Build your own router

In this exercise you'll build your own simulated Zigbee router. Most of the code will be provided to you, so for most part you'll build the network itself. Make sure you understand the code and do what you can to add print statements etc to follow the process along.

Use this code:

```
#include <iostream>
#include <vector>
#include <unordered_map>
#include <unordered_set>
#include <cstdint>

using std::cout;
using std::endl;
using std::uint16_t;
using std::vector;
using std::unordered_map;
using std::unordered_set;

struct RouteEntry {
    uint16_t destination;
    uint16_t nextHop;
    int hopCount;
};

class ZigbeeNode {
public:
    uint16_t address;
    vector<ZigbeeNode*> neighbors;
    vector<RouteEntry> routingTable;
    unordered_map<uint16_t, ZigbeeNode*> reversePath;

    explicit ZigbeeNode(uint16_t addr) : address(addr) {}

    void addNeighbor(ZigbeeNode* neighbor) {
        neighbors.push_back(neighbor);
    }
    void sendRouteRequest(uint16_t destination,
unordered_set<uint16_t>& visited) {
        cout << "[" << address << "] Broadcasting RREQ for
destination " << destination << endl;
        visited.insert(address);
        for (size_t i = 0; i < neighbors.size(); ++i) {
            if (!visited.count(neighbors[i]->address)) {
                neighbors[i]->receiveRouteRequest(address,
destination, address, 1, visited);
            }
        }
    }

    void receiveRouteRequest(uint16_t source, uint16_t
destination, uint16_t origin, int hopCount,
unordered_set<uint16_t>& visited) {
        reversePath[origin] = findNeighborByAddress(source);
        if (address == destination) {
            cout << "[" << address << "] Destination reached.
Sending RREP back to origin." << endl;
            receiveRouteReply(origin, address, hopCount);
        } else {
            visited.insert(address);
            for (size_t i = 0; i < neighbors.size(); ++i) {
                if (!visited.count(neighbors[i]->address)) {
                    neighbors[i]->receiveRouteRequest(address,
destination, origin, hopCount + 1, visited);
                }
            }
        }
    }

    void receiveRouteReply(uint16_t origin, uint16_t nextHop, int
hopCount) {
        cout << "[" << address << "] Received RREP to reach " <<
origin << " via " << nextHop << ", hops: " << hopCount << endl;
        routingTable.push_back({origin, nextHop, hopCount});
        if (reversePath.count(origin)) {
            reversePath[origin]->receiveRouteReply(origin,
address, hopCount + 1);
        }
    }

    void printRoutingTable() const {
        cout << "\n[Routing Table @ Node " << address << "]" <<
endl;
        for (size_t i = 0; i < routingTable.size(); ++i) {
            const RouteEntry& entry = routingTable[i];
            cout << "  Destination: " << entry.destination
                 << ", NextHop: " << entry.nextHop
                 << ", Hops: " << entry.hopCount << endl;
        }
    }

private:
    ZigbeeNode* findNeighborByAddress(uint16_t addr) {
        for (size_t i = 0; i < neighbors.size(); ++i) {
            if (neighbors[i]->address == addr) return
neighbors[i];
        }
        return nullptr;
    }
};

int main() {
    ZigbeeNode coordinator(0x0000);
    ZigbeeNode router1(0x1001);
    ZigbeeNode router2(0x1002);
    ZigbeeNode router3(0x1003);
    ZigbeeNode endDevice(0x2001);

    // This is how the coordinator and router1 get connected.
    coordinator.addNeighbor(&router1);
    router1.addNeighbor(&coordinator);

    // Continue building the network here.

    unordered_set<uint16_t> visited;
    router2.sendRouteRequest(0x2001, visited);

    router2.printRoutingTable();
    router3.printRoutingTable();
    endDevice.printRoutingTable();

    return 0;
}
```

Start by doing my work for me. Look through the class Zigbee node, understand it's methods and document them.

Then try to create a small tree-like network using the 5 nodes that has already been created for you. Run the program and see what happens.

Now try to make the network more complicated by creating a mesh network. How is that different? Sketch the network out on paper first!

Finally see if you can add the possibility for links to be broken. How would that affect things?

As a bonus assignment, try to visualize the network!