COP5615- Distributed Operating System Principles

Fall 2023
Programming Assignment #2

Team 27
Group members:

Name	UFID	Email
Rishika Reddy Alugubelli	**2888-0035**	**r.alugubelli@ufl.edu**
Aashritha Reddy Donapati	**7242-2083**	**a.donapati@ufl.edu**
Vivek Reddy Gangula	**5208-7488**	**gangula.v@ufl.edu**
Dheekshita Neella	**6325-8153**	**d.neella@ufl.edu**

Table of Contents

1. How to run the program?

- Editor Used : Visual Studio Code
  - Programming Language: F#
  - Supported OS: Windows, MacOS
- Install Ionide for F# Plugin for Visual Studio Code
- Install .NET SDK from dotnet.microsoft.com/download
- Download the Team27.zip file and extract it.
- Open the program.fs file in VSCode.
- Open a terminal and go to the path of the folder.
- Install the AKKA package using the following command:
  `dotnet add package Akka.FSharp`
- For compilation, run the `'dotnet build'` command in the terminal. The terminal displays a 'Build succeeded' message when the compilation is successful.
- For execution, run the following command in the terminal:
  `dotnet run <number of nodes> <number of requests>`

  Note : Both the Number of Nodes and Number of requests in the input should be integers.

2. What is working?

- This program uses the Actor model of the AKKA module to simulate the Chord protocol and peer-to-peer system.
- The simulator generates a chord by sequentially adding nodes to the zeroth node.
- The zeroth node does a lookup for new node IDs and assigns its successor to the resultant node of the lookup.
- When a node is added, the stabilize command is invoked to alter the successors and predecessors of the adjacent nodes.
- Once the chord has been stabilized, the finger tables are updated by activating the AddInFingerTable method on a regular basis.
- The chord creation is complete when all of the nodes are inserted.
- Then, at random, each node begins to issue lookup queries. The lookup output is provided to the simulator after it is located.
- The simulator terminates the program after each peer has sent the specified number of lookup outputs.

- The Chord Protocol usually returns the average number of hops in such a way that the average number of hops increases logarithmically with the increase in nodes. The main aim of this project is to see the behavior of the value of Average no. of hops with the increase in nodes.
- We have assigned the nodes and keys random values between 0 and 2^m - 1, where we have considered m to be 20.

3. Execution Results

Input: dotnet run 10 10

Output:

```
PS C:\Users\91938\Downloads\PA2> dotnet run 10 10
[INFO] Creating node 959569
[INFO] Creating node 101873
[INFO] Creating node 267146
[INFO] Creating node 554759
[INFO] Creating node 843547
[INFO] Creating node 310860
[INFO] Creating node 885082
[INFO] Creating node 944629
[INFO] Creating node 116233
[INFO] Creating node 129095
Waiting for 30 sec to get system stabilized
NodeID: 959569 converged with hopCount: 18, numRequest: 10
NodeID: 101873 converged with hopCount: 12, numRequest: 10
NodeID: 267146 converged with hopCount: 14, numRequest: 10
NodeID: 554759 converged with hopCount: 11, numRequest: 10
NodeID: 843547 converged with hopCount: 30, numRequest: 10
NodeID: 310860 converged with hopCount: 8, numRequest: 10
NodeID: 885082 converged with hopCount: 25, numRequest: 10
NodeID: 944629 converged with hopCount: 29, numRequest: 10
NodeID: 116233 converged with hopCount: 15, numRequest: 10
NodeID: 129095 converged with hopCount: 16, numRequest: 10
Total number of hops: 178
Average number of hops: 1.780000
PS C:\Users\91938\Downloads\PA2>
```

Input: dotnet run 100 10

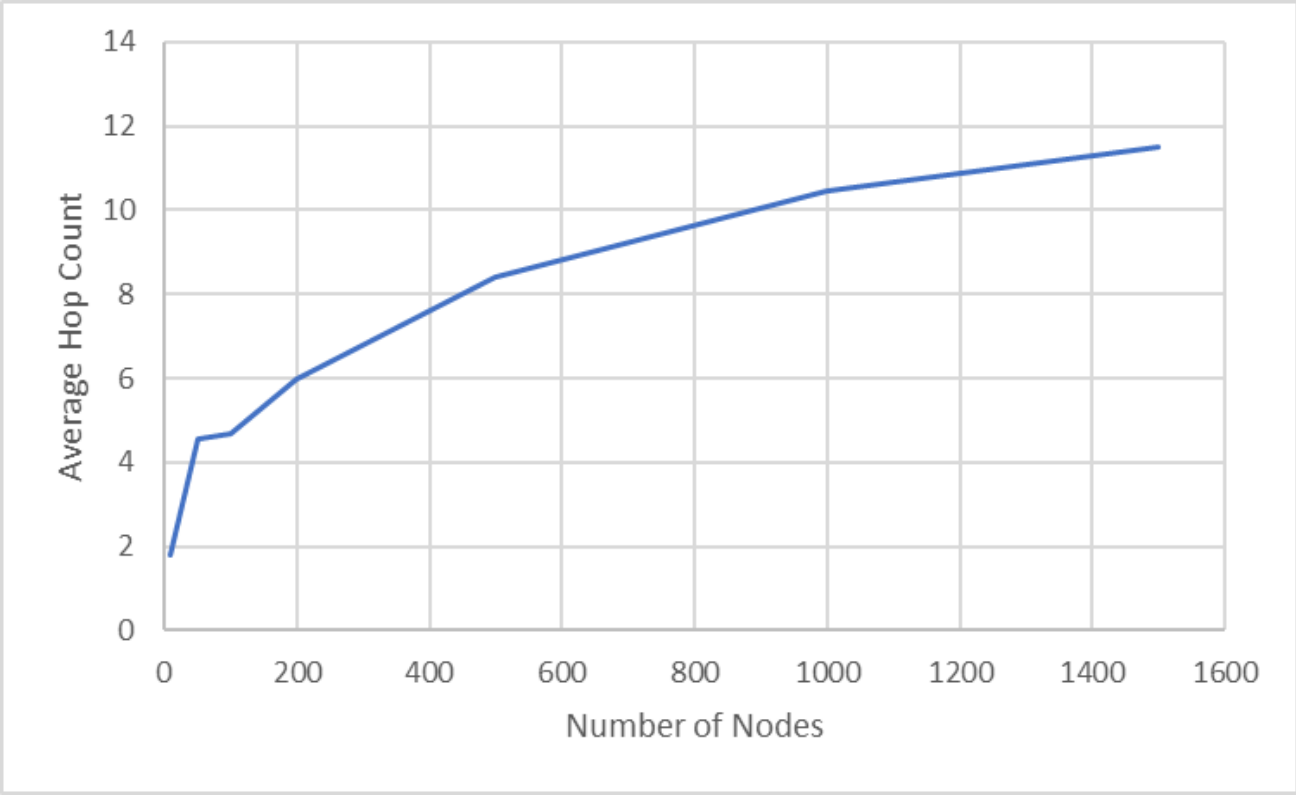
Output:

```
NodeID: 154207 converged with hopCount: 54, numRequest: 10
NodeID: 955374 converged with hopCount: 46, numRequest: 10
NodeID: 470790 converged with hopCount: 47, numRequest: 10
NodeID: 345999 converged with hopCount: 44, numRequest: 10
NodeID: 14824 converged with hopCount: 44, numRequest: 10
NodeID: 668191 converged with hopCount: 62, numRequest: 10
NodeID: 141642 converged with hopCount: 52, numRequest: 10
NodeID: 517582 converged with hopCount: 47, numRequest: 10
NodeID: 313353 converged with hopCount: 37, numRequest: 10
NodeID: 573201 converged with hopCount: 61, numRequest: 10
NodeID: 262773 converged with hopCount: 30, numRequest: 10
NodeID: 377336 converged with hopCount: 39, numRequest: 10
NodeID: 458657 converged with hopCount: 44, numRequest: 10
NodeID: 305009 converged with hopCount: 38, numRequest: 10
NodeID: 820391 converged with hopCount: 50, numRequest: 10
NodeID: 528530 converged with hopCount: 36, numRequest: 10
NodeID: 337689 converged with hopCount: 50, numRequest: 10
NodeID: 855934 converged with hopCount: 73, numRequest: 10
NodeID: 532920 converged with hopCount: 71, numRequest: 10
NodeID: 908545 converged with hopCount: 52, numRequest: 10
Total number of hops: 4699
Average number of hops: 4.699000
PS C:\Users\91938\Downloads\PA2>
```

4. Table

Number of Nodes	Number of Requests	Average Hop Count
10	10	1.78
50	10	4.542
100	10	4.699
200	10	5.9895
500	10	8.388
1000	10	10.4439
1500	10	11.521933

5. Graph (Number of Nodes VS Average Hop Count)



6. Assumptions

The following are the assumptions made in the protocol

- The network's nodes are distributed throughout the software environment.
- The value of m is set to 20 when creating m-bit identifiers for nodes.
- The distributed network is a dynamic one, with nodes entering and leaving as needed.
- The protocol enables the graceful termination of all nodes and the system environment.

7. Largest Network

Although we are confident that our program can handle more nodes and requests, the largest network we have evaluated is of 1500 nodes with 10 requests.

Average Hop Count for this network = 11.521933