DOSP PROJECT 3

Manish Kumar Gangula, Jaswanth Reddy Kankanala

Description:

Chord is one of the protocols used to communicate with the other nodes in the distributed network, where a distributed hash table is maintained with keys indicating each node. An optimized algorithm must be maintained for organizing the keys to nodes. Chord protocol specifies a way that the keys to be assigned and interaction between one node with another node using the keys assigned.

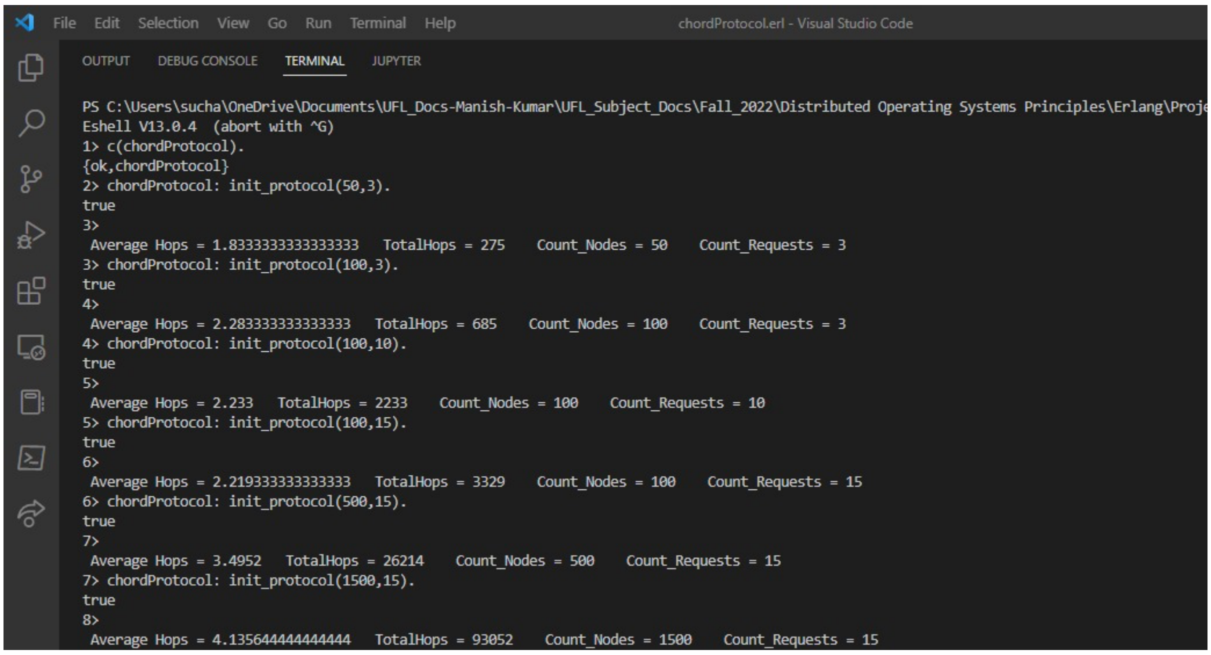
This implements the arrangement of the nodes and hashes in a circular format with at most of 2^m nodes starting from 0 to 2^{m-1}. Each of the node is assigned with a key or it is left null. Core objective is to identify the successor for the given key. A linear search would be necessary with the basic approach, and by maintaining finger table, the search will be optimized, and speeds are improved and order changes to Log(N). This includes various applications like load balancing in a network, P2P file transfers, etc.,

Steps to Run:

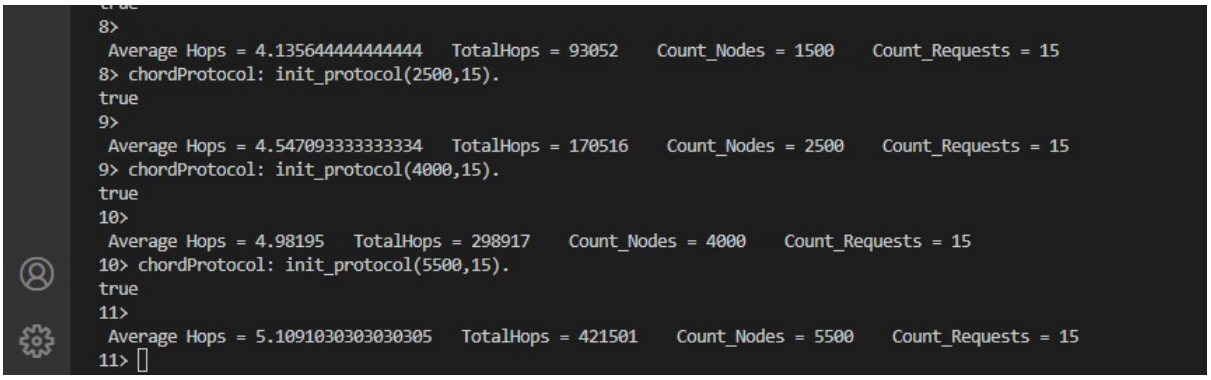
1. The source code is basically zipped, needs to unzip the folder, and navigate to the current folder to execute the code.
2. Enter the Erlang shell by typing erl.
3. Compile the code using c(chordProtocol). Command.
4. Upon successful compilation, we receive {ok, chordProtocol} on the console.
5. Now run the main method with the input as number of nodes and number of requests each node must make in the process of communication. Sample code run includes, chordProtocol:init_protocol(100, 10).
6. Perform the necessary triggers to the source code with varying inputs.

Results:

The output of the project gives us the idea about the chord protocol with the implementation of network join and routing across the nodes in the network with the use of keys assigned to each of the nodes. Output of the project gives the average number of node connections that have to be traversed for delivering the message for a node across the network.


File Edit Selection View Go Run Terminal Help
chordProtocol.erl - Visual Studio Code
OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

```
PS C:\Users\sucha\OneDrive\Documents\UFL_Docs-Manish-Kumar\UFL_Subject_Docs\Fall_2022\Distributed Operating Systems Principles\Erlang\Proje
Eshell V13.0.4  (abort with ^G)
1> c(chordProtocol).
{ok,chordProtocol}
2> chordProtocol: init_protocol(50,3).
true
3>
 Average Hops = 1.8333333333333333    TotalHops = 275    Count_Nodes = 50    Count_Requests = 3
3> chordProtocol: init_protocol(100,3).
true
4>
 Average Hops = 2.283333333333333    TotalHops = 685    Count_Nodes = 100    Count_Requests = 3
4> chordProtocol: init_protocol(100,10).
true
5>
 Average Hops = 2.233    TotalHops = 2233    Count_Nodes = 100    Count_Requests = 10
5> chordProtocol: init_protocol(100,15).
true
6>
 Average Hops = 2.219333333333333    TotalHops = 3329    Count_Nodes = 100    Count_Requests = 15
6> chordProtocol: init_protocol(500,15).
true
7>
 Average Hops = 3.4952    TotalHops = 26214    Count_Nodes = 500    Count_Requests = 15
7> chordProtocol: init_protocol(1500,15).
true
8>
 Average Hops = 4.135644444444444    TotalHops = 93052    Count_Nodes = 1500    Count_Requests = 15
```



```
8>
 Average Hops = 4.135644444444444    TotalHops = 93052    Count_Nodes = 1500    Count_Requests = 15
8> chordProtocol: init_protocol(2500,15).
true
9>
 Average Hops = 4.547093333333334    TotalHops = 170516    Count_Nodes = 2500    Count_Requests = 15
9> chordProtocol: init_protocol(4000,15).
true
10>
 Average Hops = 4.98195    TotalHops = 298917    Count_Nodes = 4000    Count_Requests = 15
10> chordProtocol: init_protocol(5500,15).
true
11>
 Average Hops = 5.1091030303030305    TotalHops = 421501    Count_Nodes = 5500    Count_Requests = 15
11> 
```