CS 6390
PROGRAMMING PROJECT

DESIGN DOCUMENT

Group Members : 1) Hitesh Ginjupalli – hxg172430
2) M.V Chaitanya Eswar – vxm171330

For the **Implementation** of the project – we are going to use Python or Java. We are thinking about using Python but want to take professor's suggestion on which language would be the best to implement this project since multi threading and socket programming are involved.

We also need to use multithreading and socket programming (TCP and UDP sockets for communication between nodes).
We maintain multiple threads for listening ports(UDP and TCP) , processing an incoming request, forwarding a request as well as the periodic update. The TCP and UDP listening thread will listen for incoming requests from the IOT nodes and depending upon the maximum_response_times it will either process the request or forward the request to its best neighbor.
At each thread, we implement the respective job and after processing the request we add a message specifying the node that processed the request from the IOT node to the response. Once a connection opens we map the {neighbor_ip : socket} so that we can efficiently look up out of which socket we need to send the request.
We store the { node : [(neighbor_ip : port_number)]} mapping of each node to make the look up fast and efficient and easy to understand.

Choosing the best 1-hop Neighbor : When a fog node receives a request it will process or forward the request depending on the response time. So the fog nodes need to periodically exchange information with its 1-hop neighbors to have the latest queuing times and immediately determine who is the best neighbor and forward the request to it.

We maintain a Queue to store the incoming request to the fog nodes. We can determine the maximum response time(the fog node which has the smallest queue size). If the request has not reached the *Forward_Limit*, the fog node will identify the best neighbor among its 1-hop neighbors and forward the request

to that neighbor. The best neighbor is the one that reported the smallest response time(smaller size queue) in the most recent round with an exception that the best neighbor cannot be the neighbor who has forwarded this request to the current fog node(For this we maintain a list “visited” which specifies.

So once the current_limit(variable we initialize to store the current forward limit count) becomes equals to the forward_limit, we forward the request to the cloud.

We maintain a larger size queue at the cloud which can add any number of incoming requests to its queue and process them.

We can implement this using a Priority queue in which we can store the information (queuing delay, neighbor) in the priority queue. It is a min heap which pops and gives out the best neighbor based on the least queuing delay. So we can efficiently fetch the neighbor with the best queuing delay.

We periodically update the queuing delays for the neighbors in the queue so that the latest information about the best neighbor is readily available. If the best neighbor is the neighbor that the present fog node received the request from the request will be forwarded to the second best neighbor.

“”””””””””””””””””””””””

This is how we thought about implementing the project. Please review and let us know if we need to change our design.

“”””””””””””””””””””””””