# COP5615 Project 3:Implementation of Chord Algorithm using Actor Model in Erlang

Documentation

Srinivas Koushik Kondubhatla (UFID: 69238911)
Dharani Kanchanapalli (UFID : 75351996)

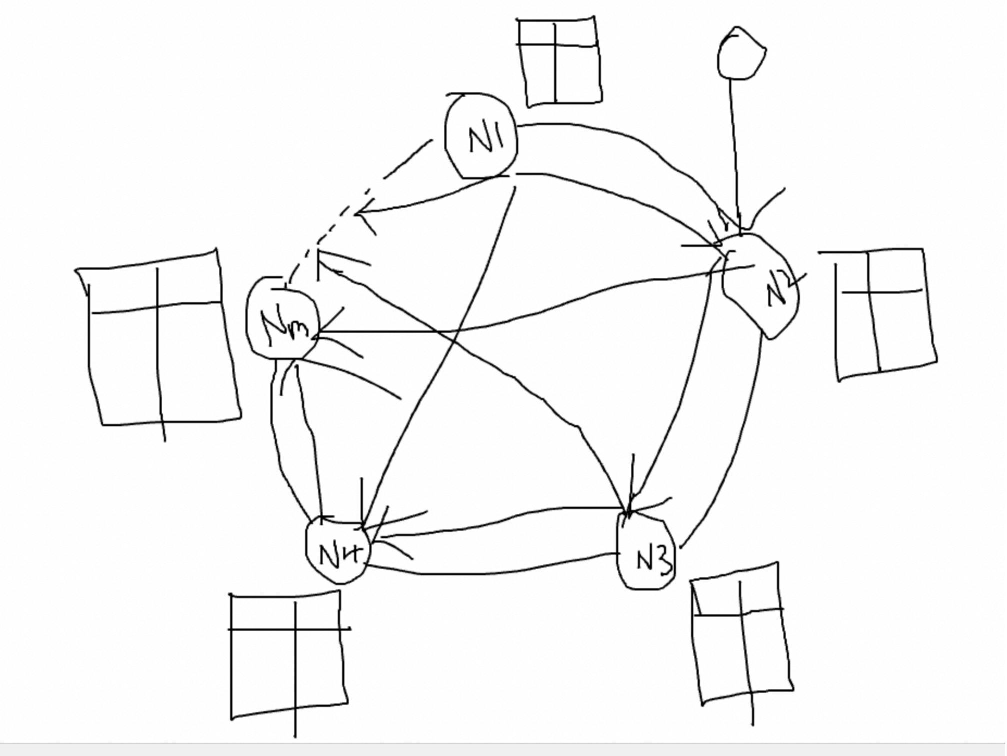
## Problem Statment

Chord is P2P protocol distributed hash-table storing key-value pairs  key is assigned to all the nodes(computers) in the network.

## Implementation

The implementation of Chord Protocol simulator using actor model in Erlang by passing a message to the node or successor of that node which is In network using finger tables for message transfer.

### Architecture



## Nodes Functionalities

### Network Creation

- Compute the hash of the address of the node using SHA1 algorithm.
- The successors of all hashes between keys of nodes Ni and Ni+1 will be Ni+1

### Finger Table Creation

- The key of the computer is the hash let's say ith node hash is $N_i$
- The nodes ith node is connected to successors of $(N_i + 2^0, N_i + 2^1 + N_i + 2^2 + \ldots + N_i + 2^{k)}$

### Lookup

- A node is randomly chosen to lookup for key
- S = ***sucessor****(key)*
- Node n will check it's finger table and send the message to the key which is greater than or equal to S.
- Continue until n = S.

### Node Addition

- node N is inserted between Ni and Ni+1
- The successor of all hashes between Ni and N will now be changed to N from Ni+1 and finger tables are updated(stabilized) accordingly
- The successors of all hashes between keys of nodes Ni and Ni+1 will be Ni+1

## Actor Communication

- Let S = predecessor(key)
- Node computes the the largest key K which is less than or equal to S
- If K = S, terminate
- Else send K a message to lookup for key

## Run Instructions

-> Start erlang compiler my entering erl

-> Run the following commands

```
    -> c(chord).
    -> chord:main().
```

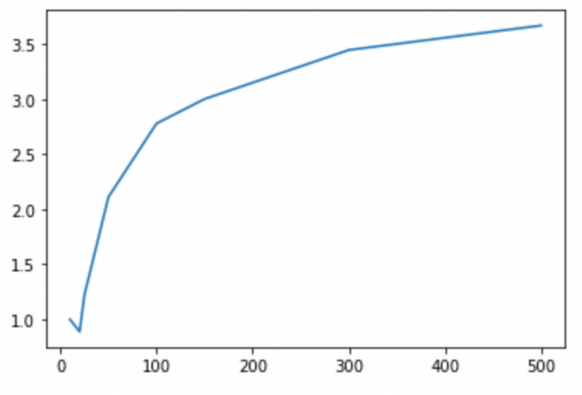
# Results

The implementation of chord Implementation is checked w.r.t to maximum number of nodes in a network(m bits) and the size of network the total number of nodes in the network. The code has run 10 times for each of these configuration and competed the average number of hops from node 1 to desired node(**successor**(key)). The 10 keys are generated with 10 random addresses and the average number of hops is plotted against size of network.

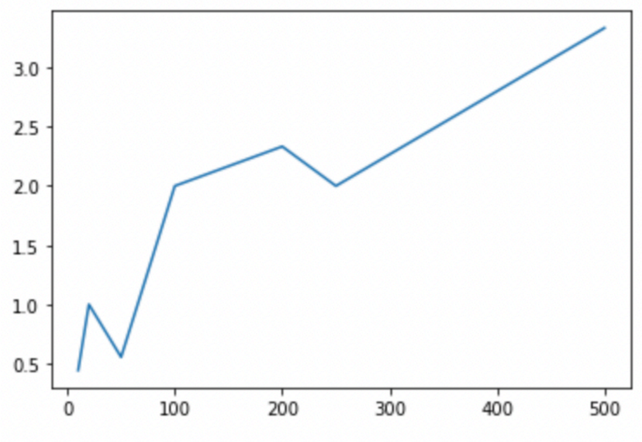
The plotting has been done for various m i.e [140,10,20,30]. The Number of hoops are plotted

## Graphs(size of network vs Average Number of hoops)

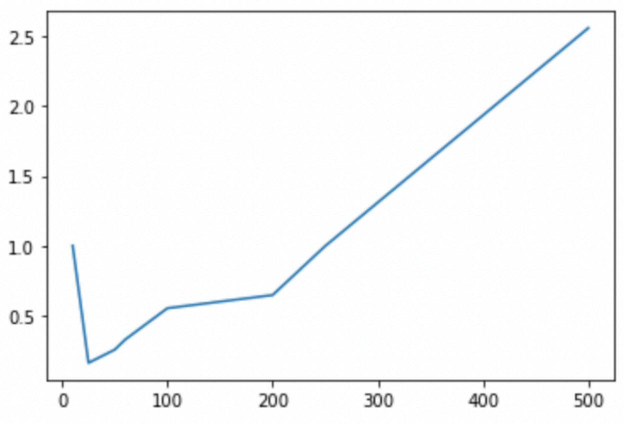
m=10



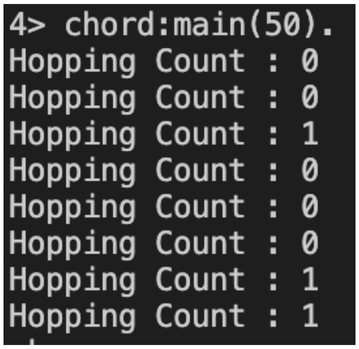
m=20



m=140



## Output

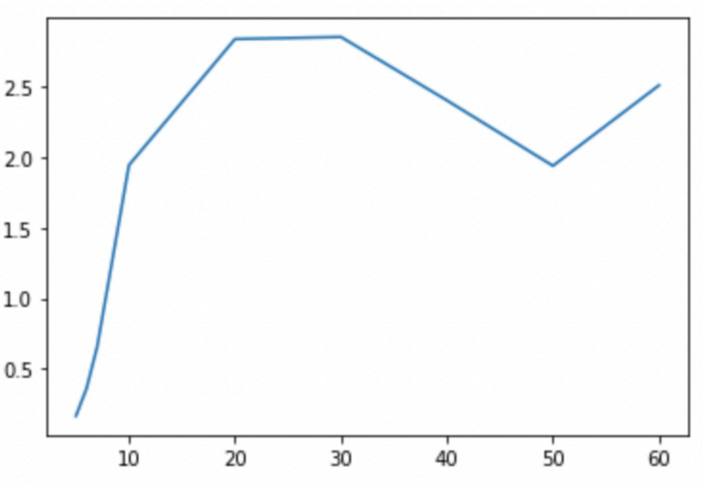


```
4> chord:main(50).
Hopping Count : 0
Hopping Count : 0
Hopping Count : 1
Hopping Count : 0
Hopping Count : 0
Hopping Count : 0
Hopping Count : 1
Hopping Count : 1
```

## Observations

As the size of the network increases or m is increased, the time taken for the lookup is increasing. As the network size increases, the number of lookups have increased. The m and the Total number of nodes are directly proportional to each other. The time complexity for the lookup will be at most O(log$^2$n).

## Comparison (m vs lookups)



## Largest Network we could work with for all the topologies.

For all the topologies the process is killed after network size **10,000**. The time take for network size 1000 is around 6 mins.

## Conclusion

The working of implementation of chord protocol using actor model and distributed Erlang has been compared with various network sizes and maximum number of nodes. The time complexity for the lookup will be at most O(log$^2$n). The functionalities of chord protocol are

- Add Node
- Delete Node
- Lookup
- Stablize

The time taken for lookups are directly proportional to size of network and m. The maximum number of hops for network size 1000 are 7. The time taken is approximately 6 minutes.