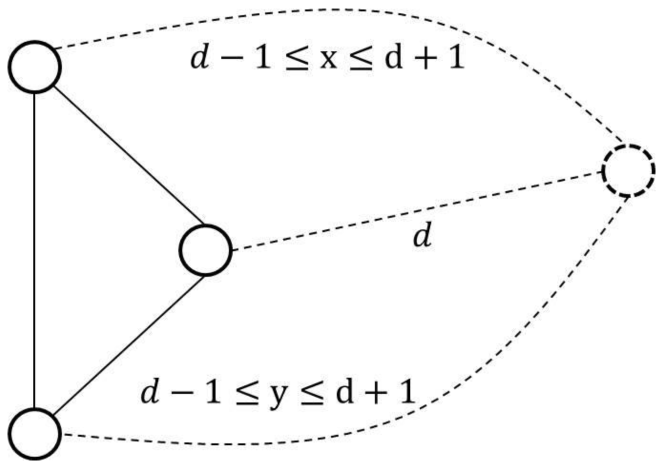


Figure.2.Distance of neighbor from a particular node

Proof: The proof of this theorem lays structural property of the network.
For any node u at distance r from v, if there existed a path shorter than $d - r$ distance, we would have chosen that path to compute minimum distance d.
Same way minimum path cannot be greater than $d - r$ there already exist a path of $d - r$ length through v.

C. LOCAL FLOODING

When a packet is received at any particular node we will use Local Flooding to get the address of the next node. First the router will check in its own table for entry if the entry is not found, it will generate a special packet with information about target node and a counter initiated with *r* - 1. Then it will flood this packet to all its neighbors. On receiving this packet the neighbor node will search for that entry in its table. If an entry is found it will send back an acknowledgement. So the flooding node will send the packet to that particular node. It will also decrement the counter and flood the packet to its entire neighborhood. This process will continue until counter becomes zero. In figure 3 router *v* floods special packet to its neighborhood of size 1. Among its neighbors node *u* has an entry for target node, therefor as shown in figure it will send back an acknowledgement.

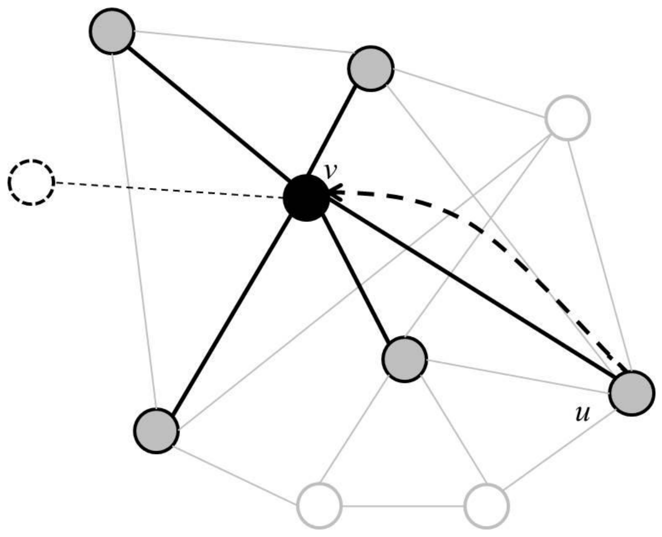


Figure.3.Flooding to neighborhood of size 1

C. TIME COMPLEXITY OF ROUTING

Here we will calculate the time complexity not on basis of actual time but the number of hopes travelled, e.g. if the maximum hops in the shortest path between any two nodes in the network is *n* we will say the time complexity of that algorithm is *O(n)*. In the current greedy algorithm made on small world network the worst case time complexity can be achieved $O(\log^2 n)$. In our algorithm the time complexity will be

$$T(n) = (r-1)\ logn + c\ logn$$

Where r = size of neighborhood
n = no. of nodes in network
c = the constant time for flooding

here *c* is constant so we can write time complexity as

$$T(n) = O(r\ logn)$$

The size of the neighborhood will be always smaller than *log n* as *log n* will be the order of the shortest path for the entire network.

III. ACQUAINTANCES PER NODE AND SHARING DATABASE

If the acquaintance per node is k and total number of nodes in network is n. Average path between nodes in network will be *log n/log k* [1]. If the database is shared between nodes of distance r, the number of entries stored per node e will be

$$e = \frac{n}{k^r}$$

This is significant as the number of entries can be easily reduced by increasing r. If k=20 and r=2 for a network of 1 million nodes each router only has to store only 2500 entries. At the same time the process of finding the correct entry among routers is also shared, therefor due to parallel searching the time reduces significantly.

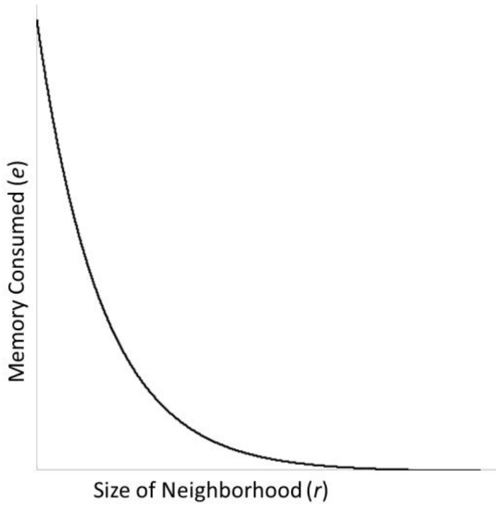


Figure.4. Relation between memory consumed and size of neighborhood

Figure 4 shows the number of entries per router with respect to size of the neighborhood. At neighborhood size zero entry per router is same as total no. of nodes in the network, so there is no sharing. We can see that number of entries stored in a single router decreases exponentially with increase in size of the neighborhood. At neighborhood size zero entry per router is same as total no. of nodes in the network, so there is no sharing.

Sharing database should be done in a way that the node in any neighborhood should contain entries of those nodes, which are nearest to it. This can be achieved by flooding of address by newly entered node; we will see it in detail in the next section.

IV. INITIALIZATION AND UPDATION OF DISTRIBUTED NETWORK TABLE

When applying approach with distributed database an important question arises of updating the database whenever a new node is added. Previously in this article we have assumed that network table is always updated, but we haven't defined any mechanism for doing it. Another question can be asked of initialization of network table but in fact we can use the same method used for updating for initialization of network table by updating it along with growth of the network.

To update each and every neighborhood about a newly added node we will use flooding. A newly added node will flood a special packet with its address to the entire network. However on receiving this packet each node will notify its neighborhood that a particular node has been added to the network table. This action will prevent other nodes from adding the same node again.

A. UPDATING THE NETWORK TABLE WITH SHORTEST PATH USING SYNC TREE

In the classical greedy approach we were flooding each update from a single node for a single node entry update in network table. So if we consider n node in a particular small world network and having a consideration of each node flooding the update to other n-1 nodes, then the total cost to implement such algorithm will have worst case time bound of $O(n^2)$. Also if we consider total n neighborhoods for n nodes and each neighborhood is subjected to update entry for each update for a single node, then also the cost will be worst. To solve such problem we will assume here small world network as a sync tree if n nodes arranged in $log\ n$ levels.

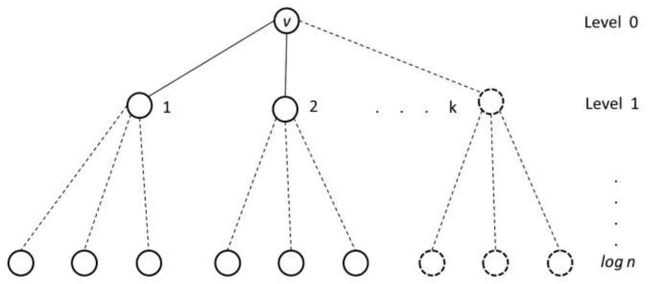


Figure.5. Sync tree of n nodes

As shown in figure 5, we are inserting node *v* with size of neighborhood *k* to update network entries of our small world network, we will consider node *v* as level zero node and all other nodes and their neighborhoods connected to the node *v* as level 1, level 2 and so on. With the network having *n* nodes *log n* levels will be generated. Separation of any node with *v* will be order of *log n*. Now if we flood according to the below algorithm then total time or cost will be order of *O(n log n)*

```
NODE_UPDATE(V)
1.  for each node $v$ ∈ network. V
2.          flood()
3.          for  level 1 to $log\ n$
4.                  update()
```

Figure.6. Node update algorithm

As per above algorithm the outer for loop shows that it will run for each *k* size neighborhood and the inner for loop is for the flooding levels and update till *log n* level. Hence the cost will be *O (n log n).*

V. CONCLUSION

From the above discussion we propose a better model with distributed network table for small world network. We have also proved it mathematically that with this approach we can route with cost of *O(r log n)* which is significantly better than cost of the popular greedy algorithm $(log\ n)^2$. We also showed that updating the network table is also cost efficient with cost of *O(n log n)*.

REFRENCES

[1] D Easley, and J Kleinberg. "Networks, Crowds and Markets: Reasoning about a Highly Connected World" Cambridge University Press 2010
[2] J. Kleinberg. The Small-World Phenomenon: An Algorithmic Perspective. In Proceedings of the 32nd ACM Symposium on Theory of Computing, 2000.
[3] S. Milgram. "The small world problem," Psychology Today 1, 61 (1967).
[4] O. Sandberg. Distributed routing in small-world networks. In Proc. of the Eighth Workshop on Algorithm Engineering and Experiments (ALENEX06), 2006.
[5] O. Sandberg and I. Clarke. "The Evolution of Small World Networks". The Department of Mathematical Sciences, Chalmers Technical University and Gothenburg University, 2007
[6] D.J. Watts and S. Strogatz. Collective dynamics of small world networks. Nature, 393:440, 1998.
[7] D.J. Watts. Small Worlds: The Dynamics of Networks between Order and Randomness. Princeton University Press, 1999