NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY
COLLEGE OF ELECTRICAL AND MECHANICAL ENGINEERING
Department of Computer & Software Engineering
Complex Engineering Problem (CLO5->PLO3)

NOTE:

- **Students should score 50% in OBE specific questions to ensure their accumulated scores towards respective PLOs are above 50%**
- **As part of OBE paradigm, in this course, you are expected to learn complex concepts and principles through complex Engineering Problem as your Semester Project. This problem helps students acquire depth of knowledge (WP1), Depth of Analysis (WP3) and Range of Conflicting Requirements (WP2).**

Designing topology and implement routing algorithm of wireless Network

A wireless network consists of a collection of devices that are equipped with radio transmitters and receivers. We shall refer to these devices as nodes. Pairs of these nodes can communicate with each other through radio signals, provided they are within each other's transmission ranges. A pair of nodes that are physically far away from each other may be able to use other nodes as intermediaries" through which messages can be routed. Various models of wireless networks have been proposed, so that wireless networks and protocols on these networks can be mathematically analyzed. A simple model for wireless networks is the disk graph. Each vertex in the disk graph is a point in the plane and associated with each vertex u is a positive real number t(u), that denotes the transmission range of u. For any pair of vertices u and v, there is a directed edge (u; v) in the disk graph provided the distance between u and v, denoted |uv|, is no greater than t(u). Thus, the edge (u; v) represents the fact that v is in the transmission range of u and therefore u can send messages directly to v. Note that while v may be in u's transmission range, u may not be in v's transmission range, implying that the disk graph may contain edge (u; v), but not edge (v; u). If the transmission ranges for all the vertices are equal then we call the graph a unit disk graph (in short, a UDG). Note that all edges in a UDG go both ways, that is, (u; v) is an edge if and only if (v; u) is an edge. Therefore, a UDG can be thought of as an undirected graph. See Figure 1 for an illustration of a UDG.

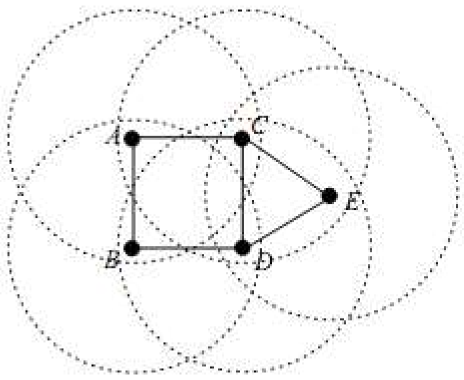

Figure 1: Figure 1: A unit disk graph with 5 vertices. The edges are defined by the disks shown in the Figure. For example, the disk of which B is the center contains A and D as well and this implies the edges {A,B} and {B,D}.

Disk graphs and UDGs are simple models of wireless networks and there is a large body of recent research on wireless networks that uses these models. Also, many researchers use these models to experimentally simulate various protocols designed for wireless networks. In the projects for this course, you will have to simulate some recently proposed protocols for wireless networks. Project has 3 parts: (I) UDG generation, (II) topology control, and (III) routing. For this project, we will only be dealing with UDGs and not with arbitrary disk graphs, and therefore it is sufficient to implement undirected graphs. You should use the most effective and time efficient representation for your data holder.

The wireless Network

You are required to implement wireless Network that provides methods for generating wireless networks and for simulating protocols on these networks. The wireless Network structure would use the Graph ADT extensively. For this project, the wireless Network is required to contain the following three methods.

- A loader for generating a wireless network.
- A method for simulating a protocol for topology control.
- A method to perform compass routing on the network.

More details are provided below.

- **Generating wireless networks**:

The loader should generate a UDG obtained by distributing n points, uniformly at random, on a square of dimensions size square units. For example, if n were 1000 and size were 10, then the constructor would distribute 1000 points uniformly at random on a 10 X 10 square. By "uniformly at random" I mean that the likelihood of a point falling anywhere in the square is the same. You should use the class Random to distribute points in this way. Each pair of points at distance at most 1 unit are connected by an edge. This means that each point represents a wireless device with a transmission range of 1 unit. In addition to the above loader, also provide a "default" loader that takes no parameters, and generates the UDG obtained by distributing 500 points, uniformly at random, on a 10 x 10 square.

- **Topology control:**

In wired networks, more the connections between nodes, the better it is. However, in wireless networks this is not the case. If lots of nodes are physically close together and start attempting to communicate with each other, there is interference between the signals and nodes end up receiving garbled messages. To reduce interference, each node in a wireless network chooses a subset of the nodes in its transmission range and decides just to communicate with these. The protocol that each node uses to choose a subset of nodes to communicate with is called a topology control protocol. At the end of topology control, we end up with a network that has all the nodes in the original network, but far fewer edges.

I would like you to implement a simple topology control protocol called XTC (R. Wattenhofer and A.Zollinger, "XTC: A practical topology control algorithm for ad-hoc networks,WMAN 2004). a simple rule to determine whether an edge {u, v} should be kept or dropped. The authors of the XTC paper prove that after topology control is run on G, the output graph H has a number of properties very useful for routing. Some of these are:

- if G is connected, then H is connected.

- every vertex in H has at most 6 neighbors.
- H is planar, that is, no two edges in H cross

➢ **Compass routing**:

A wired network typically consists of nodes with considerable computational power and these devices can compute and store large routing tables that facilitate routing. Suppose a node receives a message m intended for a destination t. The node immediately looks up its routing table to see which neighbor to forward m to, so that it will get to t. A wireless network typically consists of nodes that have limited computational power and limited amount of memory. Therefore, doing routing with the help of routing tables is not a feasible option for wireless networks. In a wireless network, when a node receives a message m intended for a destination t, it needs to decide which neighbor to forward m to without the help of a routing table. One “greedy" approach to this problem attempts to use geometric information and do compass routing (E. Kranakis, H. Singh and J. Urrutia, “Compass routing on geometric networks", Proceedings of the 11th Canadian Conference on Computational Geometry, 1999.)

Here is a description of how compass routing works. Suppose a node x receives a message **m** intended for destination t. Further suppose that x knows its location, the location of t, and the location of its neighbors in the plane. Then x sends m to a neighbor v that minimizes the angle <txv. The intuition is that we always want to travel in a direction that is as close to the direction xt as possible. See figure for illustration of compass routing. In the paper on compass routing mentioned above, it is also shown that compass routing does not always work and there are instances for which a message may forever cycle among a set of nodes without reaching the destination. You should simulate compass routing by implementing a method in the wireless Network ADT that takes as parameters a source vertex s and a destination vertex t. The method should either return a path from s to t that was discovered by compass routing or should return an indication that no path from s to t could be found. There is a simple recursive implementation of compass routing based on the following idea: if the current vertex x equals t, then we are done; otherwise find a neighbor v of x that makes the smallest angle with the direction xt and do compass routing with source v and destination t.

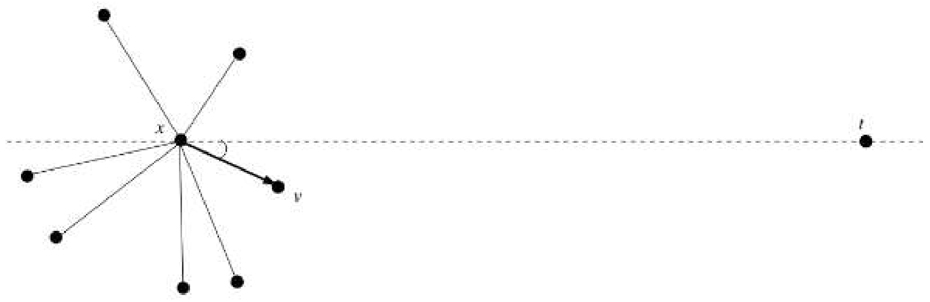

Figure 2: The node x, on receiving a message m, intended for the destination t, forwards m to a neighbor v that makes the smallest angle with xt~ .

To summarize, the wirelessNetwork class should at least contain the following methods.

wirelessNetwork();

wirelessNetwork(float size, int n);

wirelessNetwork topologyControl();

void compassRouting(int s, int t,int []);

The function headers given above should be treated as rough guides; if there are other parameters you feel the need to pass or if you want to pass parameters differently, feel free to do so. You may need to implement additional “helper" functions in the wirelessNetwork ADT.

Experiments:

Using your implementation of the wirelessNetwork ADT, you are required to perform the following experiments.

1. Generate 10 wireless networks modeled as UDGs, by distributing n = 500; 550; 600; 650; 700; 750; 800; 850; 900; 950 points on a 10 x 10 grid. For each network, report the average degree and the maximum degree. Recall that the degree of a vertex is equal to the number of neighbors it has. Plot these, showing how they increase with respect to n. Run topology control on each of these networks to produce 10 new and much sparser networks. Report and plot the average and maximum degree of these 10 networks as well. Based on your data and plots, write 2-3 sentences commenting on whether performing topology control has resulted in a much sparser graph or not.

2. Generate a wireless network G with 1000 points distributed on a 10x10 square. Then repeat the following 10 times. Uniformly at random, pick a vertex and designate it the source s and again uniformly at random, pick a vertex and designate it the destination t. Run compass routing on G, with source s and destination t. Report on whether compass routing was able to find the destination or not and if it was, report on the length of the path from s to t that was discovered. Present your results in a tabular form.

3. Start with the same wireless network G generated for Experiment (2) and run topology control on it to get a sparser network H. Run Experiment (2) again but use the network H this time. Tabulate your results as for Experiment (2). Write 2-3 sentences commenting on how the path lengths generated in Experiment (3), compare with path lengths generated in Experiment (2). Try to explain your observations.

Place the code for your experiments in a single file called experiments.cpp. The code in the file experiments.cpp makes extensive calls to methods in the wirelessNetwork ADT and possibly to methods in the myGraph ADT as well.