Content Addressable Network

Mona Mishra

The project is done in Java using the features of Java Swing. The functionalities implemented in this program are:

1. Node Addition
2. Node Deletion

This program is divided into 3 packages:

1. DAO: This package contains the details of design of all the data that is used while building the network. It contains 3 classes: 1. Point, 2. Zone, 3. Neighbor
2. Draw: This package contains the Swing classes that is used to visually display the network
3. Peer: This package contains the class that has the main implementation of CAN.

Input

The program displays a menu on execution. The menu contains four options:

1. Peer Addition: The program starts with 1 node. This node takes the IP address of the current system and creates a peer. Once option1 is selected, programs takes 2 inputs: PeerID and PeerIP. PeerID is a unique identifier and PeerIP is the IP address of the remote machine. For demonstration purpose, we provide the program with a dummy IP address. There is no validation on PeerIP.
2. Peer Deletion: This option asks for one input, i.e., PeerID. The program verifies that the PeerID provided is present in the network. Once it finds the PeerID, it performs deletion.
3. View Peer Structure: This option does not take any user input. It simply displays the visual representation of the network using java JFrame.
4. View Peer Details: This Option also does not take any user input. It simply displays the console version of option 3. It displays the details of Node coordinates and their neighbors.

Output:

The output of this program is displayed using the console as well as JFrame of Java Swing. The frame displays the geometric representation of network while the console displays the details of all the peers, for example, their coordinates, their zones and their neighbors.

Algorithm:

1. Node Addition:
   a. Take input PeerID and PeerIP

b. Checks if there is any node with similar PeerID
c. The program assigns a random coordinate space to the peer
d. It assigns a random node, say node1, to the new node
e. If the coordinate assigned to the peer is not in the zone of node 1 then find another random node
f. Repeat d and e till the new node find the node in whose zone its coordinates lie.
g. Split the old node to split with new node
h. Identify the neighbors of new Node
i. Update the neighbor's neighbor.

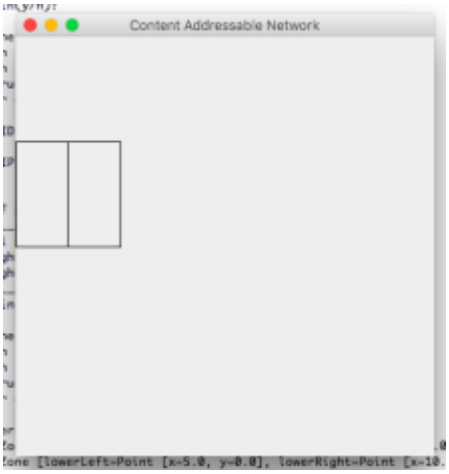
2. Node Deletion:
   a. Get the peerID as input and check if it is present in the network.
   b. If it is not present, repeat step a
   c. Once the departing node is recognized, find the node it should be dissolved in, say node1
   d. Update node1's neighbors
   e. Update departing node's other neighbors

Programs Update:
The program fully works when it comes to node addition and deletion. The option 4 of the menu is the proof of the correct functionalities of the program. The only thing that might have some errors is the visual representation of the network. Being a beginner in Java Swing, I would not say that it is completely correct, though, it seems to display correct result for the amount of testing I have done.

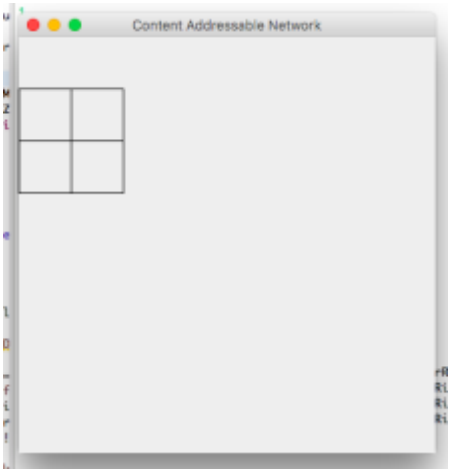
Below is the visual representation of CAN that is created by my program. I believe that this can serve as the evidence of correctness of my program.

1. *view Peer Network without any node addition*

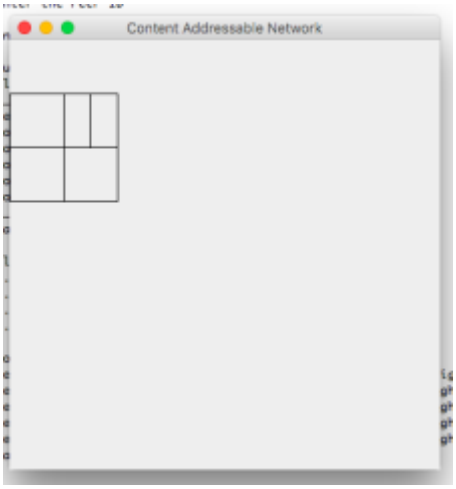
2. *Node Addition*



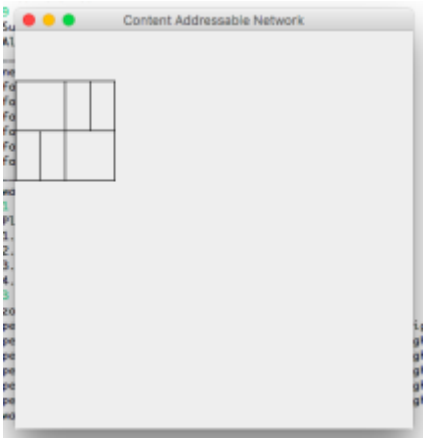
3. Node Addition



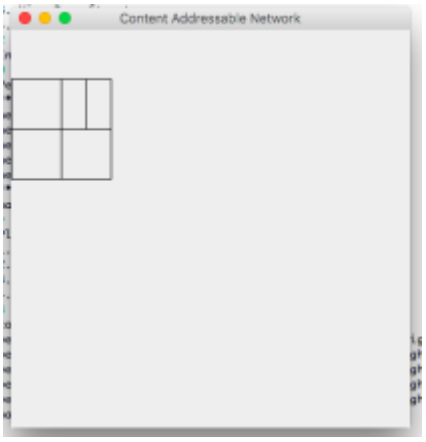
4. Node Addition



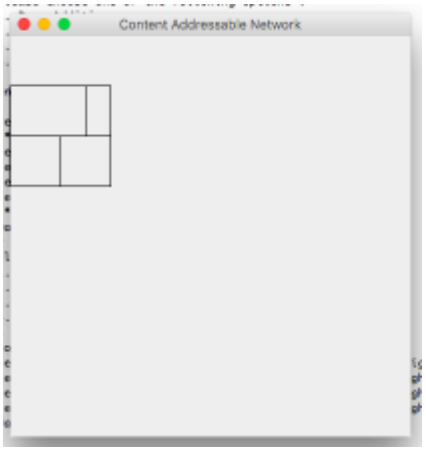
5. *Node Addition*



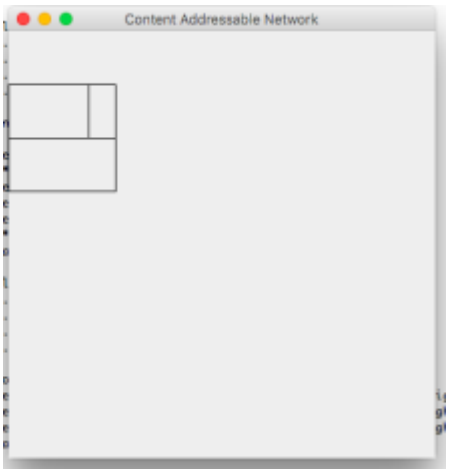
6. *Node Deletion*



7. *Node Deletion*



8. *Node Deletion*


Content Addressable Network