COP5615 - Fall 2019

Project 3 - Tapestry algorithm

Alin Dobra

October 3, 2019

- **Due Date**: October 22, 2019
- One Submission per group
- Submit using eLearning
- **What to Include**:
  - **README** file including group members, other requirements specified below
  - Project3.tgz the code for the project
  - Project3-bonus.tgz the code for the bonus part, if any

1. Problem Definition

We talked extensively in class about the overlay networks and how they can be used to provide services. The goal of this project is to implement in Elixir using the actor model the Tapestry Algorithm and a simple object access service to prove its usefulness. The specification of the Tapestry protocol can be found in the paper-
Tapestry: A Resilient Global-Scale Overlay for Service Deployment by Ben Y. Zhao, Ling Huang, Jeremy Stribling, Sean C. Rhea, Anthony D. Joseph and John D. Kubiatowicz. Link to paper- *https://pdos.csail.mit.edu/~strib/docs/tapestry/tapestry_jsac03.pdf*. You can also refer to Wikipedia page: *https://en.wikipedia.org/wiki/Tapestry_(DHT)* . Here is other useful link: *https://heim.ifi.uio.no/michawe/teaching/p2p-ws08/p2p-5-6.pdf* .
Here is a survey paper on comparison of peer-to-peer overlay network schemes- *https://zoo.cs.yale.edu/classes/cs426/2017/bib/lua05survey.pdf*.

You have to implement the network join and routing as described in the Tapestry paper (Section 3: TAPESTRY ALGORITHMS). You can change the message type sent and the specific activity as long as you implement it using a similar API to the one described in the paper.

2. Requirements

Input: The input provided (as command line to your program will be of the form:
mix run project3.exs *numNodes numRequests*

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

Output: Print the average number of hops (node connections) that must be traversed to deliver a message.

Actor modeling: In this project you have to use exclusively the actor facility in Elixir (i.e. GenServer) **(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.

README file: In the README file you have to include the following material:

- Team members
- What is working
- What is the largest network you managed to deal with

3. BONUS

In the above assignment, there is no failure at all. For a 20% bonus, implement node and failure models (a node dies, a connection dies temporarily or permanently). Dealing with failures are described in Section 3 of the paper. Write a report describing how you tested that the systems are resilient and your findings.
(Also, set up failure parameter as an input argument so your bonus results can be tested without having to modify code)