**CHORD SUMMARY**

Chord is a scalable distributed lookup protocol which solves the major problem which most of the peer to peer applications face. It locates the node where the data is stored by assigning a key to each data item. Chord can also dynamically adapt itself depending on the entry and exit of the nodes and they are also not system dependent. The main operation is a basic chord protocol is to assign a key to each data as soon as they enter the node and using the key they can access the data easily and conveniently. This process of assigning key is done using consistent hashing since each node will receive the same number of keys as other nodes and each nodes have details about some other nodes present in the system, i.e. (O log N) other nodes. Chord is better than other peer to peer systems since it solves the following problems: Load balance, decentralization, availability, scalability and flexible naming. The chord software links with the client and server applications in the form of library and they interact with chord library in one of the following two ways: It can either produce the key algorithm which specifies the IP address of the node or it can either intimate the application about the change in key and the node responsible for it. Some of the applications which uses these mechanism are time-shared storage, large-scale combinatorial search, etc. Also node joins is one of the most important mechanism in chord as each node has to know what node is entering and exiting the system and the new nodes has to know about the other nodes. This can be done using the following process: Initializing fingers and predecessors where the newly entered nodes appoint some other nodes to know about its predecessor, Updating fingers of existing nodes and transferring keys from other nodes. Also there are some modifications needed to the basic chord system in-order to make it more efficient in dealing with the nodes entering and leaving the system. Some of the modifications which are done to make the chord much more efficient is by improving the stabilization, which helps in keeping the nodes and successor protocol up to date. In some cases, there can be some problem with joining of nodes which affects the chord ring and in these cases first an initial lookup is done and details get added to the finger table and then the stabilization process continues. Also if a particular node fails it should not affect the entire system and the process should run through. This can be done only when the data’s in that particular node finds its successor node in the finger table. Since already all chords maintain a successor list using a finger table failure and replication of nodes can be performed. So in future this process can be improved by healing the partitioned rings. This can be done by making each node check for other nearby nodes and when some node is missing partition occurs. Another approach can be done by making the nodes to maintain a long term memory so that all the details can be stored in that memory about all other nodes. Also generally hacking the data in the node is one major problem with chord where a hacker can send a malicious node into the chord ring and get the data. This can be checked or prevented if nodes use IDs derivatives from SHA-1 hash. Also in future the lookup latency can also be improved by using server selection where each finger node points to first entry in the node. Thus, chord is the most preferred product in peer to peer system environment and is mainly used for large systems where there is large amount of data.