
WWJMRD 2017; 3(12): 83-86
www.wwjmrd.com
*International Journal*
*Peer Reviewed Journal*
*Refereed Journal*
*Indexed Journal*
*UGC Approved Journal*
*Impact Factor MJIF: 4.25*
e-ISSN: 2454-6615

**Shaik Naseera**
School of Computer Science and Engineering, VIT University, Vellore, India

**Gayathri P**
School of Computer Science and Engineering, VIT University, Vellore, India

**Santhi H**
School of Computer Science and Engineering, VIT University, Vellore, India

**Gopichand G**
School of Computer Science and Engineering, VIT University, Vellore, India

**Geraldine Bessie Amali D**
School of Computer Science and Engineering, VIT University, Vellore, India

**Shubham Kumar**
School of Computer Science and Engineering, VIT University, Vellore, India

**Correspondence:**
**Shubham Kumar**
School of Computer Science and Engineering, VIT University, Vellore, India

# A Study on Six Degrees of Separation Using Self-Adapting Algorithm

**Shaik Naseera, Gayathri P, Santhi H, Gopichand G, Geraldine Bessie Amali D, Shubham Kumar**

**Abstract**
Six Degrees of Separation is a theory which states that any two random people in the world can be associated with each other with no more than six intermediate links. In this paper we use this theory to create an application which is going to help the user in expanding his professional network and make use of opportunities that he otherwise would not be able to. This is achieved through maintaining a central search server which contains the details of all the users using our application. The server continuously finds links between various users based on their activities and updates the link table if that link has not been established before. Doing so will result in a much smaller wait time for the clients as in most of the cases the link will already be present in the link table. The search time for links of 3rd degree are easy to establish and thus will be done by default for all clients limited to their friend list. This move will make it much easier to return client requests as most of the time they are limited to their friends circle. We have tried to make use of self-adapting algorithms in this paper which will be responsible for calculating such statistics like average degrees of separation, establishing links on its own for most active clients, updating 3rd degree links for every client and updating the link tables. The system will make use of a main server side compute algorithm where the main handling logic will run on a parent process or thread. As soon as any client connects to the server through socket connection a new child thread will be created by the server in order to serve the new client. The client will only send its client id and the target name. This data will then be used by the server to compute and find the link between the client and the target. As soon as the server finds the link the client thread will send this data to the client and display it to the user. After this the client thread will be killed and the connection with the client will be terminated. The main server thread will be responsible for accepting client requests, creation of new threads, background passive searching, updating the link table, and modifying the statistics. The details about the clients will be stored in a central database which will be accessed only by the server side algorithms.

**Keywords:** Social Network, connectivity, sig degree, separation, search, link

## Introduction
Since in our application the size of the network and complexity will grow as the number of users increases our main objective is to minimize the time taken to establish a valid link between the user and the target. The time taken to establish a link can depend on various factors like the position of the target relative to the user in the search network, average number of friends in the network, complexity of the domain, diversity and various other dynamic factors.

Two directional search is capable of moving in both the direction in the search domain, which means we can search in both the direction in the search domain simultaneously. This is beneficial for us as it helps to reduce the depth that each search side has to visit.

The application system will follow client server architecture with all the computation done on the server side. The clients only need to send requests which include the target to be searched for.

**Link-Table Search**

The architecture followed for this model is based on client server architecture as shown in Fig. 1. As mentioned earlier because our search domain is huge we have to find a solution to minimize the search time to find valid links. To do this we have used a link table which will record all the links found by the application so that they can be directly used in the future whenever the same link or a subset of that link is searched for.

This helps us to make sure that search time for links can be reduced considerably. Apart from this to reduce the timing we have designed the algorithm in such a manner that each new user will by default be connected up till the 3rd degree automatically. This process of finding links will be limited to the user's friend list and will take place in the background. The users will be seen as clients who will have a small GUI that they can use to find and add contacts and also send requests to the server to find a link between them and a person. The server will accept this request and create a new child process to handle this request. The search will always start from the link table. If a link is found here the child process returns this link to the client and kills itself. If not the algorithm will query all the third degree links and continue to use the last link to search the next three links. This helps because half of the link is already preprocessed and helps to reduce the time considerably. If the link is not present in the user's friend network then the process is moved to the background and the search starts using the next valid network.

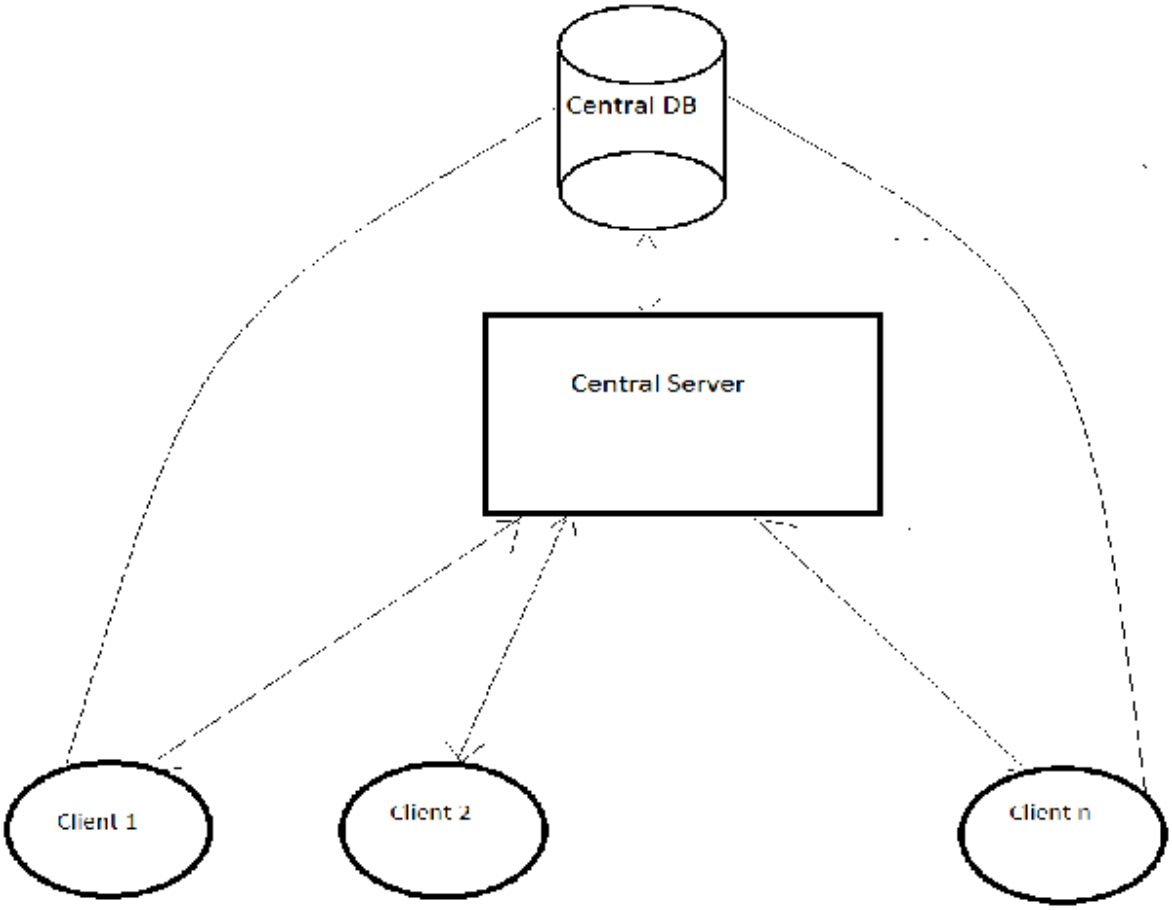


**Fig. 1:** Architecture of the Application

The system will make use of a main server side compute algorithm where the main handling logic will run on a parent process or thread. As soon as any client connects to the server through socket connection a new child thread will be created by the server in order to serve the new client. The client will only send its client id and the target name. This data will then be used by the server to compute and find the link between the client and the target.

As soon as the server finds the link the client thread will send this data to the client and display it to the user. After this the client thread will be killed and the connection with the client will be terminated. The main server thread will be responsible for accepting client requests, creation of new threads, background passive searching, updating the link table, and modifying the statistics.

The details about the clients will be stored in a central database which will be accessed by the server side algorithms. The ER diagram showing the relationship is given in Fig.2.

The database consists of three tables. A main table that stores all the user login information and their refined data like their hometown, state, email id, organization etc.

The second table is called the friends table which has all the information of the friendships formed by all the users. This table is widely used by the compute algorithm to find the links.

The last table is the link-table which stores all the previously found links and also the links up to the 3rd degree for every user within their friends list.

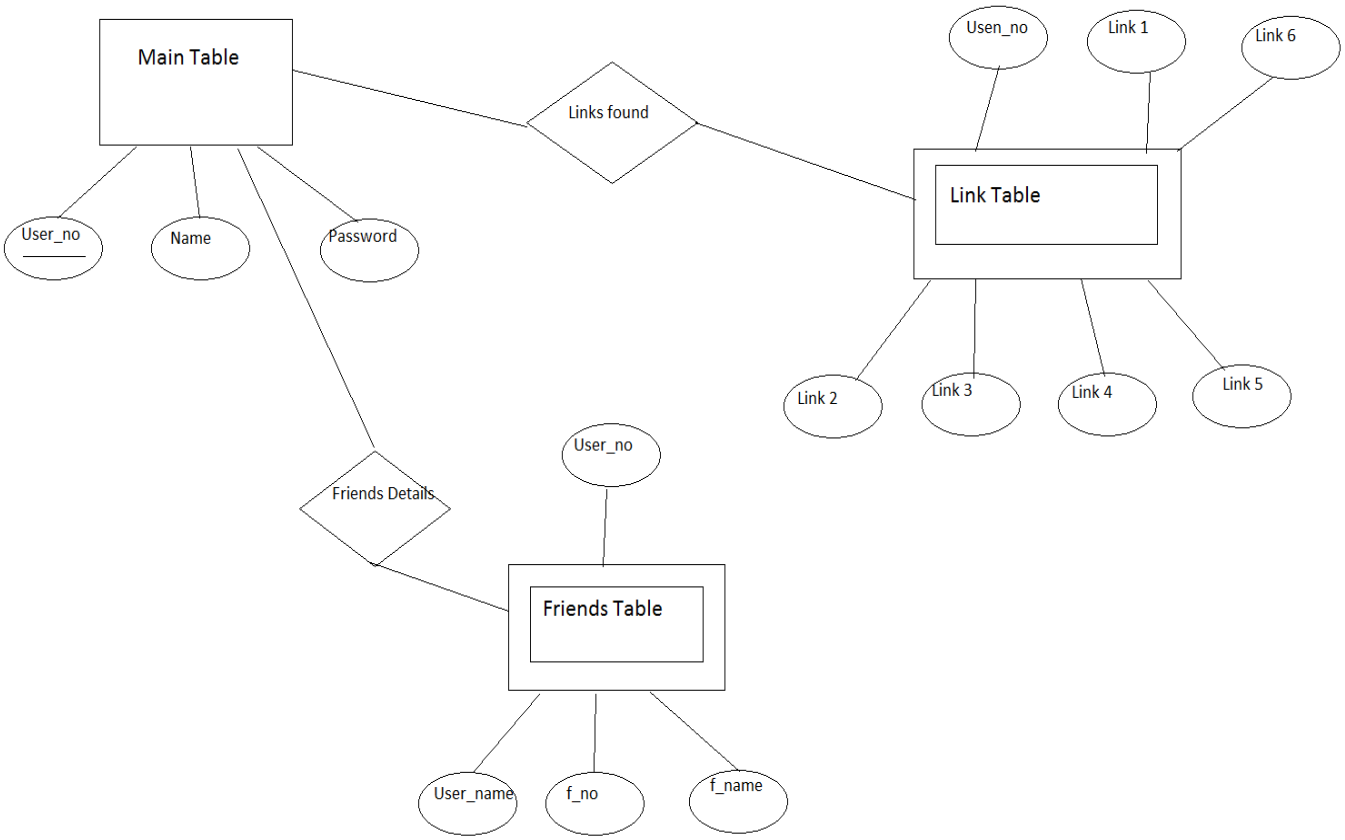


**Fig. 2:** ER Diagram of the Database Used

**Design and implementation**

In order to achieve minimized search time we will set up a record table called link table which will store all the established links that the application finds for future use. In addition to this we will have a server side compute which will keep on finding links continuously between various users using statistical data and storing those links for future use.

The client simply needs to send a target request using socket programming to the server and the server will return back a valid match between the client and the target.

The server starts by searching the link table for the requested link and if found simply returns the link to the client. If the link is not present in the link table then the server starts to search for the link using the server side compute logic. In order to make this search time small we have designed the server to automatically find all the links between every client and its friends list. This means that if the link is not present in the link table then the server will have to only start from the 4th degree onwards. This move will help to save time by making use of previous computations.

If the link to be found is outside the clients friend list then we will put this search thread in the background as searching through the whole search domain can take a lot of time. The client in this case would be sent a wait message and the link will be sent as soon as the server completes the search.

We have also made use of statistical values in order to make the algorithm self-adaptive. The initial offset for automatic searching of links inside the client's friend list will be set to 3. The server will automatically determine based on the client traffic the average value for the links that are being searched for. If this value changes by 1 over the previously set value of 3 the server will automatically start to search for the new next links and update them in the link table.

The link table is maintained inside the central database in the form of links that are successfully found in the previous searches. This helps to query the database with just the first link id which will be the id of the client and all the previously found links originating from that clients will be used to search for the present link required by the user.

**Experimental Results**

The application is expected to produce valid links up to a maximum of six degrees of separation in acceptable amounts of time. The wait time that a client gets before obtaining the resultant link has been minimized to a great extent by using link tables and statistical values. The client in the worst case will have to wait a longer duration as its search thread will be sent to run in the background. Fig. 3. Shows the time taken by various algorithms to find the links.

In most of the cases the client is expected to search for links in its own friend domain which will extend utmost to the 3rd degree. As these links have already been computed the clients for the most cases will obtain their results instantaneously.

The server is also expected to control the client threads properly and monitor the growth of the network and also update the statistical data which is used by the search algorithms.

Apart from this the background searching for the most used clients should also be properly executed by the server.

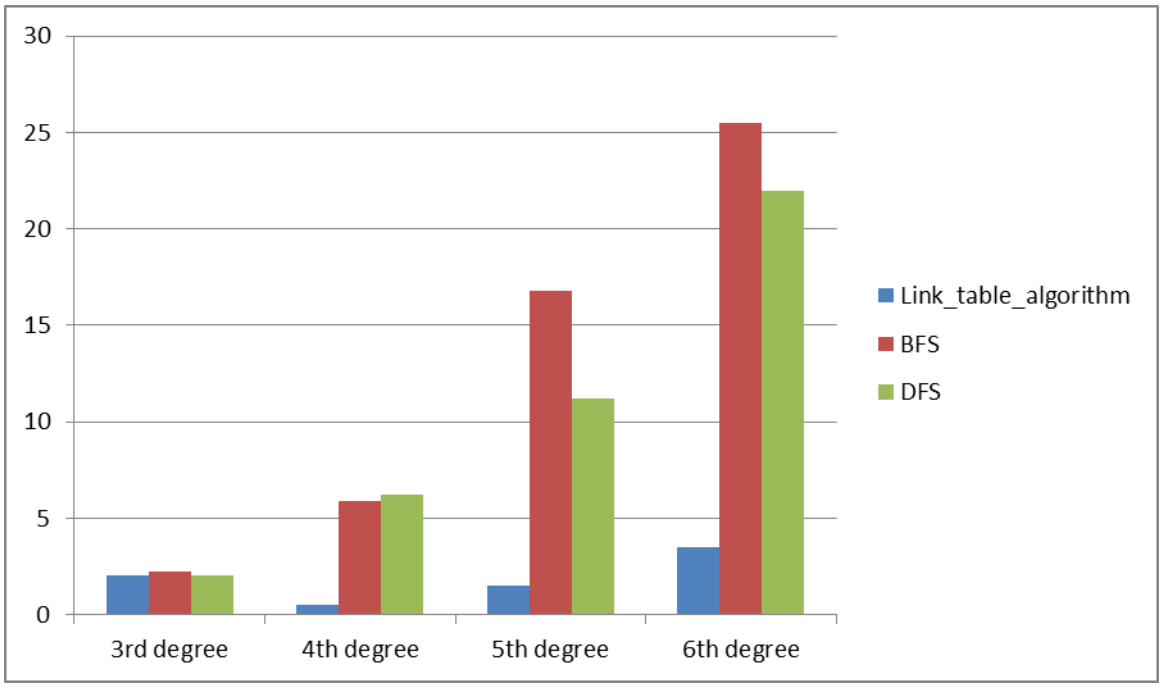


**Fig. 3**: Time Taken to Find a Test Links by Various Algorithms

## Conclusion

The idea that anyone in the world can be connected in six or less steps was enough to motivate us to look for its application in real life. Upon extensive research we came up with this idea to help people connect better with others using their common friends. Our application does a good job to achieve this but as the proper implementation of six degrees of separation requires a large search domain that is well connected our application would fail to utilize the true meaning of "six degrees of separation" for every user without a well-populated database.

Also the use of various optimizing techniques such as hash tables, look-back tables and heuristic priority enables our algorithm to get search results quite fast but this is only tested on a small database, which we feel may need a lot more optimization in future while dealing with a much larger database.

In general due to the use of link tables and making use of already found links our algorithm has proven to be better at finding links than other common search algorithms like DFS and BFS. Since our algorithm already has the first 3 links stored in the database it has to search only the remaining 3 links when searching for the entire 6 degrees of separation. This reduces the search time drastically as the algorithm actually has to look only for the remaining 3 links. Hence reducing the compute load by half. The worst case scenario again comes out to be O (N3) instead of O (N6).

Future scope of this model is huge. As the user base will increase so will the computational requirements. This will require high end server side computing power and storage space. To reduce the financial cost that will be required. We can make use of SETI's compute at home program. This is a free ware background computing application from SETI which is sponsored by NASA. It lets a normal user install a screen saver application which starts to compute the application logic once the user is away. The user will need to download a dataset which will be used by the algorithm to find the links. Once the dataset is completed the user application will upload the processed dataset to the server. Hence we can use this and save on computing costs.

## References

1. E. Edwin Lawrence and Dr. R. Latha (2015), "Analysis of Six degrees of separation in Facebook using Ant colony optimization", IEEE proceedings of International Conference on Circuit, Power and Computing Technologies (ICCPCT), 19-20 March 2015, Nagercoil, pp. 1-5.
2. Ke Xiao-hua (2008), "A Social Networking Services System based on the "Six Degrees of Separation" Theory and Damping Factors", IEEE proceedings of Second International Conference on Future Networks (ICFN-2010), 22-24 January 2010, Sanya, Hainan, pp. 438-441.
3. Reza Bakhshandeh, Mehdi Samadi, Zohreh Azimifar and Jonathan Schaeffer(2011), "Degrees of Separation in Social Networks", Proceedings of The Fourth International Symposium on Combinatorial Search (SOCS), 15-16 July 2011, Barcelona, Spain, pp. 18-23.
4. Lei Zhang and Wanqing Tu (2009), "Six Degrees of Separation in Online Society", Proceedings of the WebSci'09, Society On-Line, 18-20 March 2009, Athens, Greece, pp. 1-5.
5. Hakan Kardes, Abdullah Sevincer, Mehmet Hadi Gunes, and Murat Yuksel (2012), "Six Degrees of Separation among US Researchers", IEEE/ACM Proceedings of International Conference on Advances in Social Networks Analysis and Mining, 26-29 August 2012, Kadir Has University, Istanbul, Turkey, pp. 654-659.
6. Facebook, https://www.facebook.com.
7. Six Degrees.com, http://www.sixdegree.com.
8. LinkedIn, https://www.LinkedIn.com.