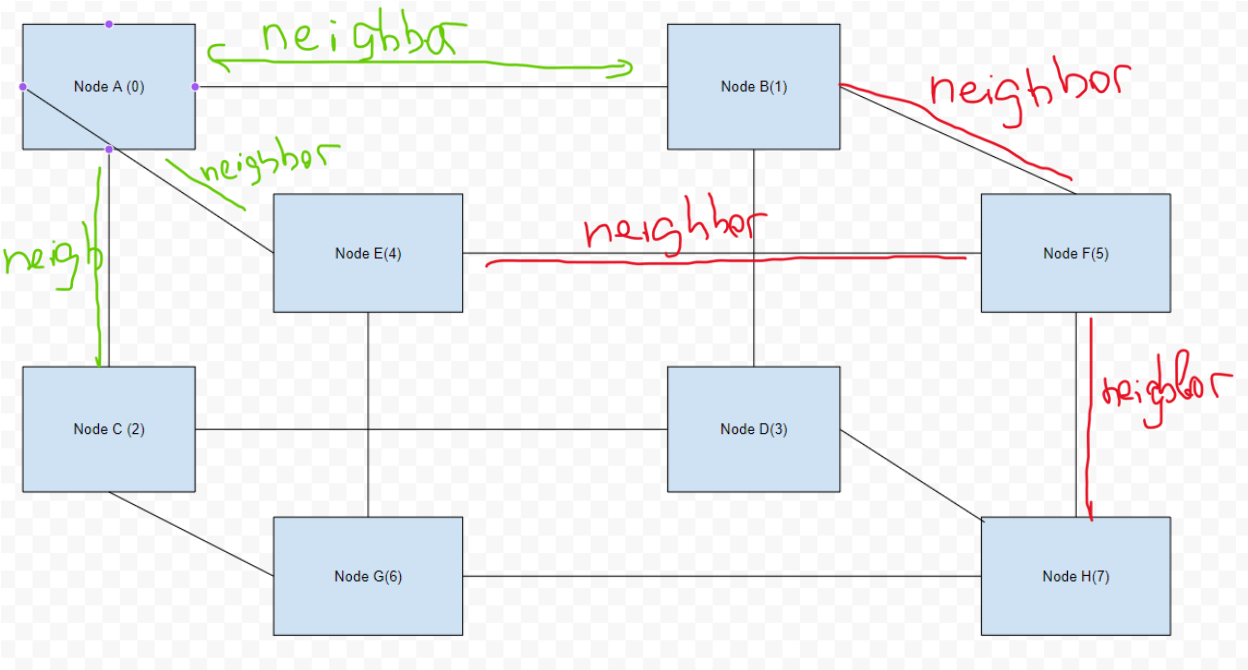
How program is designed

There are 8 peer nodes from:

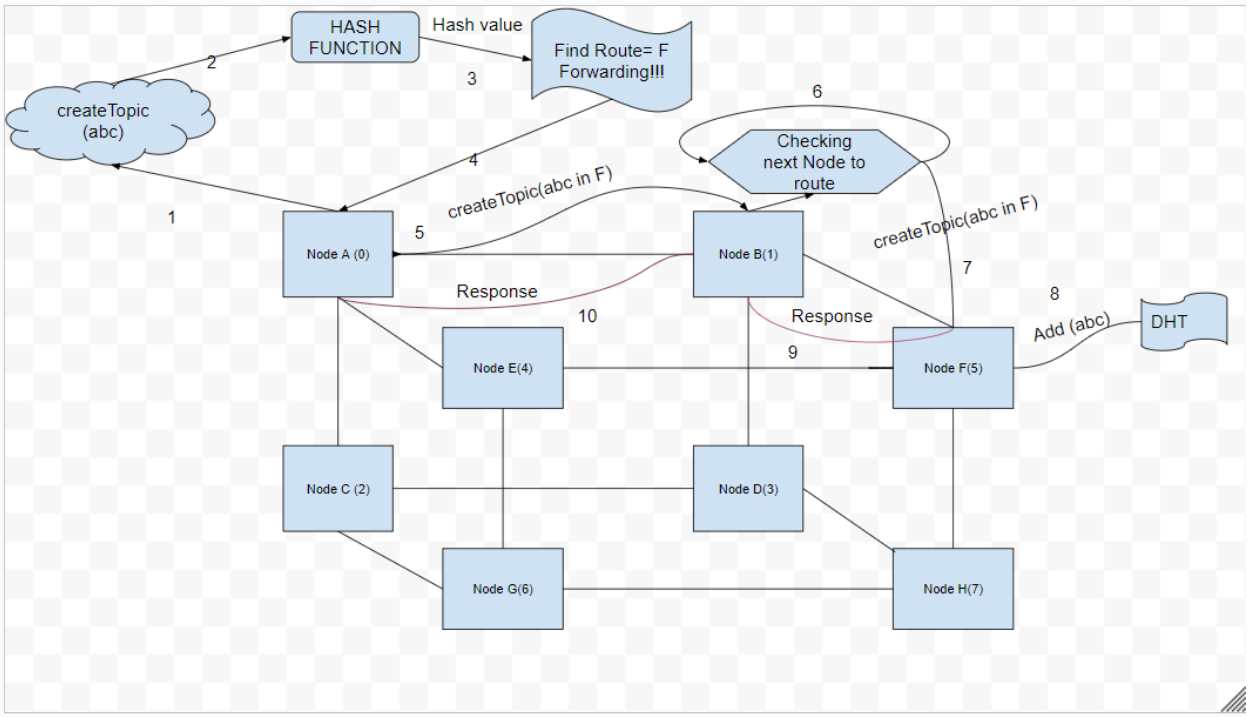
Node A – 0, Node B – 1, Node C – 2, Node D – 3, Node E- 4, Node F-5, Node G-6, Node H-7



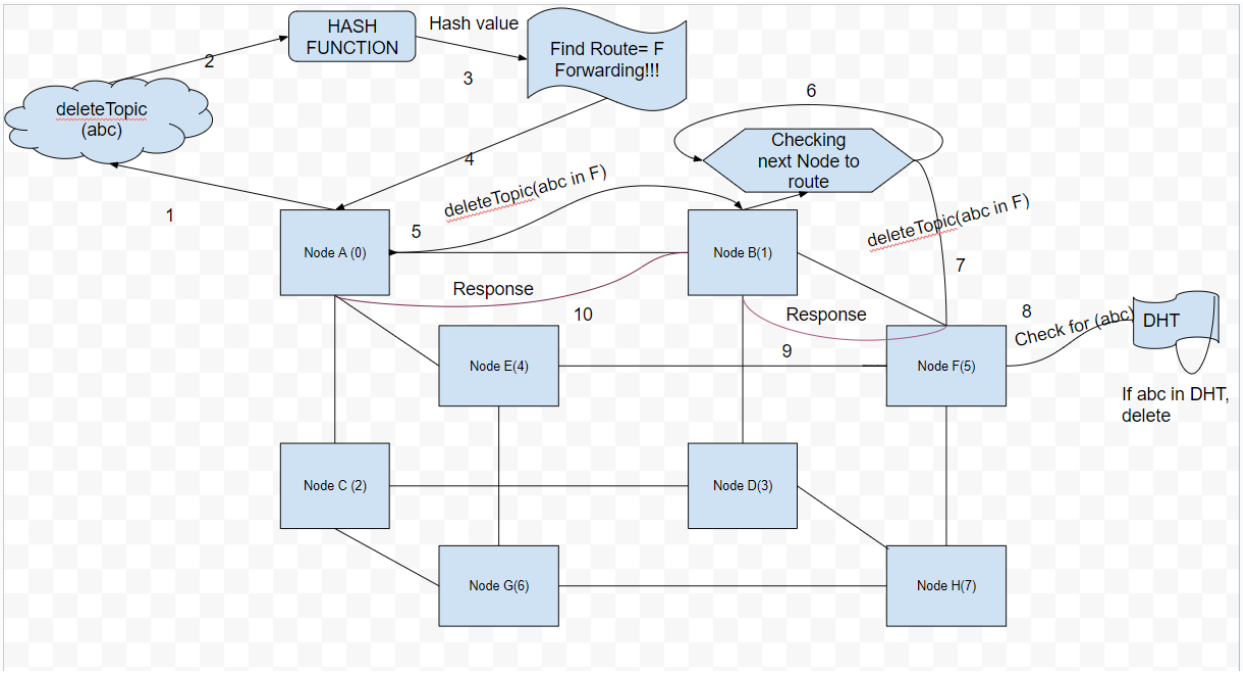
There are 5 available APIs : {createTopic, deleteTopic, send message, subscribe and pull}

Most of the scenarios are similar to each other that they follow the procedures and test cases.

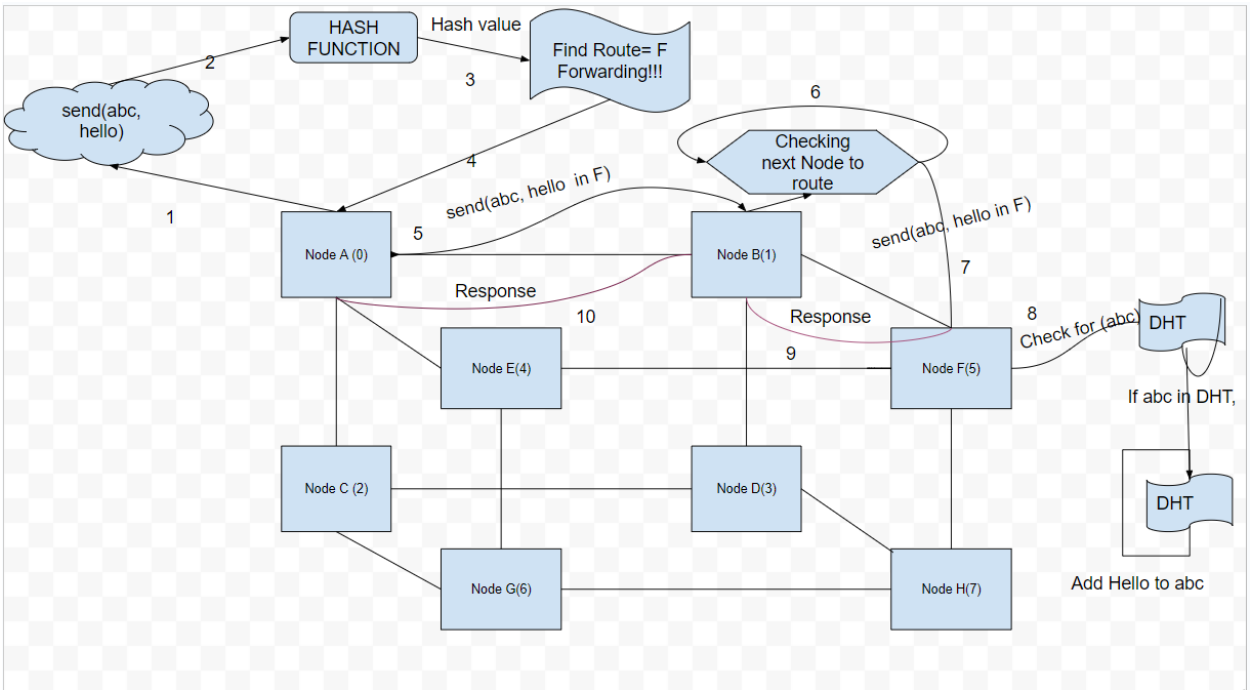
Create topic abc



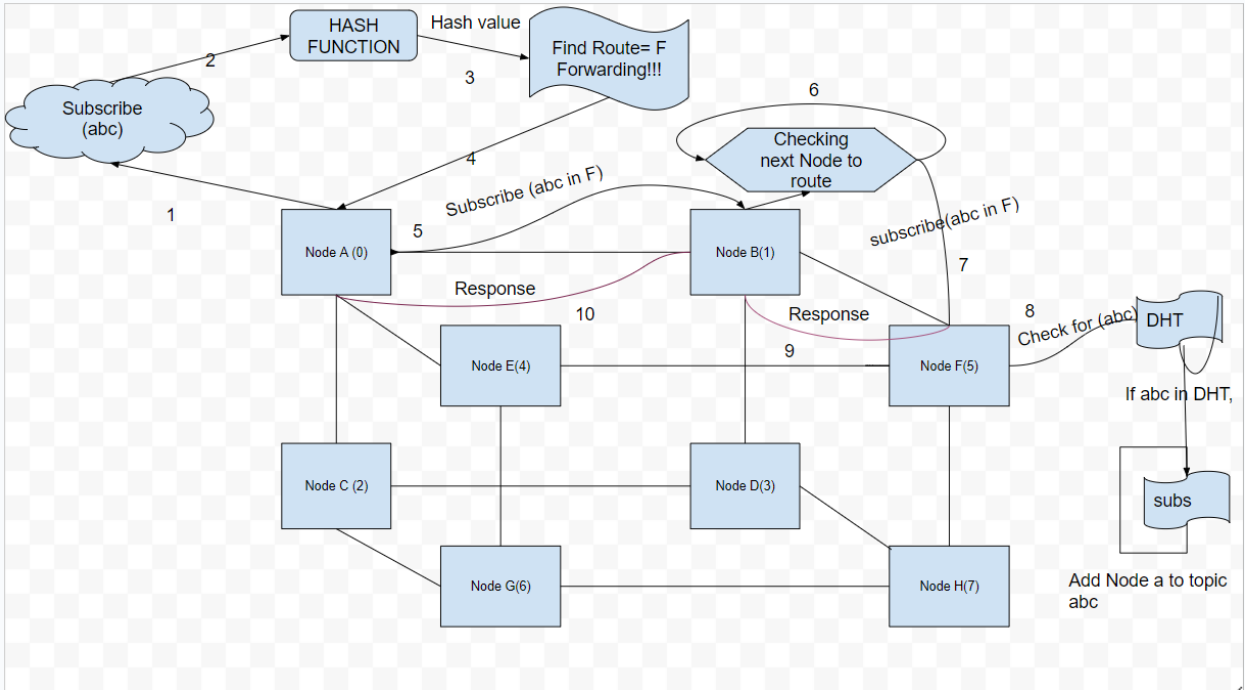
deleteTopic abc



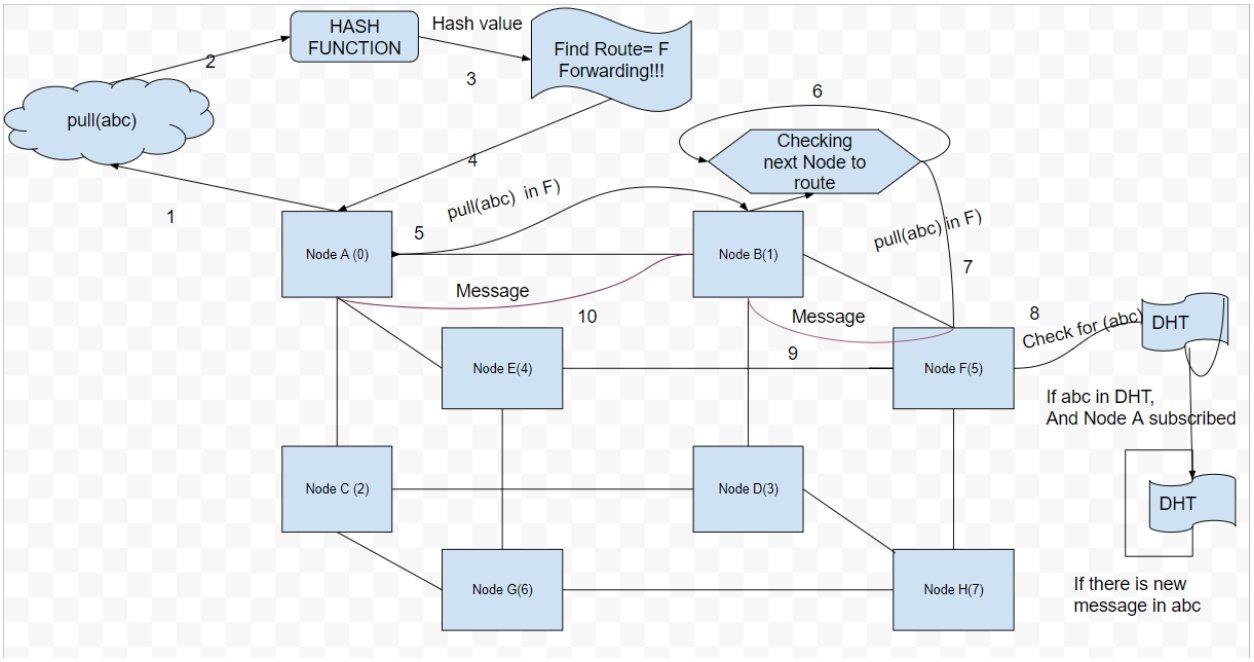
Send message Hello to topic abc



subscribe



Pull messages from topic abc



Possible improvement to the and extensions to the program and discussion

I think if the number of nodes goes up, I can introduce the central transferring node that will Distribute requests and message delivery evenly across peers to avoid bottlenecks. Even if there's some nodes that goes down, we can introduce the mechanism for message delivery. I believe there is a way to implement the adaptive routing mechanism that will consider the travel history and choose different commination nodes to reduce the latency on a single path. Introduction of multiple paths to the destination can lead to better network availability and reliability

APIs for this system is quite efficient, however bottleneck for the system is that is does choose specific routing path if there's forwarding needed. It means that in each case when any node needs to forward to the specific node, it will always follow the same forwarding route. It may lead to the latency issues if there's many nodes in the system. I think we should also keep track of the recently used path and routing mechanism always need to find the optimal path based on the workload/number of messages passed on a specific route. It will be beneficial to find the low latency and better performance routing.