# Experiment 3: Neighbor Table Determination

**Aim:** To create neighbor table for a given network topology

**Objective:** After carrying out this experiment, students will be able to:

- Generate neighbor table for all the nodes in a given topology.
- Analyze how this is useful in the process of routing data

**Problem statement:** You are required to write a program that calculates neighbor table for all the nodes in a given network. Consider a network with 10 nodes that is deployed in an area of 500 $m^2$. Your program should initially determine the distance between each node and all other nodes. Then the range of the nodes is given as input to the user. Using this range information, determine the neighbors of all the nodes.

**Analysis:** While analyzing your program, you are required to address the following points:

- How this is useful in the process of routing data?
- For a 3D topology, how would your program need to be changed?

**MARKS DISTRIBUTION**

| Component | Maximum Marks | Marks Obtained |
|---|---|---|
| Preparation of Document | 7 | |
| Results | 7 | |
| Viva | 6 | |
| **Total** | **20** | |

Submitted by: Subhendu Maji

Register No: 18ETCS002121

1. Algorithm/Flowchart

The algorithm for the neighbour table determination program is as follows:

```
STEP 1: Start

STEP 2: x, y, distance ← [], count ← 0, inRange ← “”

STEP 3: nodes, gridSizes, commRange ← from user

STEP 4: for i from 0 to nodes, do

            4.1: x[i] ← random number % (gridSize + 1)

            4.2: y[i] ← random number % (gridSize + 1)

STEP 5: for i from 0 to nodes, do

            5.1: for j from 0 to nodes, do

                  5.1.1: distance[count] ← √((x[i] − x[j])² + (y[i] − y[j])²)

                5.1.2: if distance[count] <= commRange, inRange =
                “Yes”, else inRange = “No”

                5.1.3: print result

                5.1.4: count++

STEP 6: Stop
```

2. Program



```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <time.h>

#define MAX_NODES 1000

int main() {
    // declaring variables
    int x[MAX_NODES], y[MAX_NODES];
    int count = 0;
    int nodes, gridSize;

    float distance[MAX_NODES];
    float commRange = 0;
    char* inRange;

    // seeding random number generator
    srand(0);

    // gathering user input
    printf("Enter the number of nodes: ");
    scanf("%d", &nodes);
    printf("Enter the grid size: ");
    scanf("%d", &gridSize);
    printf("Enter the communication range: ");
    scanf("%f", &commRange);

    // generating random coordinates for nodes
    for (int i = 0; i < nodes; i++) {
        x[i] = rand() % (gridSize + 1);
        y[i] = rand() % (gridSize + 1);
    }

    printf("Nodes\tCoordinates\t\t\tWithin Range\tDistance between nodes\n");
    // generating neighbour table
    for (int i = 0; i < nodes; i++) {
        for (int j = 0; j < nodes; j++) {
            // finding distance between ith node and jth node
            distance[count] = sqrt(pow(x[i] - x[j], 2) + pow(y[i] - y[j], 2));

            // checking if distance is within range
            inRange = (distance[count] <= commRange) ? "Yes" : "No";

            // printing result
            printf(
                "%d & %d\t(%d, %d) & (%d, %d)\t\t\t%s\t\t%.2f\n",
                i+1, j+1, x[i], y[i], x[j], y[j], inRange,
                distance[count]
            );

            count++;
        }
    }
}

```

*Figure 1 source code*



3. Results

makefile

```
all:
    @gcc lab3.c -lm

run: all
    @./a.out

clean:
    @rm *.out

.PHONY: clean
```

*Figure 2 makefile*

**Note:** No. of nodes = 10 prints a big output (difficult to paste screenshot). Hence running the program with no. of nodes = 5 for simplicity.

```
 subhendu@LAPTOP-AL8CTHTV  /mnt/d/RUAS-sem-05/CN/lab03  make run
Enter the number of nodes: 5
Enter the grid size: 3
Enter the communication range: 2
Nodes   Coordinates                    Within Range    Distance between nodes
1 & 1   (3, 2) & (3, 2)                Yes             0.00
1 & 2   (3, 2) & (1, 3)                No              2.24
1 & 3   (3, 2) & (1, 3)                No              2.24
1 & 4   (3, 2) & (2, 0)                No              2.24
1 & 5   (3, 2) & (1, 1)                No              2.24
2 & 1   (1, 3) & (3, 2)                No              2.24
2 & 2   (1, 3) & (1, 3)                Yes             0.00
2 & 3   (1, 3) & (1, 3)                Yes             0.00
2 & 4   (1, 3) & (2, 0)                No              3.16
2 & 5   (1, 3) & (1, 1)                Yes             2.00
3 & 1   (1, 3) & (3, 2)                No              2.24
3 & 2   (1, 3) & (1, 3)                Yes             0.00
3 & 3   (1, 3) & (1, 3)                Yes             0.00
3 & 4   (1, 3) & (2, 0)                No              3.16
3 & 5   (1, 3) & (1, 1)                Yes             2.00
4 & 1   (2, 0) & (3, 2)                No              2.24
4 & 2   (2, 0) & (1, 3)                No              3.16
4 & 3   (2, 0) & (1, 3)                No              3.16
4 & 4   (2, 0) & (2, 0)                Yes             0.00
4 & 5   (2, 0) & (1, 1)                Yes             1.41
5 & 1   (1, 1) & (3, 2)                No              2.24
5 & 2   (1, 1) & (1, 3)                Yes             2.00
5 & 3   (1, 1) & (1, 3)                Yes             2.00
5 & 4   (1, 1) & (2, 0)                Yes             1.41
5 & 5   (1, 1) & (1, 1)                Yes             0.00
 subhendu@LAPTOP-AL8CTHTV  /mnt/d/RUAS-sem-05/CN/lab03
```

*Figure 3 Execution of program with no. of nodes = 5*

## 4. Analysis and Discussions

This experiment involves creating a routing table for nodes in a network using a naïve approach. This table is used by other nodes to determine whether they can communicate with each other, their position and the distances between different nodes.

The nodes calculate the distance between each other using the distance formula:

$$distance = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$



If the nodes are within range of each other, they can communicate. The grid size is taken from the user and x and y coordinates are randomly chosen according to the data given. The distances between the nodes are calculated using the distance formula. Using this data, the neighbour table from the calculations.

For a 3D topology, another dimension must me taken into account. The distance formula becomes:

$$distance = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$$

5. Conclusions

   The neighbour table for nodes in a network was successfully created and the program to do so was implemented.

6. Comments

   **a. Limitations of Experiment**

   The program does not take into account the multitude of other factors that determine whether nodes can communicate with each other.

   **b. Limitations of Results**

   The table generation is based on static data.

   **c. Learning Happened**

   The method to construct the neighbour table using distance formula was learned.