Documented Design

Adjacency List

Used to store the connections between nodes. This is helpful when displaying the cables on screen and when calculating the path the packets of data should take from node to node.

The data structure I will use to store this data will be a dictionary containing multiple lists, with each node as the key and the nodes connected to it as the values. To remove a node from the adjacency list, the program will search through the keys and values of the dictionary and remove the node from them both.

Drag and drop algorithm

This algorithm will be used often in my program. It will work by detecting when the user’s mouse is over the object that they want to drag onto the canvas. Next, the program will wait for the user to press the left mouse button down while still hovering over the object. Once the user presses the mouse button down, the object will follow the mouse’s direction (by getting its coordinates, clearing the screen to prevent a trail of the object and then ‘blitting’ everything to the screen, including the object in its new position). Finally, the program will detect when the user has let go of the mouse button, indicating that they want to place the object in that position, and the object will then stop following the mouse and the function will end.

Path finder between connected nodes

This algorithm will store

Rearrange adjacency list so that nodes other than Switch, Hub, Servers are stored as from so that packets travel in same direction – FROM node to SWITCH, etc

if nodefrom and nodeto not in list:

if nodefrom in switch:

nodefrom = nodeto