**INTRODUCTION**

Wireless network has become increasingly popular during the past decades. There are two variations of wireless networks-infrastructure and infrastructure less networks. In the former, communications among terminals are established and maintained through centric controllers. Examples include the cellular networks and wireless Local Networks (IEEE802.11). The latter variation is commonly referred to as wireless Ad-hoc network. Such a network is organized in an Ad-hoc manner, where terminals are capable of establishing connections by themselves and communicate with each other in a multi-hop manner without the help of infrastructures. This infrastructure less property makes an ad hoc networks be quickly deployed in a given area and provides robust operation. Example applications include emergency services, disaster recovery, wireless sensor networks and home networking. Communication has become very important for exchanging information between people from, to anywhere at any time. MANET is group of mobile nodes that form a network independently of any centralized administration. Since those mobile devices are battery operated and extending the battery lifetime has become an important aim. Most of the researchers have recently started to consider power-aware development of efficient protocols for MANETs. As each mobile node in a MANETs performs the routing function for establishing communication among different mobile nodes the ``death’’ of even a few of the nodes due to power exhaustion might cause disconnect of services in the entire MANETs. So, Mobile nodes in MANETs are battery driven. Thus, they suffer from limited energy level problems. Also the nodes in the network are moving if a node moves out of the radio range of the other node, the link between them is broken. Thus, in such an environment there are two major reasons of a link breakage.

**SYSTEM STUDY**

In the recent years, one could assist to a spectacular growth in the use of wireless equipment. The number of mobile devices such as PDAs, mobile phones laptops, is also tremendously increasing. To ensure the connectivity between all these devices, ad hoc networks appear to be a promising solution. An ad hoc network is a collection of wireless mobile nodes, which communicate together without the assistance of any fixed nor central infrastructure. Thus, participants without the assistance of any fixed nor central infrastructure. Thus, participants are not within the same radio range. MANET can be used in scenarios where no infrastructure exists, or where the existing infrastructure does not meet application requirements for different reasons such as security, cost or quality. They also open new fields of applications in the domain of networking like battlefield operations, sensor networks, emergency rescues or roaming networking. A MANET with the characteristics described above was originally developed for military purposes, as nodes are scattered across a battlefield and there is no infrastructure to help them form a network. In recent years MANETs have been developing rapidly and are increasingly being used in many applications, ranging from military to civilian and commercial uses since setting up such networks can be done without the help of any infrastructure or interaction with a human such as search-and-rescue missions, data collection, and virtual classrooms and conferences where lap tops, PDA or other mobile devices share wireless medium and communicate to each other.

**2.1 EXISTING SYSTEM**

Wireless network has become increasingly popular during the past decades. There are two variations of wireless networks infrastructure and infrastructure less networks. In the former, communications among terminals are established and maintained through centric controllers. Examples include the cellular networks and wireless Local Networks (IEEE802.11). the latter variation is commonly referred to as wireless adhoc network. Such a network is organized in an adhoc manner, where terminals are capable of establishing connections by themselves and communicate with each other in a multi hop manner without the help of fixed infrastructures. This infrastructure less property makes an adhoc networks be quickly deployed in a given area and provides robust operations. Example applications include emergency services, disaster recovery, wireless sensor networks and home networking. Communication has become very important for exchanging information between people from, to anywhere at any time. MANET is group of mobile nodes that form a network independently of any centralized administration. Since those mobile devices are battery operated. The problem is battery driven routing.

**2.2 PROPOSED SYSTEM**

As technology rapidly increases, diverse sensing and mobility capabilities have become readily available to devices and, consequently, mobile ad hoc networks (MANETs) are being deployed to perform a number of important tasks. In MANET, power aware is important challenge issue to improve the communication energy efficiency at individual nodes. We propose Low Energy Adaptive Clustering Hierarchy (LEACH), a new power aware routing protocol that increases the network lifetime of MANET. In contrast to conventional power aware algorithms, LEACH identifies the capacity of a node not just by its residual battery power, but also by the expected energy spent in reliably forwarding data packets over a specific link. Using a min-max formulation, LEACH selects the path that has the largest packet capacity at the smallest residual packet transmission capacity. This protocol must be able to handle high mobility of the nodes that often cause changes in the network topology. This paper evaluates three ad hoc network routing protocols (LEACH, AODV, and DSR) in different network scales, taking into consideration the power consumption. Indeed, our proposed scheme reduces for more than 20% the total energy consumption and decreases the mean delay, especially for high load networks, while achieving a good packet delivery ratio.

**2.3 FEASIBILITY STUDY**

The feasibility of the project is analyzed in this phase. The objective is to showcase the strengths of the project. The key things to observe are, the energy of each and every node gradually decreases due to its mobility and for being an intermediate node. So maintaining a complete record of every node’s energy would help in selecting the optimized path. The main strength is, if a sender node wants to transmit a date packet to a receiver, then he has to use the intermediate nodes. So if he has the list of every node’s energy then he has the ability to choose the path of nodes with higher energy. So that the intermediate nodes won’t collapse and data loss doesn’t occur.

**2.4 SYSTEM SPECIFICATIONS**

**2.4.1 SOFTWARE REQUIREMENTS**

* Network Simulator 2.35.
* Network Animator.
* Operating system: Ubuntu LTS 14.04.

**2.4.2 HARDWARE REQUIREMENTS**

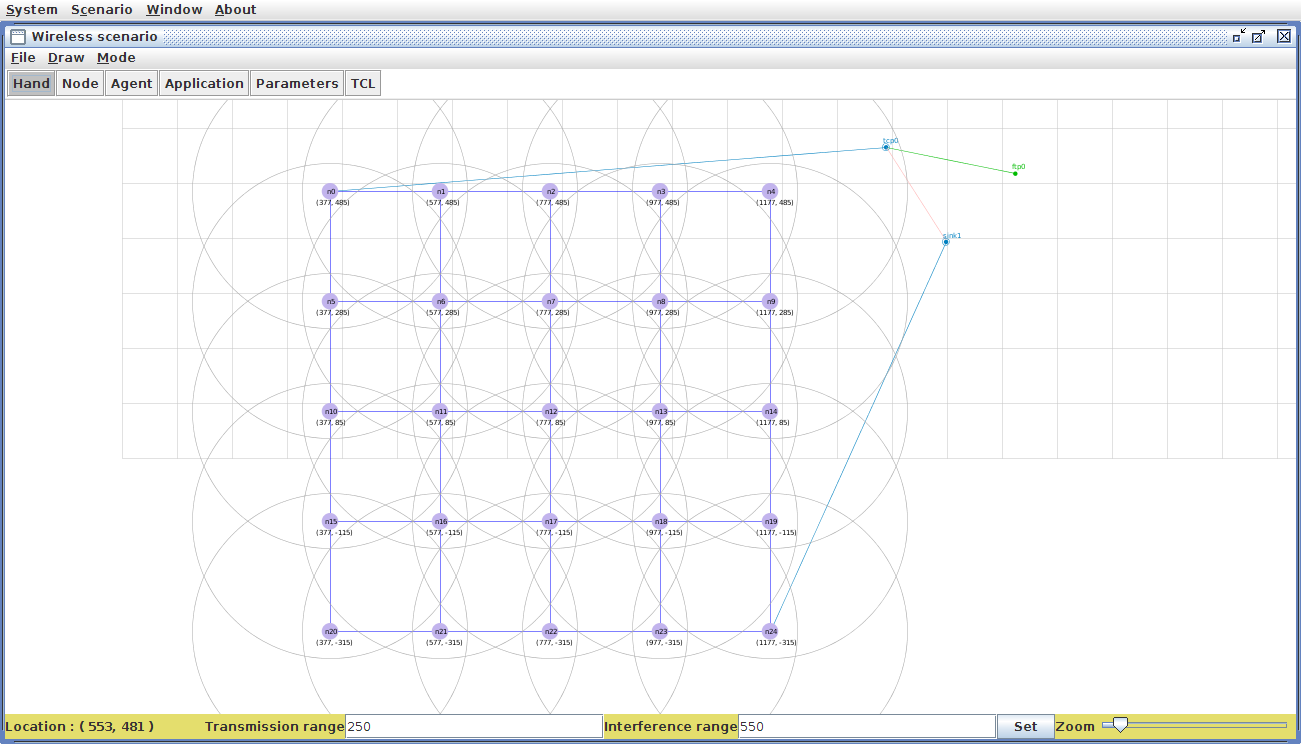
* Processor Pentium IV and above.
* RAM 1 GB
* Hard disk 100 GB.

**SYSTEM DESIGN**

**3.1 OVERVIEW OF THE PROJECT**

**3.1.1 INPUT DESIGN**

This is the actual network area representation. This is done using Network simulator generator application (NSG). This figure represents a 5X5 node grid. Specify the source and destination nodes for the network data packets transmission. Mobility of the nodes also give at the network. So that which node receives the packet can be identified easily.

****

**3.1.2 OBJECTIVES**

In wireless sensor networks (WSNs), due to the limitation of nodes’ energy, energy efficiency is an important factor should be considered when the protocols are designing. As a typical representative of hierarchical routing protocols, LEACH Protocol plays an important role. In response to the uneven energy distribution that is caused by the randomness of cluster heads forming. LEACH protocol which is intended to balance the energy consumption of the entire network and extend the life time of the network.

**3.1.3 OUTPUT DESIGN**

The life time remains almost constant as the number of nodes increases. Life time decreases because MANET have to cover more nodes as the number of nodes in the network size increases. We observe that the improvement achieved through LEACH protocol. Energy is uniformly drained from all the nodes and hence the network lifetime is significantly increased. Maintained the energy efficient forms at the output tcl file.

**3.2 MODULARIZATION**

1**.** Tools Installation

2. Network Design

3. Coding

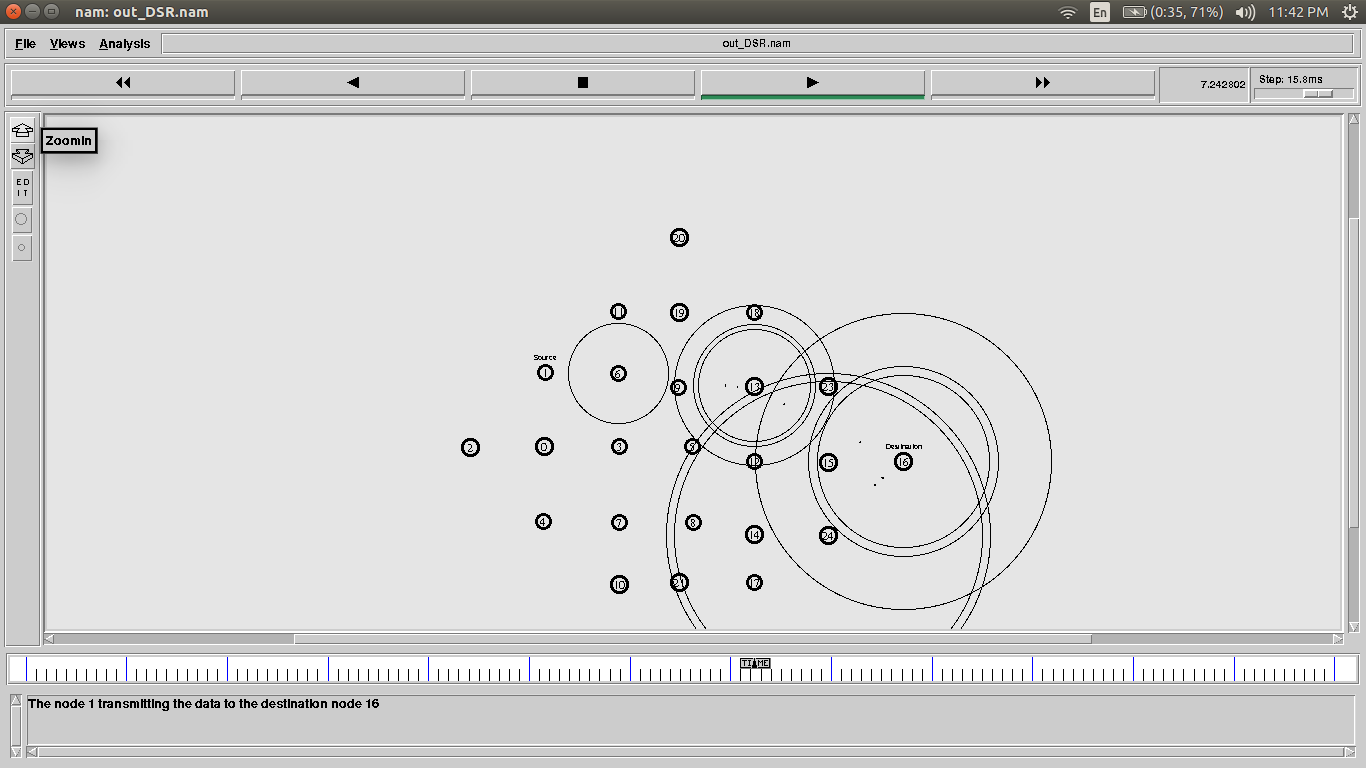
4. Performance Analysis.

**1. Tools Installation:**

* **Tracegraph**: Tracegraph is a third party software helps in plotting the graphs for NS2 and other networking simulation softwares.
* **Network Simulator (NS 2.35):** Ns is a discrete event simulator targeted at networking research. Ns provides substantial support for simulation of TCP, routing, and multicast protocols over wired and wireless (local and satellite) networks.
* **Network Animator (NAM):** The Network Animator (nam) is a completely separate program that is distributed with the NS simulator. This program is named nam and it shows the progression of the packets through the network. The nam program reads an input file (containing the packet transmission events) and draw the network events graphically.

**2. Network Design:**

The sensor nodes are randomly distributed in a sensing field. We are using mobile ad hoc network (MANET). This is the infra-structure-less network and a node can move independently. In a MANET, each node not only works as a host and also acts as a router. We can find the communication range for all nodes. Every node communicates only within the range. If suppose any node out of the range, node will not communicate those nodes.

****

**3. Command syntaxes:**

**Writing and Executing a TCL Script with NS-2**

i) Define simulator.

Creating a simulator is essential for any NS-2 simulation. It is done using the following simple codes.

set simulator\_name [new Simulator]

ii) Define topology.

In this step, define the area (in terms of size) under which the simulation occurs. WE must define it keeping in view the possible locations of our nodes.

set topography\_name [new Topography]

$topography\_name load\_flatgrid size\_x size\_y

iii) Define output trace files.

Trace files record events that occur during any simulation like node creation and data transfer, among others. There are several trace formats and choosing one among them would depend on the results we are looking for and what the files log. A trace file is defined as follows.

set handler [open filename w]

$simulator\_name trace\_type $handler

iv) Define the General Operations Director (“GOD”).

Quoted from CMU document, "GOD (General Operations Director) is the object that is used to store global information about the state of the environment, network or nodes that an omniscient observer would have, but that should not be made known to any participant in the simulation.

create-god number\_of\_nodes

v) Define access point/base station and configure the AP/BS.

To create and configure a base station or an access point in NS-2, it is preferable to define the options we want to configure the AP/BS with. The configuration is handled as follows.

$simulator\_name ap-config –option\_1 value\_1 \

–option\_2 value\_2 \

…………………........

option\_n value\_n

vi) Define wireless node, configure it and attach to AP/BS.

Creating and configuring a wireless node is very similar to what we needed to do with the AP. To set the configurable options, we use the following code.

$simulator\_name node-config –option\_1 value\_1 \

–option\_2 value\_2 \

…………………........

–option\_n value\_n

To create a wireless node (in fact even a wired node):

set node\_name [$simulator\_name node]

Disable random motion and set location as we did earlier.

Finally we “attach” the wireless node to the AP.

$node\_name base-station [AddrParams addr2id [$ap\_name node-addr]]

vii) Define the wired source, configure it and connect it to the AP/BS.

The creation and configuration of the wired source is similar to that of the wireless node, except we do not “attach” it but rather “connect” it with a wired link.

$simulator\_name duplex-link $node\_name $ap\_name bandwidth delay queue\_type

vii) Define the wired source, configure it and connect it to the AP/BS.

The creation and configuration of the wired source is similar to that of the wireless node, except we do not “attach” it but rather “connect” it with a wired link.

$simulator\_name duplex-link $node\_name $ap\_name bandwidth delay queue\_type

viii) Define traffic generator agent and attach it to the source and destination node.

To define the traffic flow, we need to designate the nodes as source or destination and decide the type of application/traffic.

The first step is to define the source agent.

set src\_agent\_name [new Agent/src\_agent\_type]

Then designate which node assumes the agent we defined.

$simulator\_name attach-agent $src\_node\_name $src\_agent\_name

Define the traffic/application type and attach it to source agent.

set $traffic\_name [new Application/Traffic/traffic\_type]

$traffic\_name attach-agent $src\_agent\_name

It should be noted that each traffic type comes with default values of traffic parameters like rate and packet size which can be easily overridden with the set command.

set traffic\_parameter value

Now define the destination agent and attach it.

set dest\_agent\_name [new Agent/dest\_agent\_type].

$simulator\_name attach-agent $dest\_node\_name $dest\_agent\_name

Finally, we can connect the agents.

$simulator\_name connect $src\_agent\_name $dest\_agent\_name

It is necessary to tell the simulator when to start and stop the traffic flow. To do so we schedule events that control the agent‟s activities.

$simulator\_name at start\_time "$traffic\_name start"

$simulator\_name at stop\_time "$traffic\_name stop

ix) Run simulator.

To order the simulator to execute, we simply need to provide the following statement in the script.

$simulator\_name run

It is necessary to clear variables and close files after the simulation ends. It is customary to create a procedure for that.

$simulator\_name at simulation\_stop\_time "proc\_name"

proc proc\_name { } {

global vaiable\_list file\_list

$simulator\_name flush-trace

close $file\_name

exit 0

****

**4. Performance Analysis:**

In this module, we can evaluate the performance of simulation. We are using the x-graph for evaluate the performance. We choose the four evaluation metrics: Packet delivery ratio – it is the ratio of the number of packet received at destination and number of packet sent by the source, Packet loss – the total number of the packet losses, during the data transmission, End-to-End delay – the average time taken for a packet to be transmitted from the source to destination, Throughput – number of data received by the destination without any losses.

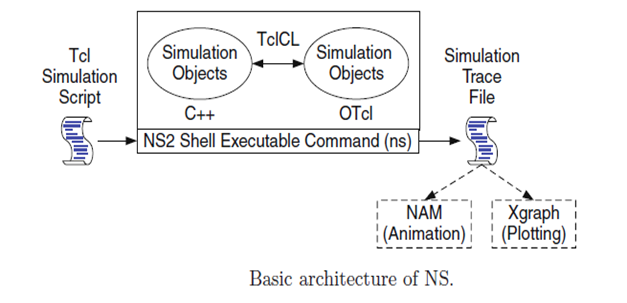
**IMPLEMENTATION**

**Network Simulator 2 (NS2):**

Network Simulator (Version 2), widely known as NS2, is simply an event driven simulation tool that has proved useful in studying the dynamic nature of communication networks. Simulation of wired as well as wireless network functions and protocols (e.g., routing algorithms, TCP, UDP) can be done using NS2. In general, NS2 provides users with a way of specifying such network protocols and simulating their corresponding behaviours. Due to its flexibility and modular nature, NS2 has gained constant popularity in the networking research community since its birth in 1989. Ever since, several revolutions and revisions have marked the growing maturity of the tool, thanks to substantial contributions from the players in the field. Among these are the University of California and Cornell University who developed the REAL network simulator,1 the foundation which NS is based on. Since 1995 the Defense Advanced Research Projects Agency (DARPA) supported development of NS through the Virtual InterNetwork Testbed (VINT) project. Currently the National Science Foundation (NSF) has joined the ride in development. Last but not the least, the group of researchers and developers in the community are constantly working to keep NS2 strong and versatile.

**Basic Architecture**

Figure 2.1 shows the basic architecture of NS2. NS2 provides users with executable command ns which takes on input argument, the name of a Tcl simulation scripting file. Users are feeding the name of a Tcl simulation script (which sets up a simulation) as an input argument of an NS2 executable command ns. In most cases, a simulation trace file is created, and is used to plot graph and/or to create animation. NS2 consists of two key languages: C++ and Object-oriented Tool Command Language (OTcl). While the C++ defines the internal mechanism (i.e., a backend) of the simulation objects, the OTcl sets up simulation by assembling and configuring the objects as well as scheduling discrete events (i.e., a frontend). The C++ and the OTcl are linked together using TclCL. Mapped to a C++ object, variables in the OTcl domains are sometimes referred to as handles. Conceptually, a handle (e.g., n as a Node handle) is just a string (e.g.,\_o10) in the OTcl domain, and does not contain any functionality. Instead, the functionality (e.g., receiving a packet) is defined in the mapped C++ object (e.g., of class Connector). In the OTcl domain, a handle acts as a frontend which interacts with users and other OTcl objects. It may defines its own procedures and variables to facilitate the interaction. Note that the member procedures and variables in the OTcl domain are called instance procedures (instprocs) and instance variables (instvars), respectively. Before proceeding further, the readers are encouraged to learn C++ and OTcl languages.



NS2 provides a large number of built-in C++ objects. It is advisable to use these C++ objects to set up a simulation using a Tcl simulation script. However, advance users may find these objects insufficient. They need to develop their own C++ objects, and use a OTcl configuration interface to put together these objects.

**After simulation**, NS2 outputs either text-based or animation-based simulation results. To interpret these results graphically and interactively, tools such as NAM (Network AniMator) and XGraph are used. To analyze a particular behavior of the network, users can extract a relevant subset of text-based data and transform it to a more conceivable presentation.

**Installation**

NS2 is a free simulation tool, which can be obtained from [9]. It runs on various platforms including UNIX (or Linux), Windows, and Mac systems. NS2 source codes are distributed in two forms: the all-in-one suite and the component-wise. With the all-in-one package, users get all the required components along with some optional components. This is basically a recommended choice for the beginners. This package provides an “install” script which configures the NS2 environment and creates NS2 executable file using the “make” utility.

The current all-in-one suite consists of the following main components:

* NS release 2.30,
* Tcl/Tk release 8.4.13,
* OTcl release 1.12, and
* TclCL release 1.18.

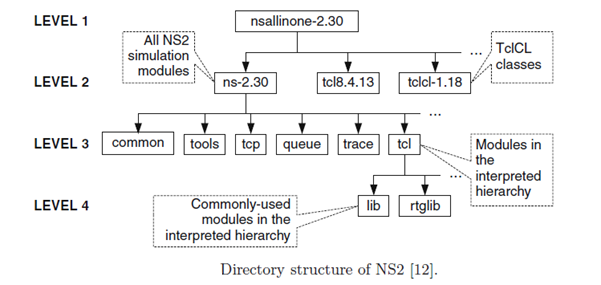
and the following are the optional components:

* NAM release 1.12: NAM is an animation tool for viewing network simulation traces and packet traces.
* Zlib version 1.2.3: This is the required library for NAM.
* Xgraph version 12.1: This is a data plotter with interactive buttons for panning, zooming, printing, and selecting display options.

**Directories**

Suppose that NS2 is installed in directory nsallinone-2.30. Figure 2.2 shows the directory structure under directory nsallinone-2.30. Here, directory nsallinone-2.30 is on the Level 1. On the Level 2, directory tclcl-1.18 contains classes in TclCL (e.g., Tcl, TclObject, TclClass).

All NS2 simulation modules are in directory ns-2.30 on the Level 2. Hereafter, we will refer to directories ns-2.30 and tclcl-1.18 as ˜*ns*/ and ˜*tclcl* /, respectively.



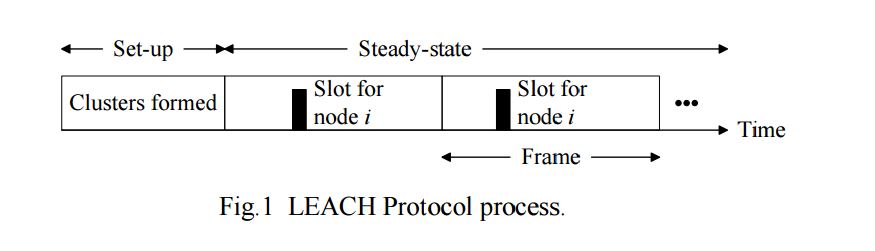
**LEACH Protocol**

LEACH Protocol is a typical representative of hierarchical routing protocols. It is self-adaptive and self-organized LEACH protocol uses round as unit, each round is made up of cluster set-up stage and steady-state stage, for the purpose of reducing unnecessary energy costs, the steady state stage must be much longer than the set-up stage. LEACH is a hierarchical protocol in which most nodes transmit to cluster heads, and the cluster heads aggregate and compress the data and forward it to the base station (sink). Each node uses a stochastic [algorithm](https://en.wikipedia.org/wiki/Algorithm) at each round to determine whether it will become a cluster head in this round. LEACH assumes that each node has a radio powerful enough to directly reach the base station or the nearest cluster head, but that using this radio at full power all the time would waste energy.

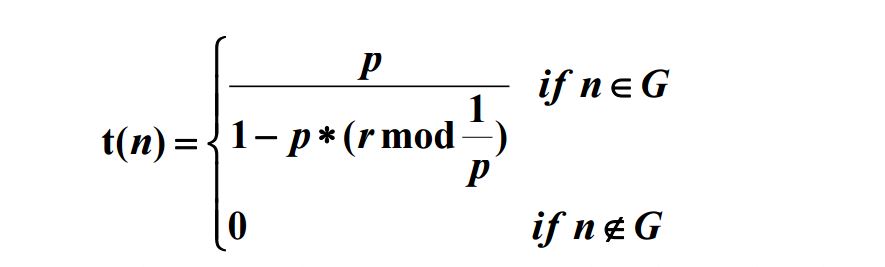
Nodes that have been cluster heads cannot become cluster heads again for *P* rounds, where *P* is the desired percentage of cluster heads. Thereafter, each node has a 1/*P*probability of becoming a cluster head again. At the end of each round, each node that is not a cluster head selects the closest cluster head and joins that cluster. The cluster head then creates a schedule for each node in its cluster to transmit its data.

All nodes that are not cluster heads only communicate with the cluster head in a TDMA fashion, according to the schedule created by the cluster head. They do so using the minimum energy needed to reach the cluster head, and only need to keep their radios on during their time slot.

LEACH also uses [CDMA](https://en.wikipedia.org/wiki/Code_division_multiple_access) so that each cluster uses a different set of CDMA codes, to minimize interference between clusters.



At the stage of cluster forming, a node randomly picks a number between 0 to 1, compared this number to the threshold values nt )( , if the number is less than nt )( , then it become cluster head in this round, else it become common node. Threshold nt )( is determined by the following:



Where p is the percentage of the cluster head no des in all nodes, r is the number of the round, G is the collections of the nodes that have not yet been head nodes in the first 1/P rounds. Using this threshold, al l nodes will be able to be head nodes after 1/P round s. The analysis is as follows: Each node becomes a cluster head with probability p when the round begins, the nodes which have been head nodes in this round will not be head nodes in the next 1/P rounds, because the number of the nodes which is capable of head n ode will gradually reduce, so, for these remain nodes, the probability of being head nodes must be increase d. After 1/P-1 round, all nodes which have not been head nodes will be selected as head nodes with probability 1, when 1/P rounds finished, all nodes will return to the same starting line. When clusters have formed, the nodes start to transmit the inspection data. Cluster heads receive data sent from the other nodes, the received data was sent to the gateway after fused. This is a frame data transmission. In order to reduce unnecessary energy cost, steady stage is composed of multiple frames and the steady stage is much longer than the set-up stage.

* Use 100's - 1000's of nodes
* Maximize system lifetime
* Maximize network coverage
* Use uniform, battery-operated nodes

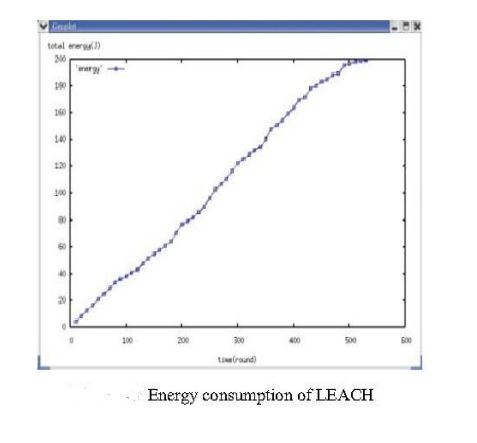
**The Network lifetime:**

The network lifetime in this article is defined as the time from the beginning of the simulation to the time when the last node died. In WSN, the network life is divided into stable and unstable period. Stable period usually means the time from the beginning of the simulation to the time when the first node dies, the unstable period refers to the time from the death of first node to the end of simulation. If it happened that some nodes begin to die, the network operation may become unstable and unreliable data transferring will occur. Therefore, the longer the stable period is, the better the performance of the network. In LEACH Protocol, cluster heads are responsible not only for communicating with the base station, but for the data fusing. Randomly distributing the nodes and randomly selecting the cluster heads causes some cluster heads die earlier because of the low energy or the long distance to base station. Secondary cluster heads are set for these clusters to be responsible for the communication with common nodes and data fusing, this balances the energy load of cluster heads and avoids premature death of these cluster heads, so the stable period of network lifetime will be prolonged.

**Performance analysis of LEACH:**

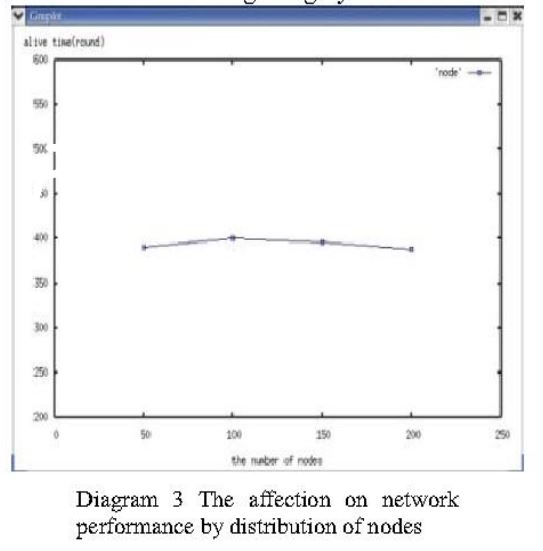
There is not a unique standard to evaluate different routing protocols for WSN now. In practice, the parameters are used to determine the effect of LEACH as follow: the number of survived nodes, which shows the total number of living nodes going with time. It also denotes if the routing protocol is energy-effective and the length of network lifetime. So the number of round when the first dead nodes appear is considered as the key parameter of network lifetime in simulation. The working time of network is defined as the time from the start of nodes' activity to the first dead node's appearance. For convenience, in the simulation model, assume that 100 nodes are distributed randomly in an area of 100x100 m2; the sink node is located at (50,175); it lasts 3600 seconds in total; the starting energy of each nodes is 2J; the percentage of cluster head in all nodes is 0.05.The changing of performance in different nodes density will be discussed with LEACH protocol.

**Energy consumption in LEACH:**

For 100 node's network assumption, the total energy is 200J. For LEACH, when the first dead node appears in the 400th round, the total consumption energy is 164.295587J and the energy is used up until the 530th round. For traditional Flooding protocol, the energy is used up before the 200th round. It proves that LEACH is a energy-effective protocol which could save energy consumption and improve the performance of network.

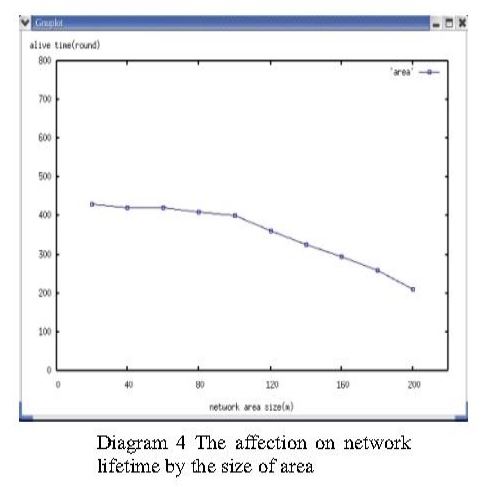
**The affection on network performance by distribution of nodes:**

Assume that all the nodes are of the same type. The key point is still the network lifetime, which is the number of round from the beginning of network to the appearance of first dead node. It indicates the energy consumption of the whole network and the integrative performance of network load balance. The change of nodes density could be easily seen in the simulation. First, keep the area (100x100) of distribution of nodes and change the number N of nodes in the area. Second, distribute the nodes randomly into an MxM area (M is the length of the side-line of the area). Thus the density of WSN nodes can be changed with the length of side-line and keeping the number of nodes. Set the number of nodes by 50,100,150,200, respectively, while the network area is still 100x100.And then obtain the network lifetime in these four situations(from the beginning of network to the appearance of first dead node).



From the result of diagram 3, when the network coverage is fixed, the affection on network lifetime by changing nodes density because of changing the nodes amount is not obviously. The network lifetime represents the consumption of nodes in the simulation model. Because the distribution area of nodes is fixed and the average distance between nodes and or sink node nearly doesn't change, of course, the load of cluster head won't change largely. In this simulation model, the performance of whole network is kept in steady status and the scale of the network could be enlarged easily. Besides changing the number of nodes, the network coverage can be changed by the nodes density. The area is changed into MxM coverage where side-length is variable from the fixed area in this simulation. The value of M could be selected from 20 to 200, while the number of nodes and the proportion of cluster head are fixed, so the scale of nodes and cluster is unchangeable.

**The affection on network lifetime by the size of area:**

****

The result of simulation is shown in diagram 4. When the length of side-line of network coverage is short, the lifetime is of stability. For example, the rounds when the first dead node appears are rather close to each other if M is 20 to 100. However, if the network coverage enlarges to a great extent, the network lifetime would be shortened obviously, that is, the performance of the WSN declines by the reducing of nodes density. That also proves why LEACH is not fit for constructing large scale network. In conclusion, LEACH is much better than traditional protocols on network performance. But the performance declines when constructing large scale network. So the LEACH is not fit for large scale network. For better performance and wider applications, many researches on improvement of LEACH are needed and some good methods have been presented.

**4.1 CODING**

The file leach\_setup.sh consists of steps of what is to be commands include change directory and copy from one location to another location.

**Leach\_setup.sh**

#!/bin/bash

tar -xvzf ns-leach-2.35.tar.gz

cd ns-leach-2.35/

cp -r mit /home/seva/ns-allinone-2.35/ns-2.35

cp apps/app.\* /home/seva/ns-allinone-2.35/ns-2.35/apps

cp mac/channel.cc /home/seva/ns-allinone-2.35/ns-2.35/mac

cp mac/ll.h /home/seva/ns-allinone-2.35/ns-2.35/mac

cp mac/wireless-phy.\* /home/seva/ns-allinone-2.35/ns-2.35/mac

cp mac/phy.\* /home/seva/ns-allinone-2.35/ns-2.35/mac

cp mac/mac.cc /home/seva/ns-allinone-2.35/ns-2.35/mac

cp mac/mac-sensor\* /home/seva/ns-allinone-2.35/ns-2.35/mac

cp trace/cmu-trace.\* /home/seva/ns-allinone-2.35/ns-2.35/trace

cp common/packet.\* /home/seva/ns-allinone-2.35/ns-2.35/common

cp common/mobilenode.cc /home/seva/ns-allinone-2.35/ns-2.35/common

cp tcl/mobility/leach-c.tcl /home/seva/ns-allinone-2.35/ns-2.35/tcl/mobility

cp tcl/mobility/leach.tcl /home/seva/ns-allinone-2.35/ns-2.35/tcl/mobility

cp tcl/mobility/mte.tcl /home/seva/ns-allinone-2.35/ns-2.35/tcl/mobility

cp tcl/mobility/stat-clus.tcl /home/seva/ns-allinone-2.35/ns-2.35/tcl/mobility

cp tcl/ex/wireless.tcl /home/seva/ns-allinone-2.35/ns-2.35/tcl/ex

cp leach\_test /home/seva/ns-allinone-2.35/ns-2.35

cp Makefile.in /home/seva/ns-allinone-2.35/ns-2.35

The file leach-test is the main file which contains the algorithm and procedure with input files and location of outputs. This is the file to be executed.

**Leach-test**

#!/bin/bash

# This file runs a generic LEACH protocol simulation.

#This is the algorithm that we are going to run.

alg=leach

#dirname, filename =

# The directory and filename that we want our output to be written.

dirname="mit/leach\_sims"

filename=$alg

#Topology

#This file is the scenario that we are going to run.

# This file can be editted manually if you are very careful to create

# a predefined topology. To generate a random topology go to the

# ./mit/uAMPS/sims directory and run 'ns genscen'.

topology\_file="mit/uAMPS/sims/100nodes.txt"

#number of clusters we want. It is recommended to use 5% of the total

# number of nodes in the scenario.

num\_clusters=5

# energy values. How much energy does each node have initially

eq\_energy=1

init\_energy=2

# stop is the time to stop the simulation if it is still running

stop=3600

# x,y is the size of the field

x=1000

y=1000

# bs\_x, bs\_y is the location of the base station in the field.

bs\_x=50

bs\_y=175

# Number of nodes. WARNING! This should be 1 higher then the number

# of nodes generated in the scenario.

nn=101

ns tcl/ex/wireless.tcl \

-sc /home/seva/ns-allinone-2.35/ns-2.35/mit/uAMPS/sims/nodescen \

-rp $alg \

-x $x \

-y $y \

-nn $nn \

-stop $stop \

-eq\_energy $eq\_energy \

-init\_energy $init\_energy \

-filename $filename \

dirname $dirname \

-topo $topology\_file \

-num\_clusters $num\_clusters \

-bs\_x $bs\_x \

-bs\_y $bs\_y 2>$dirname/$filename.err 1>$dirname/$filename.out &

The leach.out file has the resultant output which has the threshold value of each node.

**Leach.out**

Creating sensor nodes...

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22

23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50

51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78

79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 (100 == BS)

Loading scenario file...

Node 0: (65.745973803916002,92.581722416254564)

Node 1: (21.008649990432268,92.380389195112684)

Node 2: (37.201202259027028,40.606367467253641)

Node 3: (71.218022131928251,61.297971318148988)

Node 4: (35.003944130150579,11.288995440718249)

Node 5: (34.146372151629237,98.076752432657756)

Node 6: (75.978135678906995,64.526355389750734)

Node 7: (94.45503554048716,5.7823289678349754)

Node 8: (83.602962402441989,14.98909784247591)

Node 9: (21.767438492629417,45.338745622587737)

Node 10: (8.2976788321033492,59.088131160982016)

Node 11: (94.220422624713009,62.643053551503947)

Node 12: (41.80104012684945,50.081411958710021)

Node 13: (18.290790039250062,13.30818967582108)

Node 14: (70.743881524886874,92.416788773805266)

Node 15: (48.968921345178465,20.661048414493468)

Node 16: (50.240702391714187,95.485097540302704)

Node 17: (18.034359867700545,3.486296443029445)

Node 18: (94.184317995879951,55.832556754272687)

Node 19: (77.781369061107455,71.469810032970187)

Node 20: (93.097224129874832,85.045950806255433)

Node 21: (67.295200735002382,30.438753185066748)

Node 22: (84.124781416787201,85.201271942444734)

Node 23: (77.777536668711122,7.0587910278974055)

Node 24: (37.10080587170124,53.244285682795699)

Node 25: (76.709470747369096,56.074851032381346)

Node 26: (50.021301233219582,8.0098267216281158)

Node 27: (21.157710403743067,97.638755709695985)

Node 28: (14.567212860364101,31.146544139434834)

Node 29: (79.967351481303268,11.276346264070062)

Node 30: (21.551660225517892,18.753410279170335)

Node 31: (88.566562015827074,38.207800005659372)

Node 32: (58.494695116996155,20.340831354419155)

Node 33: (68.352573722765115,1.7065585133184484)

Node 34: (82.128933343165059,40.982698575119812)

Node 35: (96.214952038701142,84.698914449987427)

Node 36: (34.655160938694685,49.28989664152725)

Node 37: (15.292854148565258,26.999674936290681)

Node 38: (83.536654237442022,0.54776868808445001)

Node 39: (6.3483406353501328,96.561058329679554)

Node 40: (1.7073469244443564,95.37975913629856)

Node 41: (47.611803769884546,11.585960449458081)

Node 42: (25.23727404197551,62.864823482402052)

Node 43: (69.088268731296182,66.532566895025113)

Node 44: (12.85180468710689,0.28137620551575732)

Node 45: (29.089886103332923,13.7157387164029)

Node 46: (20.420606583552718,9.134849770523072)

Node 47: (29.42009318127301,63.506097655513372)

Node 48: (46.98329621319813,48.259455220894637)

Node 49: (96.663897576119709,30.126561843849046)

Node 50: (37.124909570964476,58.355159199962003)

Node 51: (75.160673761349486,25.443907000796827)

Node 52: (35.744962392256113,65.582926648474725)

Node 53: (52.248180914785806,35.176634804893581)

Node 54: (13.701165846409818,75.494380609828227)

Node 55: (34.054909382972362,60.861999616428278)

Node 56: (7.6275533100718409,96.288482377439962)

Node 57: (20.523317633440399,35.399465232807891)

Node 58: (58.812167802272441,56.104252792943385)

Node 59: (44.176690999500771,77.645628609529524)

Node 60: (90.080040362700842,75.238375912997114)

Node 61: (31.383969742517902,70.379462498416885)

Node 62: (67.626210892398944,93.726468548982623)

Node 63: (60.756902750933961,41.264534947119898)

Node 64: (33.038856244198442,84.056896243270913)

Node 65: (44.255160654082502,96.485113164635891)

Node 66: (25.296958035462051,65.973702010686381)

Node 67: (20.00969360582982,2.9204331817666223)

Node 68: (83.720485951621313,90.207388899385649)

Node 69: (15.585231974527813,40.993795888961202)

Node 70: (82.727505770850698,1.1894906876559792)

Node 71: (91.769987434041681,78.17880393852424)

Node 72: (51.15779477691175,9.0568155558113084)

Node 73: (17.899046520655531,29.27487265750527)

Node 74: (22.784754691079613,43.372092975011142)

Node 75: (54.766631012208123,62.767422181911492)

Node 76: (32.064611386537841,9.9235735414193815)

Node 77: (85.500510635553155,7.082251742054825)

Node 78: (31.405028715452659,24.317620612828815)

Node 79: (6.2496398139044826,37.69635229264216)

Node 80: (62.592982436806423,0.25581540551773058)

Node 81: (99.489520536497949,20.371656920933937)

Node 82: (86.437870136666049,61.283386946322118)

Node 83: (89.884406835718266,87.225687916961363)

Node 84: (2.1368203694637962,13.539949578018836)

Node 85: (65.932557762569076,28.498315498464887)

Node 86: (71.188582699368055,66.509428278780277)

Node 87: (23.961081460100171,13.896099903572399)

Node 88: (51.751079341280779,80.390488906013999)

Node 89: (22.947043377415763,70.958044226727466)

Node 90: (91.849318608571465,11.497854260493934)

Node 91: (44.436556121537727,45.198734684474182)

Node 92: (55.133841957493615,34.481779595130021)

Node 93: (35.269655350255618,77.097471746195794)

Node 94: (77.207638312693518,28.777121439938025)

Node 95: (57.080041038375363,44.249731974792546)

Node 96: (5.2453003382521217,57.762785003410087)

Node 97: (19.127552313323857,76.771730034040161)

Node 98: (2.466682112993059,57.526273074339272)

Node 99: (44.071560420129238,10.71598111219517)

Max Distance for this Simulation is 139.0

Load complete...

Starting Simulation...

THRESH = 0.050000000000000003

0:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

1:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

2:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

2: Is a cluster head at time 0

THRESH = 0.050000000000000003

3:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

4:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

5: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

6: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

7: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

8: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

9: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

10: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

11: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

12: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

13: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

14: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

15: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

16: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

17: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

18: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

19: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

20: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

21: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

22: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

23: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

24: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

25: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

26: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

27: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

28: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

29: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

30: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

31: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

32: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

33: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

34: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

35: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

36: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

37: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

38: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

39: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

40: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

41: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

42: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

42: Is a cluster head at time 0

THRESH = 0.050000000000000003

43:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

44:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

45:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

46:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

47:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

48:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

49:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

50:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

51:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

52:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

53:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

54:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

55:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

56:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

57:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

58:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

59:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

60:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

61:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

62:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

63:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

64:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

65:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

66:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

66: Is a cluster head at time 0

THRESH = 0.050000000000000003

67: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

68: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

69: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

70: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

71: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

72: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

73: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

74: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

75: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

76: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

77: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

78: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

79: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

80: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

81: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

82: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

83: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

84: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

84: Is a cluster head at time 0

THRESH = 0.050000000000000003

85:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

86:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

87:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

88:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

89:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

90:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

91:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

92:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

93:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

94:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

95:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

96:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

97:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

97: Is a cluster head at time 0

THRESH = 0.050000000000000003

98: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THRESH = 0.050000000000000003

99: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

If any errors occurs they are written in leach.err

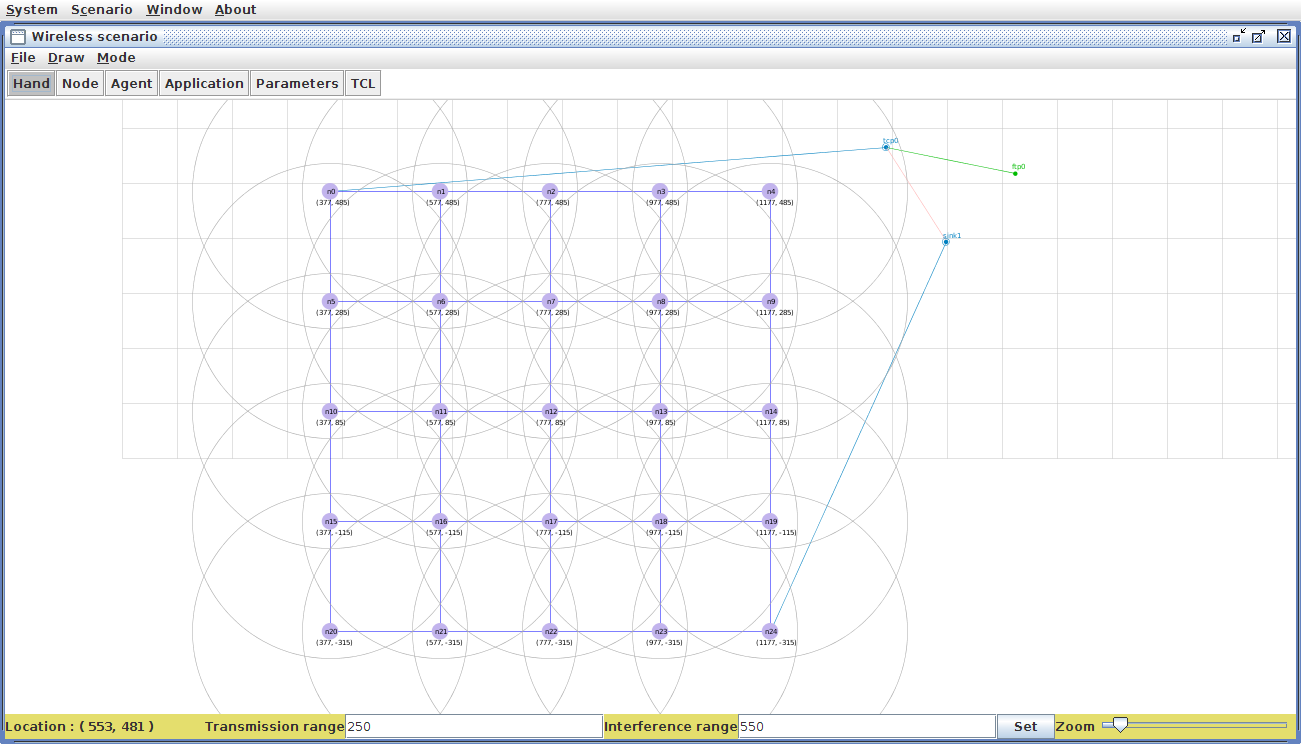
**Leach.err**

INITIALIZE THE LIST xListHead

SORTING LISTS ...DONE!

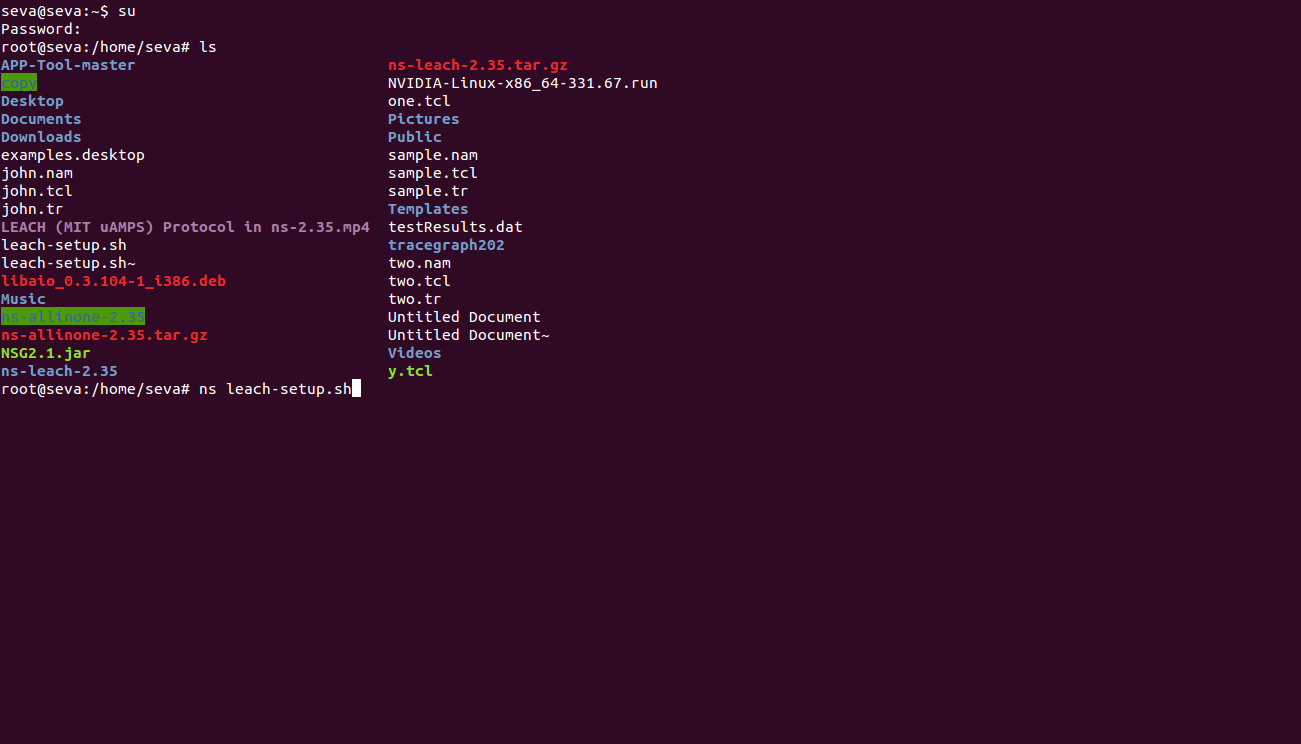
**RESULTS AND DISCUSSION**

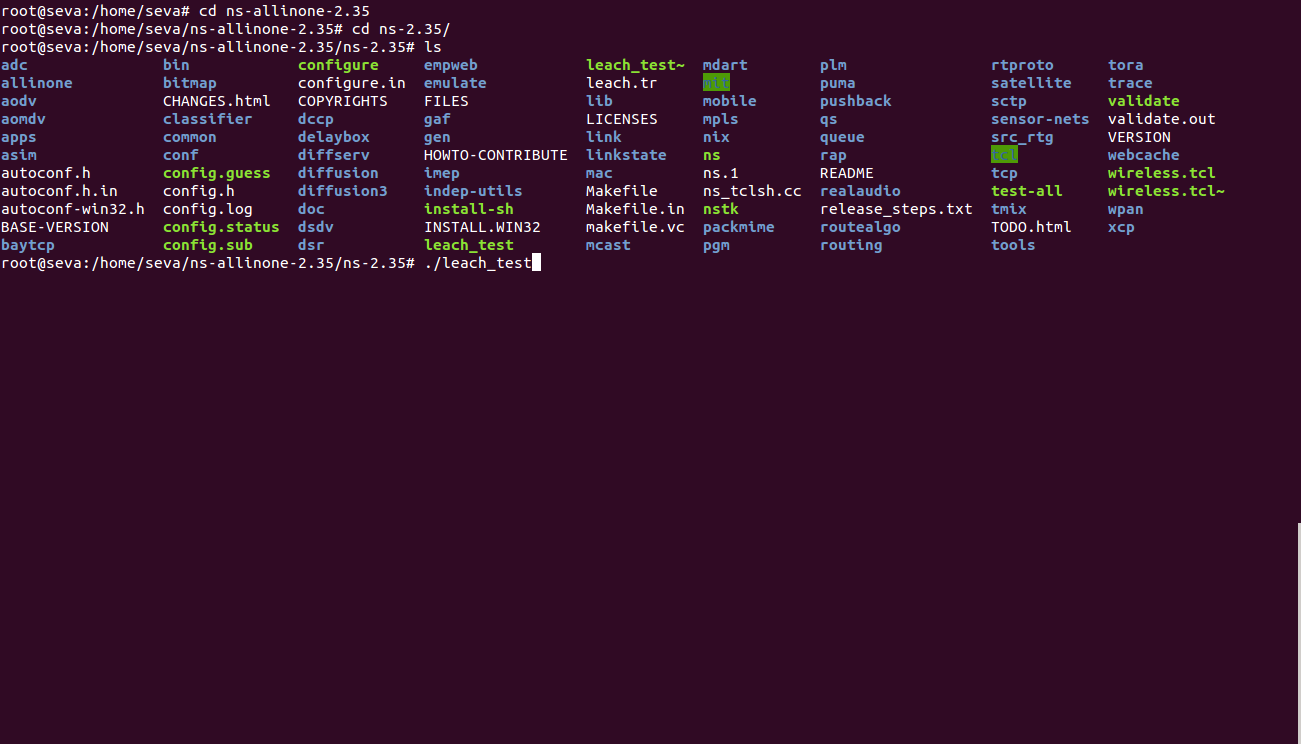
This is the actual network area representation. This is done using Network simulator generator application (NSG). This figure represents a 5X5 node grid.

****

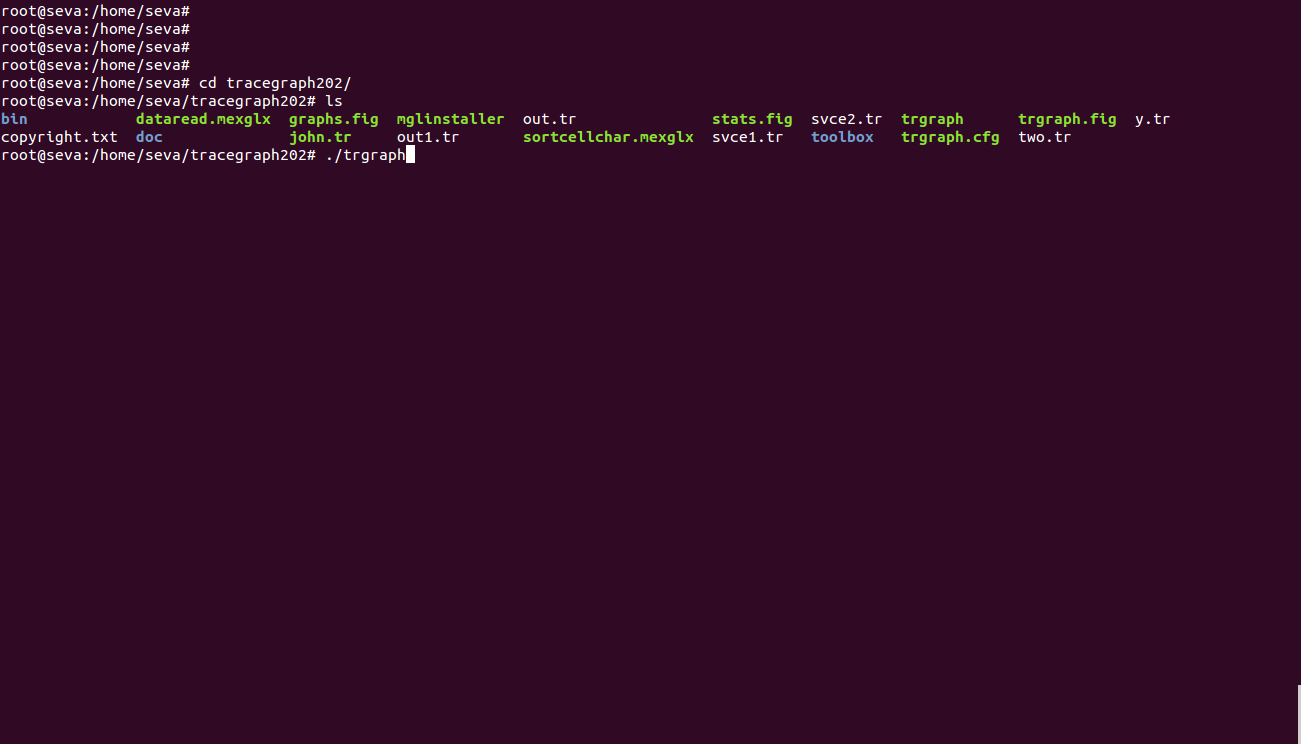
**Fig: NSG**

The screenshot describes the location of leach-setup.sh. Any file with extension “.sh” is named as shell script in which we give list of commands to be performed. The shell script is executed using the command

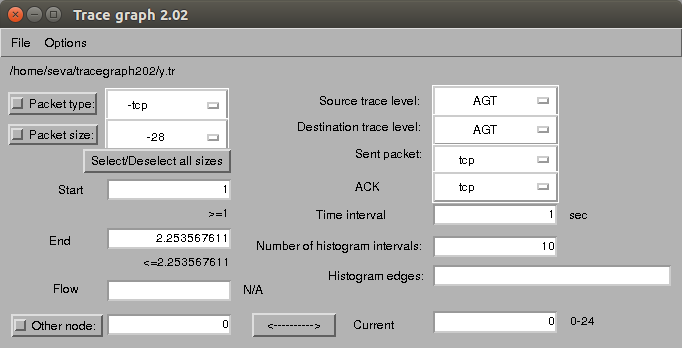
****“ns leach-setup.sh”

As mentioned before the leach\_test is the main file to be executed. Inorder to execute that filee we redirect to location of that file and use the command“ ./leach\_test”

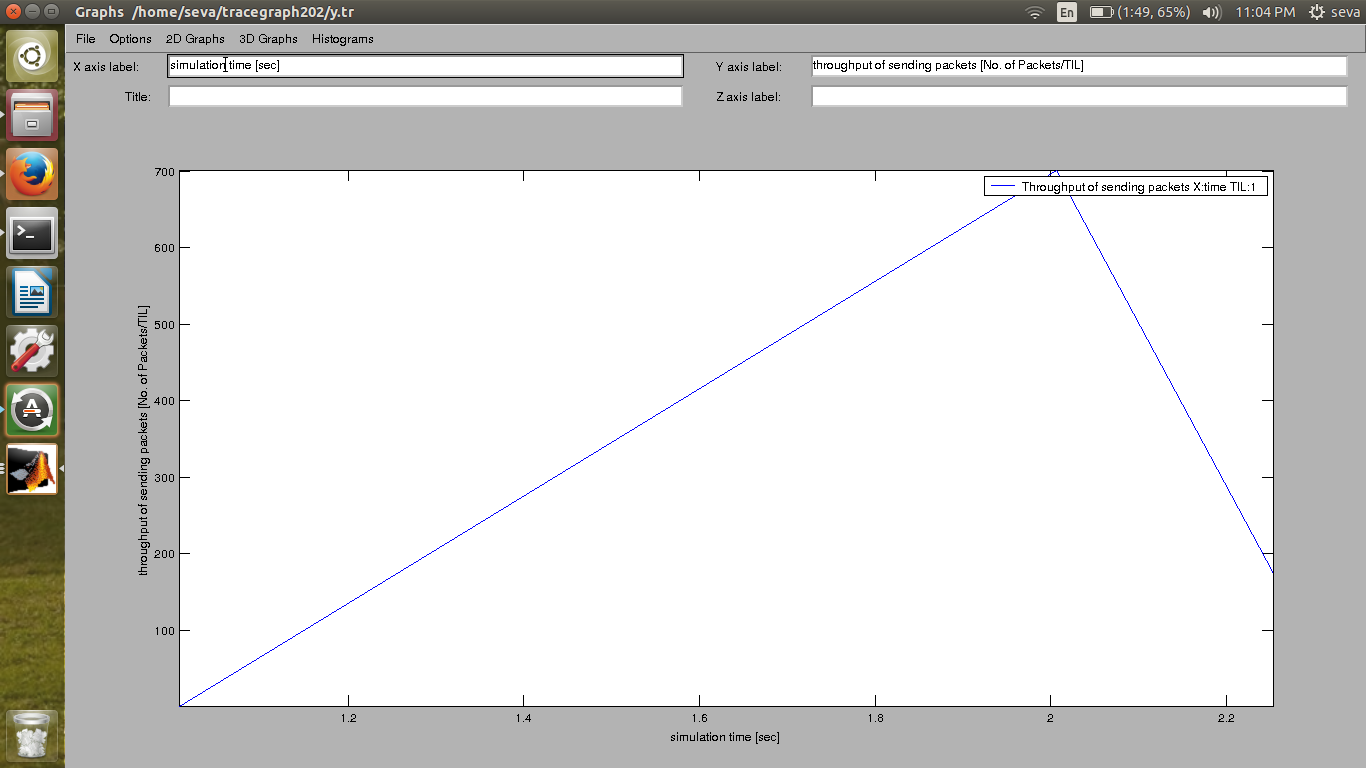
The trace geph is an application used to trace the data from network area and represent the components like e2edelay, throughput, number of packets dropped etc.. pictorially. Inorder to run this application we redirect to the location of the file and use the following command in the screenshot.

****

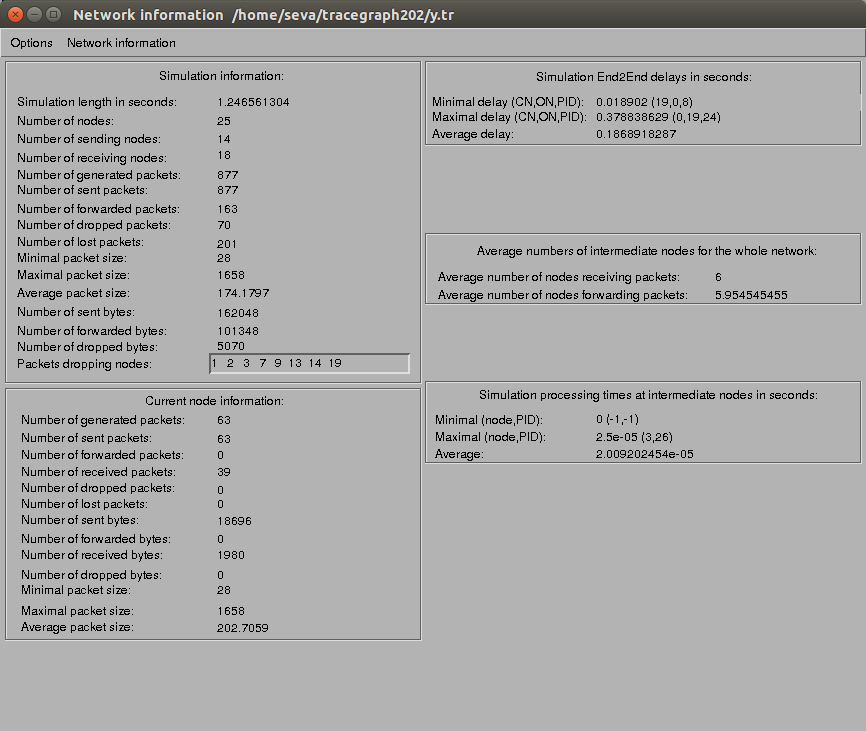
This is how the application looks like. We load a trace file into the application and results are visible

****

This is the kind of the graph that is shown by the application upon selecting the parameters.

****

This is the network information of the entire network area. Every single detail can be viewed using this part of the trace graph application

****

**CONCLUSION**

This research mainly deals with the problem of maximizing the network lifetime of a MANET, i.e. the time period during which the network is fully working. We presented an original solution called LEACH which is basically an improvement on DSR. This study has evaluated three power aware ad-hoc routing protocols in different network environment taking into consideration network lifetime and packet delivery ratio. Overall, the show that the energy consumption and throughput in small size networks did not reveal any significant differences. However, for medium and large ad-hoc networks the DSR performance proved to be inefficient in this study. In particular, the performance of LEACH and DSR in small size networks was comparable. But in medium and large size networks, the LEACH produced good results and the performance of LEACH in terms of throughput is good in all the scenarios that have been investigated. From the various graphs, we can successfully prove that our proposed algorithm quite outperforms the traditional energy efficient algorithms in an obvious way. The LEACH algorithm outperforms the original DSR algorithm by 65%.

**FUTURE ENHANCEMENT**

In contrast to conventional power aware algorithms, LEACH identifies the capacity of a node not just by its residual battery power, but also by the expected energy spent in reliably forwarding data packets over a specific link. Using a min-max formulation, LEACH selects the path that has the largest packet capacity at the smallest residual packet transmission capacity. This protocol must be able to handle high mobility of the nodes that often cause changes in the network topology. This paper evaluates three ad hoc network routing protocols (LEACH, AODV, and DSR) in different network scales, taking into consideration the power consumption. Indeed, our proposed scheme reduces for more than 20% the total energy consumption and decreases the mean delay, especially for high load networks, while achieving a good packet delivery ratio.

**REFERENCES**

[1] V. Rishiwal, S. Verma, and S. K. Bajpai, ``QoS based power aware routing in MANETs,'' *Int. J. Comput. Theory Eng.*, vol. 1, no. 1,pp. 47\_54, 2009.

[2] C. Huang, ``On-demand location-aided QoS routing in ad hoc networks,'' in *Proc. IEEE Int. Conf. Parallel Process.*, Montreal, QC, Canada,Aug. 2004, pp. 502\_509.

[3] W.-H. Lio, Y.-C. Tseng, and K.-P. Shih, ``A TDMA-based bandwidth reservation protocol for QoS outing in a wireless mobile adhoc network,'' in *Proc. IEEE Int. Conf. Commun.*, vol. 5. 2002, pp. 3186\_3190.

[4] J. Punde, N. Pissinou, and K. Makki, ``On quality of service routing in ad-hoc networks,'' in *Proc. 28th Annu. IEEE Conf. Local Area Netw.*, Oct. 2003, pp. 276\_278.

[5] P.-J. Wan, G. Calinescu, X. Li, and O. Frieder, ``Minimum-energy broadcast routing in static ad hoc wireless networks,'' in *Proc. IEEE INFOCOM*, Apr. 2001, pp. 1162\_1171.

[6] S.-L. Wu, Y.-C. Tseng, and J.-P. Sheu, ``Intelligent medium access for mobile ad hoc networks with busy tones and power control,'' *IEEE J. Sel. Areas Commun.*, vol. 18, no. 9, pp. 1647\_1657, Sep. 2000.

[7] S. Muthuramalingam, R. RajaRam, K. Pethaperumal, and V. K. Devi, ``A dynamic clustering algorithm for MANETs by modifying weighted clustering algorithm with mobility prediction,'' *Int. J. Comput. Elect. Eng.*, vol. 2, no. 4, pp. 709\_714, 2010.