Content Server

In this project, I'm tasked to design a simple link-state routing protocol for an overlay network that connects machines that replicate common content. The first task of this project is to read a configuration file located in the same directory of the content server python file. When I run the command 'python content_server.py -c conffile.conf', the node should initialize itself and store its passive neighbors to be ready to link with them when they come alive. When initializing a node, I created variables for the node's identity and data structures to store map, neighbors, and rank. I also have lists for known peers, active peers, and passive peers. The second task is to identify a node by simply printing its uuid which is stored in a uuid variable. The third task is to make sure that neighbor nodes are reachable. I implemented the keep alive function that would send out an alive message periodically to its neighbors. Meanwhile, the node will also be listening to its neighbors to make sure that they are alive as well. With the last seen map, I store the names of the neighbors and the timestamp at which they last sent their keepalive message. If the difference between the last seen time and the current time is greater than the timeout interval, I will remove the neighbor from the neighbors map, overall map, and active peer list. The program will then add that inactive neighbor to the passive peers list and the neighbor will still be in the known peers list, in case it comes back alive. The link state advertisement function will send a packet with the node's name, uuid, and its neighbors. When the neighbors receive the link state message, they will update their map and other information accordingly. I also have an extra message indicator that will update every node's map constantly to make sure that I'm not losing any nodes.