COP5615- Distributed Operating System Principles
Fall 2023
Programming Assignment #2

Chord: P2P System and Simulation

Deadline: Nov 2, 2023

LATE submission will be accepted for grading!

How to submit: Please complete the group submission via CANVAS system.

Introduction

The goal of this project is to implement the Chord protocol using F#.

The specification of the Chord protocol can be found in the paper *"Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications"* by Stoica, Morris, et.al. https://pdos.csail.mit.edu/papers/ton:chord/paper-ton.pdf. The paper above, in section 2.3 contains a specification of the Chord API and of the API to be implemented by the application.

You can also refer to the Wikipedia page: https://en.wikipedia.org/wiki/Chord(peer-to-peer).

Requirements

You are required to implement functions to:

- Create the network ring (*create()* function mentioned in the paper) with *numNodes* number of nodes in it. *numNodes* will be passed as command line argument to the program. Each node in the network must be associated with an integer key. Also create finger tables for each node.
- Add nodes to the ring dynamically (*join()* function from the paper). Your code should update the finger tables with information about the new nodes that have joined the network.
- Function for scalable key lookup as described in the Chord paper (Section 4).
- A simulator for key lookups. In the simulator, each node must perform *numRequests* number of requests. *numRequests* will be passed as a command line argument to the program. Count the number of hops required for each request made by every node, sum it up and find the average number of hops as:

$$\text{Average number of hops} = \frac{\text{Sum of number of no. of hops for all requests for all nodes}}{numRequests * numNodes}$$

Input

The input provided (as command line arguments to Program.fs) will be of the form:

```
dotnet run numNodes numRequests
```

Where *numNodes* is the number of peers to be created in the peer-to-peer system and *numRequests* is the number of requests each peer has to make. When all peers have performed those many requests, the program can exit. Each peer should send a request/second.

Output

Print the average number of hops (node connections) that are made to deliver a message.

Actor model

In this project, you have to use exclusively the AKKA actor framework (projects that do not use multiple actors or use any other form of parallelism will receive no credit). You should have one actor for each of the peers modeled.

Report

In the report, include the following:

- Team members
- How to run your program?
- What is working?
- Attach screenshots of the output you get.
- A table of the average hop count results you obtained by executing your program with various number of nodes.
- A graph of "number of nodes" vs "average hop count" that your program outputs.
- Any assumptions you have made about the protocol.
- What is the largest network you managed to deal with?

Submission Guidelines

1. The project folder should be called Chord. It should contain a Program.fs file which would be the entry point of your program and some other configuration files which are included when the project is created.
2. Do not include any executable files in your submission.
3. Include the report in .pdf or .txt format. Name it report.pdf/txt.
4. Zip all your files into a packet: Team_ID.zip
5. Upload the zip packet as attachment in CANVAS before deadline.

Your project structure should be like this:

- Team_ID.zip
  - Chord
    - Program.fs
    - Chord.fsx
    - <Other configuration files and folders>
  - Report.pdf

Grading Criteria

Correct Implementation / Outputs	80%
Report	15%
Readability / Comments / Code structure	5%
Total	**100%**