University of Florida

Department of Computer and
Information Science and Engineering

COP5615 - Distributed Operating System Principles
Fall 2023
Programming Assignment 2

Name	UFID	Email
Sheel Taskar	27534465	sheel.taskar@ufl.edu
Shivani Patil	53549503	patil.s@ufl.edu
Abdul Samadh Azath	22137268	azathabdulsamadh@ufl.edu

How to run the Program?

- Unzip the file, you will see Program.fs which contains the main code.
- To run the program, use command given below

```
dotnet run <numNodes> <numRequests>
```

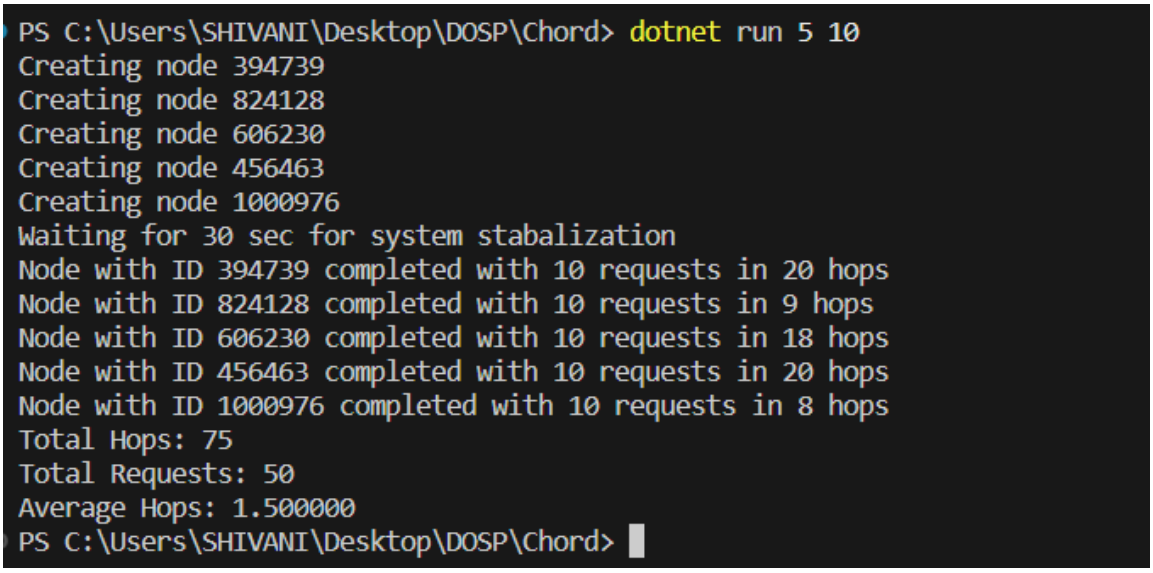
- <numNodes> denotes the total number of peers to be generated in the peer-to-peer system.
- <numRequests> determines how many requests each peer must make.

What is working?

- This encompasses two primary methods of key lookup: a straightforward linear trend approach and a scalable logarithmic trend approach.
- The initial node is established as a static entity, while the subsequent nodes are dynamically generated and incorporated into the chord ring using consistent hashing principles.
- Determining both the node's successor and the key's location relies on a combination of finger tables and the scalable lookup algorithm described in the research paper.
- Following the service of a specified number of requests by all nodes, the average hop count is computed to assess the network's efficiency.

Screenshots:

1. **Simple Key Lookup:**



Some Outcome for higher input:

```
PS C:\Users\SHIVANI\Desktop\DOSP\demo> dotnet run 100 100
Waiting for 20 sec for system stabalization
Total Hops: 490972
Total Requests: 10000
Average Hops: 49.097200
PS C:\Users\SHIVANI\Desktop\DOSP\demo> dotnet run 200 200
Waiting for 20 sec for system stabalization
Total Hops: 3983812
Total Requests: 40000
Average Hops: 99.595300
PS C:\Users\SHIVANI\Desktop\DOSP\demo> dotnet run 300 10
PS C:\Users\SHIVANI\Desktop\DOSP\demo> dotnet run 400 15
Waiting for 20 sec for system stabalization
Total Hops: 1200152
Total Requests: 6000
Average Hops: 200.025333
PS C:\Users\SHIVANI\Desktop\DOSP\demo>
```

```
PS C:\Users\SHIVANI\Desktop\DOSP\demo> dotnet run 150 150
Waiting for 20 sec for system stabalization
Total Hops: 1676011
Total Requests: 22500
Average Hops: 74.489378
PS C:\Users\SHIVANI\Desktop\DOSP\demo> dotnet run 300 300
PS C:\Users\SHIVANI\Desktop\DOSP\demo> dotnet run 300 10
Waiting for 20 sec for system stabalization
Total Hops: 449062
Total Requests: 3000
Average Hops: 149.687333
PS C:\Users\SHIVANI\Desktop\DOSP\demo> dotnet run 500 20
Waiting for 20 sec for system stabalization
Total Hops: 2480636
Total Requests: 10000
Average Hops: 248.063600
PS C:\Users\SHIVANI\Desktop\DOSP\demo>
```

2.Scalable Key Lookup:

```
PS C:\Users\SHIVANI\Desktop\DOSP\Chord> dotnet run 5 10
Creating node 241617
Creating node 904669
Creating node 51379
Creating node 576312
Creating node 968421
Waiting for 30 sec for system stabalization
Node with ID 241617 completed with 10 requests in 7 hops
Node with ID 904669 completed with 10 requests in 13 hops
Node with ID 51379 completed with 10 requests in 10 hops
Node with ID 576312 completed with 10 requests in 10 hops
Node with ID 968421 completed with 10 requests in 17 hops
Total Hops: 57
Total Requests: 50
Average Hops: 1.140000
```

Some Outcome for larger input:

```
PS C:\Users\SHIVANI\Desktop\DOSP\demo> dotnet run 5 5
Waiting for 20 sec for system stabalization
Total Hops: 35
Total Requests: 25
Average Hops: 1.400000
PS C:\Users\SHIVANI\Desktop\DOSP\demo> dotnet run 10 5
Waiting for 20 sec for system stabalization
Total Hops: 108
Total Requests: 50
Average Hops: 2.160000
PS C:\Users\SHIVANI\Desktop\DOSP\demo> dotnet run 20 10
Waiting for 20 sec for system stabalization
Total Hops: 617
Total Requests: 200
Average Hops: 3.085000
PS C:\Users\SHIVANI\Desktop\DOSP\demo> dotnet run 30 5
Waiting for 20 sec for system stabalization
Total Hops: 534
Total Requests: 150
Average Hops: 3.560000
```

```
PS C:\Users\SHIVANI\Desktop\DOSP\demo> dotnet run 2000 15
Waiting for 20 sec for system stabalization
Total Hops: 367235
Total Requests: 30000
Average Hops: 12.241167
PS C:\Users\SHIVANI\Desktop\DOSP\demo>
```

```
PS C:\Users\SHIVANI\Desktop\DOSP\demo> dotnet run 3000 10
Waiting for 20 sec for system stabalization
Total Hops: 365592
Total Requests: 30000
Average Hops: 12.186400
PS C:\Users\SHIVANI\Desktop\DOSP\demo>
```

Table of average hop count results

1. **Simple Key Lookup:**

No of Nodes	Average No of Hops
5	1.60
10	4.65
15	7.55
20	9.24
30	14.54
50	24.89
75	37.07
100	49.09
150	74.48
200	99.61
300	149.66
400	200.02
500	248.06

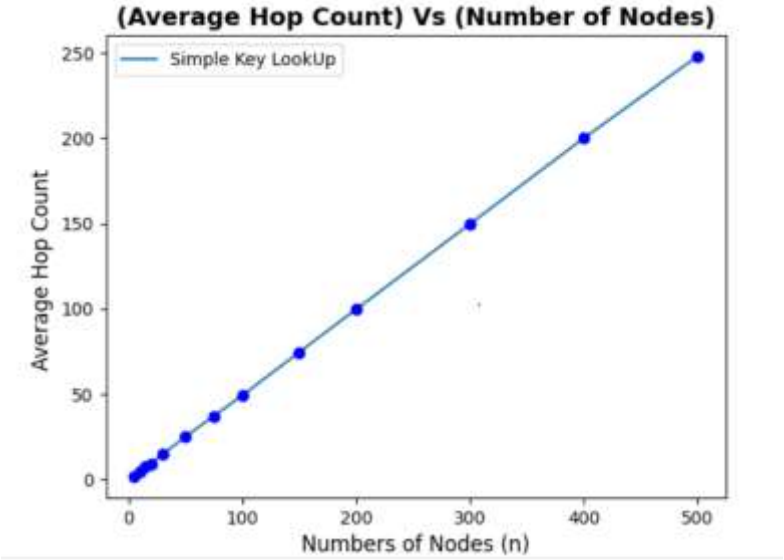
2. **Scalable Key Lookup:**

No of Nodes	Average No of Hops
5	1.40
10	2.16
20	3.08
30	3.21
50	3.85
100	5.27
150	5.46
200	6.96
300	7.59
500	8.56
750	9.37

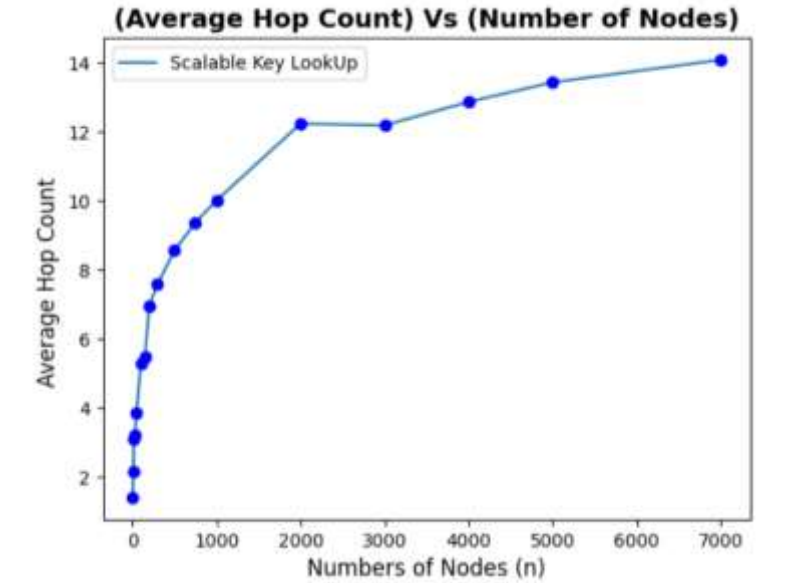
1000	10.01
2000	12.24
3000	12.19
4000	12.87
5000	13.44
7000	14.09

Graph of number of nodes vs average hop count

1. **Simple Key Lookup**



2. **Scalable Key Lookup:**



Assumptions about the protocol

- When dealing with the initial node, we set up the complete finger table using its own value.
- For all subsequent nodes, we initialize the entire finger table with the successor identified during the Join () process.
- In the context of the FindSuccessor() method, the paper does not address the situation in which both 'n' and the successor are identical. This scenario occurs when a second node seeks to join the chord ring. In such instances, we simply return the value of the successor (or 'n') itself. Consequently, the second node designates the first node as its successor, and this prompts a subsequent update of the finger tables.
- Furthermore, when the key identifier matches the node identifier, we transmit the node's own identifier as its designated successor.
- Upon a node's entry into the system, we establish schedulers for both the Stabilize () and FixFinger() procedures. These schedulers operate at identical time once the node is part of the network.

What is the largest network you managed to deal with?

Number of Nodes: 7000, Number of Requests: 11