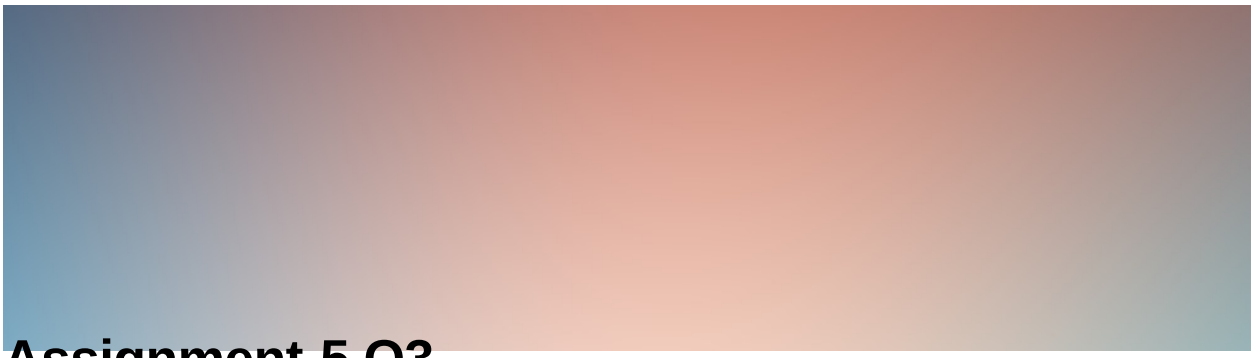

Assignment-5 Q3

Tags	ASSIGNMENT
contributors	rhythm

Q3-Internet Routing

A routing table is a set of rules, often viewed in table format, that is used to determine where data packets traveling over an Internet Protocol (IP) network will be directed. In this question you will be simulating the process of generating and using a routing table

- We first create a routing table for every node that stores the first forward edge to reach a destination from the source node. To create the routing table, we run dijikstra's algorithm n times for every node, calculate the shortest path and take the first edge of the path as the forward edge

```
vector<pair<int,int> > vec[n];
vector<pair<int,int> > routing[n];
dijikstras()
update_routing()
```

▼ Now we create a phthread for every edge of the node graph. Edge contains the necessary data like the client, server, their socket_fds, port for the connection and the routing table of the server node.

```
typedef struct edge_details
{
    pthread_t pid;
    int server_node;
    int client_node;
    int client_socked_fd;
    int server_socket_fd;
    int server_uses_this;
    vector<pair<int,int> > server_routing;
    int port;
}edge;
```

▼ in the thread function, a TCP connection is established between the client and the server taking help of the code given for the server.cpp and client.cpp. After the connection is established the thread server halts for reading message from the client node of the edge.

```
void *thread_function()
...
while(1)
    {
        string cmd;
```

```
        tie(cmd, received_num) = read_string_from_socket(client_socket_fd, buff_sz);
        curr_node = ((edge*)ed)->server_node;
        printf("Data received at node: %d : Source : %d : Destination : %d : Forwarded_Destination : %d : ",curr_node,((edge*)ed)->client_n
        cout <<"Message: " << cmd << "\n";
        if(curr_node == dest) continue;
        send_string_on_socket(edge_data[curr_node][((edge*)ed)->server_routing[dest].second]->client_socked_fd,cmd);
    }
...
```

▼ when the client receives message from a server, it sends the message to the server which is the respective forward node in the client's routing table. For message passing between 2 nodes of an edge we use the TCP pathway already established in the threads.

▼ So the server of the edge first reads the message from its client, after which is passes the message as a client to its forward node using the routing table I have pre calculated before.

Handling Server failures

In my implementation of the network communication, the client sends a message to the server, which the server receives. Now there are 2 cases of a node failure

(We assume that the communication channel between 2 nodes is secure and fail proof):

- **The server of a communication edge fails:** To handle this failure, after the client node sends message to the server it waits for an ACK message telling that the server received the message and the communication channel is active and the message send was right. The client waits till 2*delay(communication edge) + C(to account for message processing at the server). If the ACK is not received by the client it means that the server has failed and we need to update the routing tables of the client taking in account that the current forward node has failed.(run dijikstra's Algorithm) and then use the updated routing table to send messages to across the next shortest path. We mark the failed server with status -1 indicating that all the requests incoming to the server will have to be redirected to other servers. The failed server is put through a failure recovery routine where it communicates with the master server (in our network the welcome server listening to client setup in the client.cpp) and update its status whenever it resumes operations, after which the master server updates the routing table of each server again.
- **The client of a communication edge fails during sending message to the server:** To check this failure, the master server wait for delay(communication edge) + C time to check if the signal sent is received at the server end(check valid when the server is working fine). If the message is not received we know that the client couldn't send the message to the server. Thus the master retrieves the message through a backup routine and sends it to the server node to resume the communication further and mark the client node with status -1 indicating that the node has failed and all the requests to the node to be redirected to other servers. The routing tables are also updated by the master server accordingly.