New Generation Routing Protocol over Mobile Ad Hoc Wireless Networks based on Neuro-Fuzzy-Genetic Paradigm

Siddesh.G.K.[1] and K.N.Muralidhara[2]

[1] Department of Electronics and Communication Engineering, Sri Bhagawan Mahaveer Jain College of Engineeing, Jain University, Bangalore,India
professorsiddeshgk@gmail.com
[2] Department of Electronics and Communication, P.E.S.College of Engineering, Mandya, India
knm08@rediffmail.com

Abstract:

There is a vast amount of researched literature available on Route Finding and Link Establishment in MANET protocols based on various concepts such as "pro-active", "reactive", "power awareness", "cross-layering" etc. Most of these techniques are rather restrictive, taking into account a few of the several aspects that go into effective route establishment. When we look at practical implementations of MANETs, we have to take into account various factors in totality, not in isolation. The several factors that decide and influence the routing have to be considered as a whole in the difficult task of finding the best solution in route finding and optimization. The inputs to the system are manifold and apparently unrelated. Most of the parameters are imprecise or non-crisp in nature. The uncertainty and imprecision lead to think that intelligent routing techniques are essential and important in evolving robust and dependable solutions to route finding. The obvious method by which this can be achieved is the deployment of soft computing techniques such as Neural Nets, Fuzzy Logic and Genetic algorithms. Neural Networks help us to solve the complex problem of transforming the inputs to outputs without apriori knowledge of what the relationship is between inputs and outputs. Fuzzy Logic helps us to deal with imprecise and ill-conditioned data. Genetic Algorithms help us to select the best possible solution from the solution space in an optimal sense. Our paper presented here below seeks to explore new horizons in this direction. The results of our experimentation have been very satisfactory and we have achieved the goal of optimal route finding to a large extent. There is of course considerable room for further refinements.

Keywords:

Routing Protocol, MANET, Neural Nets, Fuzzy Logic, Genetic Algorithm, Soft Computing, Bayesian Estimator, Squashing Function

1. Introduction:

Wireless Ad Hoc Networks are capable of communication through wireless medium without the need for a pre-existing infrastructure. Wireless Ad Hoc Networks (MANETs for short) are characterized by their mobility, ease of deployment, self-configuration without a centralized administration and ability of nodes to communicate with each other even in out-of-range conditions with intermediate nodes performing the routing functions. MANETs are also flexible enough to get connected to cellular as well as wired networks. The features that delineate them from traditional networks are the mobility of the nodes, the absence of need for an infrastructure/ centralized administration and the ability to configure on the fly as the situation demands. These unique features impose additional overheads in protocol implementations. Compared to Cellular Networks, MANETs are adaptable to changing traffic demands and other physical conditions. Since the attenuation characteristics of wireless media are non-linear, energy efficiency will be superior and increased spatial reuse will guarantee superior capacity and spectral efficiency. These characteristics make Ad Hoc Networks highly attractive for pervasive communications, a fact this is tightly coupled with heterogeneous networks and 4G architectures.

Since MANETs are generally deployed in disaster management and critical situations, there is a substantial amount of real-time content in their operation. Time plays a crucial role in the communication activities, be it a protocol transfer session or a plain routing operation. In view of these facts, efficient protocol implementation assumes the highest level of importance in practical implementations. It therefore no surprise that a huge amount of time and effort has gone into inventing various kinds protocols to suit different needs and varying conditions. The efficiency of a routing protocol (at the outermost level) is directly related to numerous factors such as node mobility, dynamic topology, the communication capabilities of the nodes, power consumption issues, bandwidth constraints, traffic congestion, security and a host of other related parameters, all of which have to be well orchestrated to achieve an optimal performance that is adequate at the minimum level.

When we take all these factors into consideration, evolution of an optimal routing strategy is an indomitable task. These factors are mutually exclusive and there is no explicit relationship of these factors amongst themselves and more importantly we do not see how these are related to an optimal routing strategy. Herein lies a highly complex non-linear problem to solve, a problem that is not amenable to any classical solution. Artificial Neural Network (ANN for short) steps in at this juncture as our savior. An ANN is akin to a biological network, capable of thinking, reasoning, decision making and a high degree of parallelism. It draws inferences from a vast storehouse of knowledge and experience gained over a period of time in solving problems. It can work with imprecise and ill-defined parameters in arriving at solutions. Fuzzy Logic and Genetic Algorithms are additional ingredients that can make an ANN more powerful and aggressive in solving unsolvable problems by analytical methods. Fuzzy Logic helps us to work with ill-defined parameters and Genetic Algorithms represent a powerful paradigm in searching for optimal solutions in a solution space. A judicious admixture of ANN with Fuzzy Logic and Genetic Algorithms personifies a powerful mechanism in protocol development and routing strategies in Ad Hoc Networks. Given this scenario, it will not be out of place to make a brief digression and talk about these ingredients albeit briefly.

1.1. Artificial Neural Networks:

According to a simplified account, the human brain consists of about ten billion neurons and a neuron is, on average, connected to several thousand other neurons. By way of these connections, neurons both send and receive varying quantities of energy. One very important feature of neurons is that they don't react immediately to the reception of energy. Instead, they sum their received energies, and they send their own quantities of energy to other neurons only when this sum has reached a certain critical threshold. The brain learns by adjusting the number and strength of these connections. Even though this picture is a simplification of the biological facts, it is sufficiently powerful to serve as a model for the neural net. The first step toward understanding neural nets is to abstract from the biological neuron, and to focus on its character as a threshold logic unit (TLU). A TLU is an object that inputs an array of weighted quantities, sums them, and if this sum meets or surpasses some threshold, outputs a quantity. Let's label these features. First, there are the inputs and their respective weights: $X_1, X_2, \ldots, X_n$ and $W_1, W_2, \ldots, W_n$. Then, there are the $X_i * W_i$ that are summed, which yields the activation level a, in other words:

$$a = \sum_{k=1}^{n} X_k W x_k$$

The threshold is called theta. Lastly, there is the output: y. When $a >= theta$, $y = 1$, else $y = 0$.

Notice that the output doesn't need to be discontinuous, since it could also be determined by a squashing function, s (or sigma), whose argument is a, and whose value is between 0 and 1. Then, $y = s(a)$.

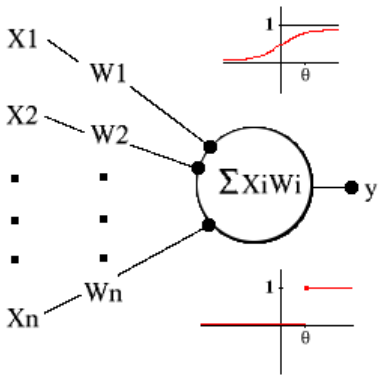


Figure 1 . Threshold logic unit, with sigma function (top) and cutoff function (bottom)

A TLU can classify. Imagine a TLU that has two inputs, whose weights equal 1, and whose theta equals 1.5. When this TLU inputs <0,0>, <0,1>, <1,0>, and <1,1>, it outputs 0, 0, 0, and 1 respectively. This TLU classifies these inputs into two groups: the 1 group and the 0 group.

Insofar as a human brain that knows about logical conjunction (Boolean AND) would similarly classify logically conjoined sentences, this TLU knows something like logical conjunction. This TLU has a geometric interpretation that clarifies what is happening. Its four possible inputs corresponding to four points on a Cartesian graph. From $X_1 * W_1 + X_2 * W_2 = theta$, in

other words, the point at which the TLU switches its classificatory behavior, it follows that $X_2 = -X_1 + 1.5$. The graph of this equation cuts the four possible inputs into two spaces that

correspond to the TLU's classifications. This is an instance of a more general principle about TLUs. In the case of a TLU with an arbitrary number of inputs, N, the set of possible inputs corresponds to a set of points in N-dimensional space. If these points can be cut by a hyperplane -- in other words, an N-dimensional geometric figure corresponding to the line in the above example -- then there is a set of weights and a threshold that define a TLU whose classifications match this cut.

1.1.1. How a TLU learns

It is obvious that **TLU**s can classify. Neural nets are also supposed to learn. Their learning mechanism is modeled on the brain's adjustments of its neural connections. A **TLU** learns by changing its weights and threshold. Actually, the weight-threshold distinction is somewhat arbitrary from a mathematical point of view. The critical point at which a **TLU** outputs 1 instead of 0 is when $\sum(X_i * W_i) >= theta$. This is equivalent to saying that the critical point is when the $\sum(X_i * W_i) + (-1 * theta) >= 0$. So, it is possible to treat -1 as a constant input whose weight, theta, is adjusted in learning, or, to use the technical term, *training*. In this case, $y = 1$ when $\sum(X_i * W_i) + (-1 * theta) >= 0$, else $y = 0$.

During training, a neural net inputs:

1. A series of examples of the items to be classified
2. Their proper classifications or targets

Such input can be viewed as a vector: $< X_1, X_2, \ldots, X_n, theta, t >$, where t is the target or true classification. The neural net uses these to modify its weights, and it aims to match its classifications with the targets in the training set. More precisely, this is supervised training, as opposed to unsupervised training. The former is based on examples accompanied by targets, whereas the latter is based on statistical analysis.

A neural network with a feedback mechanism is called a back-propagation network, where the feedback is used to correct the inferences drawn from the previous cycle. This mechanism is essentially the backbone in the neural network training.

1.2. Fuzzy Logic (FL):

FL was conceived by Prof.Lotfi Zadeh as a simple but powerful methodology in logic building. It was originally conceived in the context of building control systems based on micro-controllers. FL incorporates a simple, rule-based $IF\ X\ AND\ Y\ THEN\ Z$ approach to a solving control

problem rather than attempting to model a system mathematically. The FL model is empirically-based, relying on an operator's experience rather than their technical understanding of the system. For example, rather than dealing with temperature control in terms such as "SP =500F", "T <1000F", or "210C <TEMP <220C", terms like "IF (process is too cool) AND (process is getting colder) THEN (add heat to the process)" or "IF (process is too hot) AND (process is heating rapidly) THEN (cool the process quickly)" are used. These terms are imprecise and yet very descriptive of what must actually happen. Consider what one would do in the shower if the temperature is too cold: one would make the water comfortable very quickly with little trouble. FL is capable of mimicking this type of behavior but at very high rate.

FL requires some numerical parameters in order to operate such as what is considered significant error and significant rate-of-change-of-error, but exact values of these numbers are usually not critical unless very responsive performance is required in which case empirical tuning would determine them. For example, a simple temperature control system could use a single temperature feedback sensor whose data is subtracted from the command signal to compute "error" and then time-differentiated to yield the error slope or rate-of-change-of-error, hereafter called "error-dot". Error might have units of degs F and a small error considered to be 2F while a large error is 5F. The "error-dot" might then have units of degs/min with a small error-dot being 5F/min and a large one being 15F/min. These values don't have to be symmetrical and can be "tweaked" once the system is operating in order to optimize performance. Generally, FL is so forgiving that the system will probably work the first time without any tweaking.

In the context of modeling protocol for an Ad Hoc Network, most of the parameters are imprecise or not so-well defined. For example, mobility can be expressed in vague terms by means of a motion vector (precise values will never be known and not essential either). Similarly, distance limitations, power available at the nodes, traffic density etc. are parameters where determination of precise values are not practical and not important either. A fuzzy model helps us to work with imprecise values in a very predictable way.

1.3. Genetic Algorithm (GA):

The basic purpose of genetic algorithms (GAs) is optimization. Since optimization problems arise frequently, this makes GAs quite useful for a great variety of tasks. As in all optimization problems, we are faced with the problem of maximizing/minimizing an objective function $f(x)$ over a given space X of arbitrary dimension. A brute force which would consist in examining every possible x in X in order to determine the element for which f is optimal is clearly infeasible. GAs give a heuristic way of searching the input space for optimal x that approximates brute force without enumerating all the elements and therefore bypasses performance issues specific to exhaustive search.

We will first select a certain number of inputs, say, $x_1, x_2 \ldots x_n$ belonging to the input space X. In the GA terminology, each input is called an organism or chromosome. The set of chromosomes is designated as a colony or population. Computation is done over epochs. In each epoch the colony will grow and evolve according to specific rules reminiscent of biological evolution.

To each chromosome x_i, we assign a fitness value which is nothing but $f(x_i)$. Stronger individuals, that is those chromosomes with fitness values closer to the colony optimal will have greater chance to survive across epochs and to reproduce than weaker individuals which will tend to perish. In other words, the algorithm will tend to keep inputs that are close to the optimal in the set of inputs being considered (the colony) and discard those that under-perform the rest.

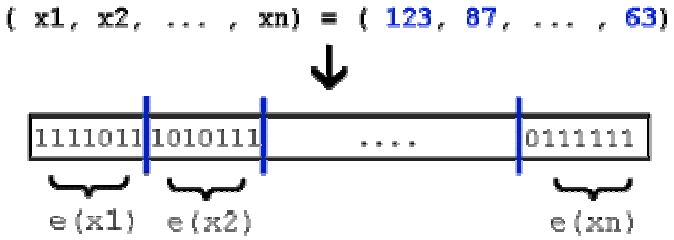
The crucial step in the algorithm is reproduction or breeding that occurs once per epoch. The content of the two chromosomes participating in reproduction are literally merged together to form a new chromosome that we call a child. This heuristic allows us to possibly combine the best of both individuals to yield a better one (evolution).

Moreover during each epoch, a given fraction of the organisms is allowed to mutate. This provides a degree of randomness which allows us to span the whole input space by generating individuals with partly random genes.

Each epoch ends with the deaths of inapt organisms. We eliminate inputs exhibiting bad performance compared to the overall group. This is based on the assumption that they're less inclined to give birth to strong individuals since they have poor quality genes and that therefore we can safely disregard them (selection).

2. The Algorithm:

Let's examine in further detail how this whole process is accomplished and how the algorithm works in practice. Let's take the example of optimizing a function f over a space X contained in Nd. Every input x in X is an integer vector $x = (x_1, x_2, \ldots, x_n)$. For the sake of simplicity, assume $0 <= x_i <= k \; for \; i = 1 \ldots n$. In order to implement our genetic algorithm for optimizing f, we first need to encode each input into a chromosome. We can do it by having $log(k)$ bits per component and directly encoding the value x_i. Each bit will be termed *gene*. Of course, we may choose any other encoding based on our requirements and the problem at hand.



At epoch 0, we generate (possibly randomly) an initial set of inputs in X. Then at each epoch t, we perform fitness evaluation, reproduction, mutation and selection. The algorithm stops when a specified criterion providing an estimate of convergence is reached.

2.1. Neuro-Fuzzy-Genetic Based Network:

Based on the previous discussion of the three essential ingredients, our ANN acts like a powerful inference engine, drawing all the inference rules from an extensive knowledge base. Our hybrid Neural Network functions with the cooperation of Fuzzy Logic, operating on inputs (which are fuzzy in nature) and generating a set of solutions in the solution space with minimal searching using Genetic algorithms. A representative schema of the proposed network would look like:

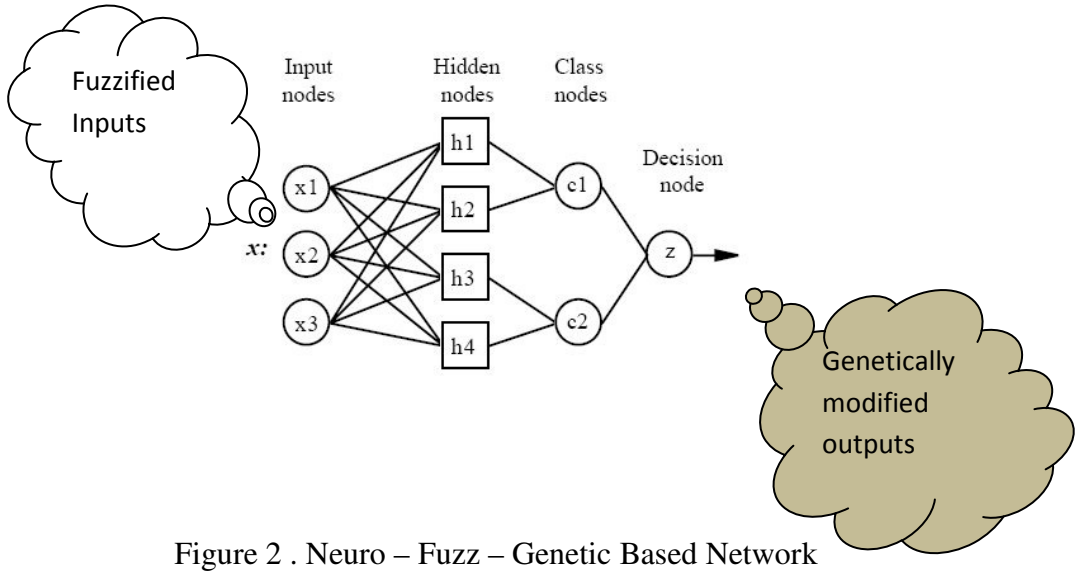


Figure 2 . Neuro – Fuzz – Genetic Based Network

Table 1. Layer Organization of our Neuro-Fuzzy-Genetic Network

	Input Layer	Input Layer (Fuzzified)	Hidden Layer	Output Layer	
1	Network Input Parameters	Fuzzified Inputs based on predefined logic levels	Activation and Processing Functions	Genetically modified results	
2	Number of Nodes in the network				
3	Mobility Measure (Motion Vector)				
4	Communication Constraints				
5	Last used Route Vector				
6	Node active List				
7	Route Map (Historical Data)				
8	Congestion Map (Historical Data)				
9	Link Failure Map				
10	Resident Power in the Nodes (List)				
11	Other relevant parameters				

3. Implementation of the Neuro-Fuzzy-Genetic Network:

Some amount of introduction on the design of our routing protocol will be in order at this point. The design of our protocol is not based on the conventional classifications such as "pro-active", "reactive" and similar attributes. The aim of the protocol is to establish the best possible route within the minimum time possible and the appropriate approach to this problem is the use of soft-computing technologies such as Neural Nets to reduce the dimensionality of the problem, Fuzzy Logic to deal with imprecise inputs and Genetic Algorithms to find an optimum solution in the solution space by search and related heuristic techniques.

The problem therefore reduces to finding an acceptable solution in an optimal sense. There are well established techniques such as Linear Programming to find optimal solutions through various cycles of iteration using a cost/ objective function. However, Linear Programming is not appropriate when the inputs to the system are not crisp. A judicious combination of Neural Network with Fuzzy Logic and Genetic Algorithms appears to be the ideal solution.

By examining the input layers of the Neural Network (there are many of them), it is obvious that the proposed NFG Network has many input layers as opposed to conventional Neural Networks. The input to output relationship depends on various parameters enumerated in the above table and this fact makes the network much more complex than the traditional ones.

Associated with each input parameter, there is a set of connection weights. In our case there are 11 input parameters $(I_x, with\ x = 0\ thru\ 10)$. The associated connection weights are: $W_{xk}\ with\ x = 0\ thru\ 11\ and\ k = 0\ thru\ n-1$. W_x is the set of weights associated with layer I_x and W_{xk} is the k^{th} weight in the set W_x. The weighted sum of the layer I_x is: $\sum_{x=0}^{n-1} I_{xn} * W_{xn}$. We propose a Neural Back Propagation Network with several input layers instead of just one. The input layers to the system are enumerated in the table mentioned above.

Corresponding to each input layer with a set of connection weights, the weighted sum is computed: $\sum_n I_k W_k$. This weighted sum, we will designate as: Q_k, where k is the index corresponding to the set of input layers. Assuming a set of N input layers, we have a set of N weighted sums: Q_k with $k \in 1 \ldots N$. Normal squashing functions used in Neural Networks have either sharp cutoff or exponential cutoff boundaries. Extensive experimentation and empirical studies have shown that the best form of squashing is achieved through the function:

$$Output = sqrt(e^{(x^2+y^2)})$$

Where the x and y values are chosen initially as 0.1 and updated through the training phase. Ultimately we have a set of N squashed values that are further processed in the route finding mechanism.

A Bayesian estimator is pressed into service at this stage to find the optimal solution. This estimator is an essential ingredient of the Fuzzy-Genetic component of the system.

3.1. Bayesian Estimator:

A Bayes estimator derived through the empirical Bayes method is called an *empirical Bayes estimator*. Empirical Bayes methods enable the use of auxiliary empirical data, from observations of related parameters, in the development of a Bayes estimator. This is done under the assumption that the estimated parameters are obtained from a common prior. For example, if independent observations of different parameters are performed, then the estimation performance of a particular parameter can sometimes be improved by using data from other observations.

There are parametric and non-parametric approaches to empirical Bayes estimation. Parametric empirical Bayes is usually preferable since it is more applicable and more accurate on small amounts of data.

The following is an example of parametric empirical Bayes estimation. Given past observations $x_1, \ldots, x_n$ having conditional distribution $f(x_i \mid \theta_i)$, one is interested in estimating θ_{n+1} based on x_{n+1}. Assume that the θ_i's have a common prior π which depends on unknown parameters. For example, suppose that π is normal with unknown mean μ_π and variance σ_π. We can then use the past observations to determine the mean and variance of π in the following way.

First, we estimate the mean μ_m and variance σ_m of the marginal distribution of $x_1, \ldots, x_n$ using the maximum likelihood approach:

$$\widehat{\mu}_m = \frac{1}{n}\sum x_i,$$
$$\widehat{\sigma}_m^2 = \frac{1}{n}\sum (x_i - \widehat{\mu}_m)^2.$$

Next, we use the relation

$$\mu_m = E_\pi[\mu_f(\theta)],$$
$$\sigma_m^2 = E_\pi[\sigma_f^2(\theta)] + E_\pi[\mu_f(\theta) - \mu_m],$$

Where $\mu_f(\theta)$ and $\sigma_f(\theta)$ are the moments of the conditional distribution $f(x_i \mid \theta_i)$, which are assumed to be known. In particular, suppose that $\mu_f(\theta) = \theta$ and that $\sigma_f^2(\theta) = K$;

we then have

$$\mu_\pi = \mu_m,$$
$$\sigma_\pi^2 = \sigma_m^2 - \sigma_f^2 = \sigma_m^2 - K.$$

Finally, we obtain the estimated moments of the prior,

$$\widehat{\mu}_{\pi} = \widehat{\mu}_{m},$$
$$\widehat{\sigma}_{\pi}^{2} = \widehat{\sigma}_{m}^{2} - K.$$

For example, if $x_i \mid \theta_i \tilde{} N(\theta_i,1)$, and if we assume a normal prior (which is a conjugate prior in this case), we conclude that $\theta_{n+1} \sim N(\widehat{\mu}_{\pi}, \widehat{\sigma}_{\pi}^{2})$, from which the Bayes estimator of θ_{n+1} based on x_{n+1} can be calculated.

While the Neural Network gives a number of feasible solutions, the Bayesian Estimator picks up the best possible solution out of the solution space.

4. Results

It is important to observe that the objective function of any protocol is to establish a link as quickly as possible, taking into account the various input constraints. Our goal has been to solve the objective function and establish a route within the shortest possible time. Our implementation surpasses traditional routing algorithms by implicitly taking into account all the network input parameters at the same time in reaching an optimal solution. These include:

1. Varying number of nodes in the network
2. The mobility of the nodes across a geographical region
3. Limitations in the communication capabilities of the nodes
4. Congested and blocked routes
5. Nodes that are currently active
6. Link failure history
7. And other unknown parameters that may subtly influence the routing
8. Facility for peer-to-peer communication across protocol stacks

The results generated from the network are independent of the protocol philosophy that one chooses to follow. The chief merit in our implementation is the independency of our simulator to traditional protocol paradigms such ‘pro-active’, ‘reactive’, ‘power-aware’ and similar such principles.

Table 2. The performance measure rated on a scale of 1 to 10 with 10 being the best and 1 the worst.

Protocol Aspect	Performance of HyperNet	Performance of NS2
Max Number of nodes that simulator can handle	9	6
Peer to peer communication across protocol stacks	9	4
Route finding and link establishment	8	2
Ability to handle dead connections (timeout conditions)	8	3
Detecting nodes going out of range due to mobility	9	3
Dynamic switching of routing strategies	8	1
Hunting facility of nodes based on instantaneous status	9	1

4.1. Trials:

Extensive trials have been conducted with 60, 140 and 250 nodes in the simulated environment and the results show a remarkable superiority in performance as opposed to NS2 simulator. The results have shown here below both for HyperNet and NS2.

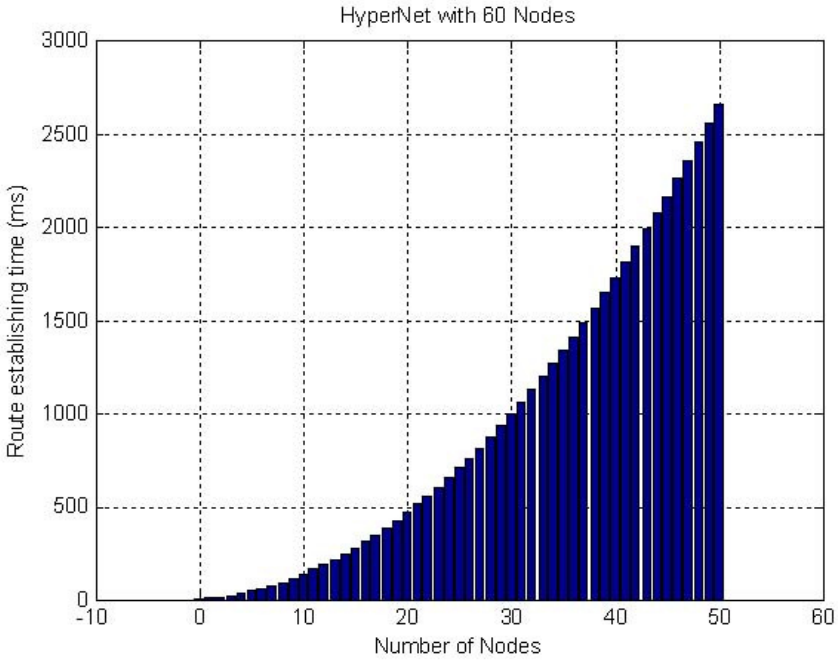


Figure 3. Result of HyperNet with 60 Active Nodes

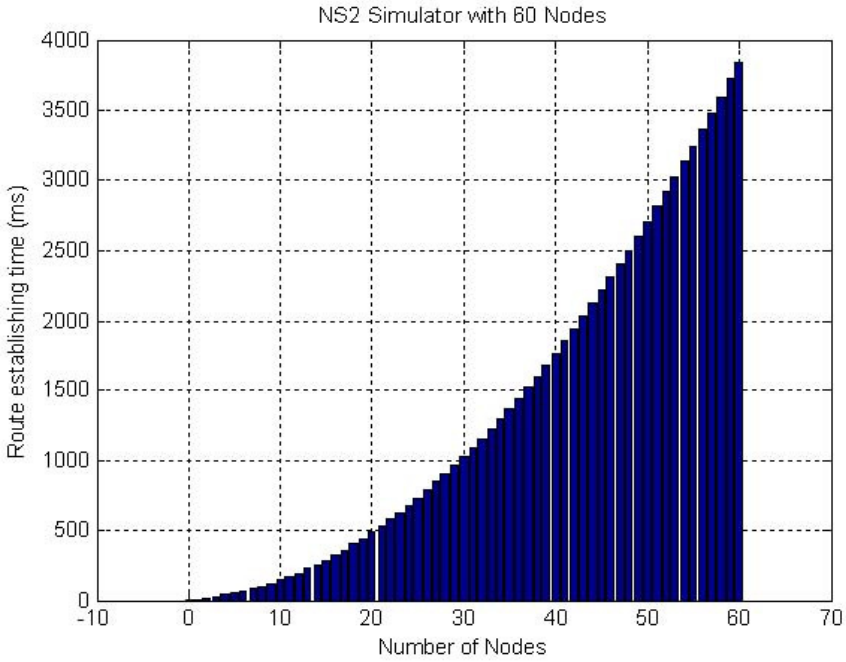


Figure4. Result of NS2 with 60 Active Nodes

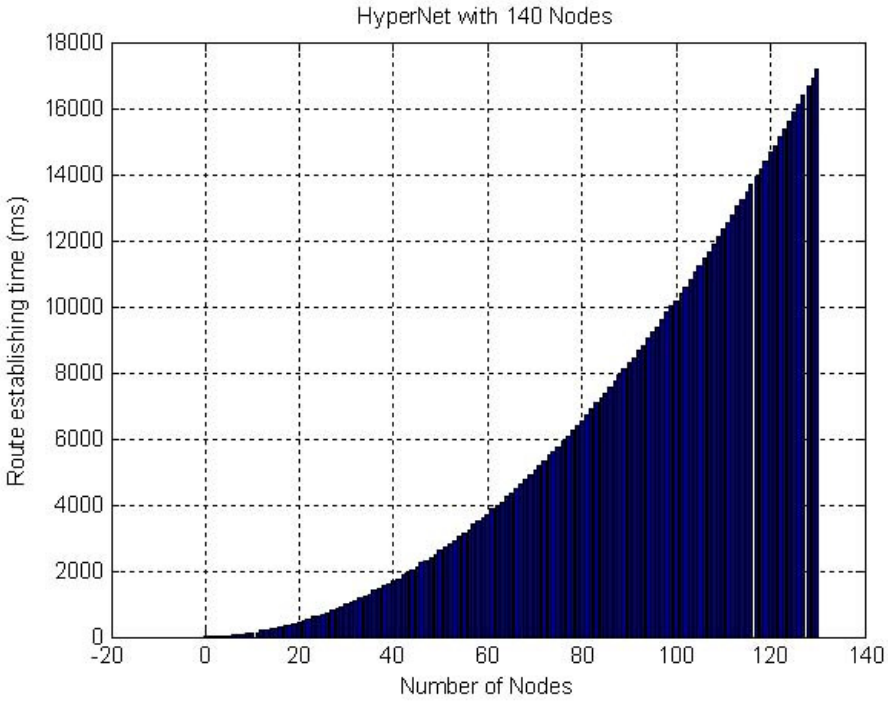


Figure 5. Result of HyperNet with 140 Active Nodes

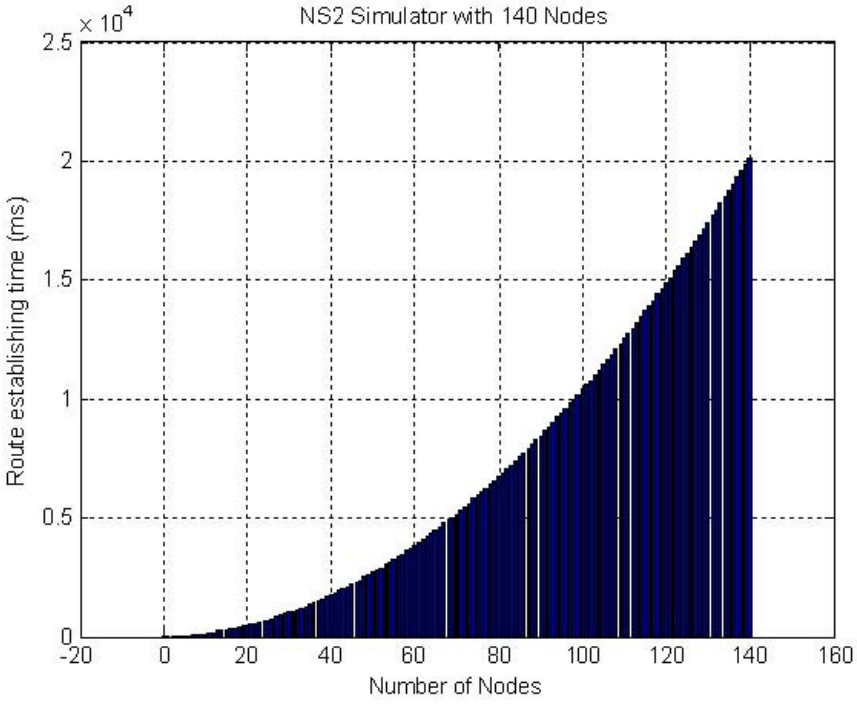


Figure 6. Result of NS2 with 140 Active Nodes

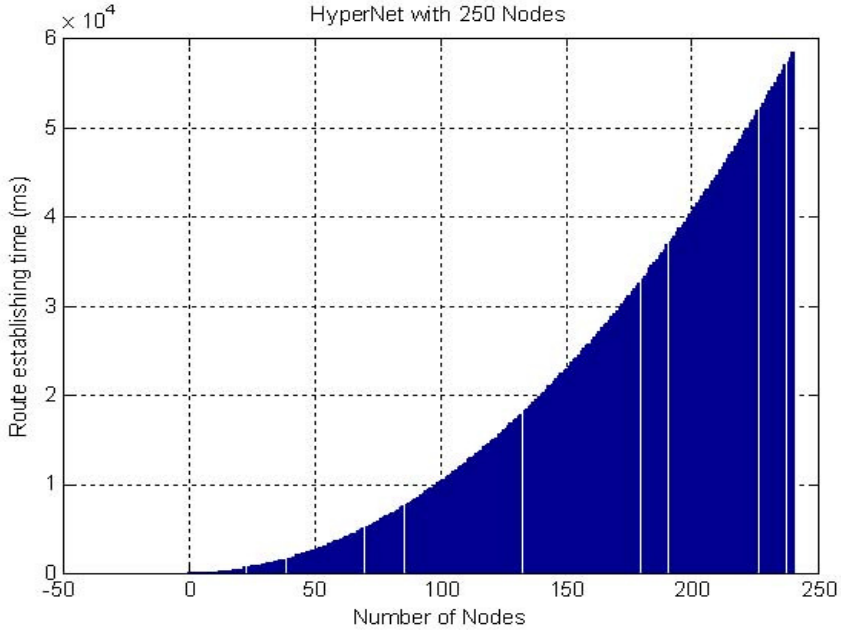


Figure 7. Result of Hyper Net with 250 Active Nodes

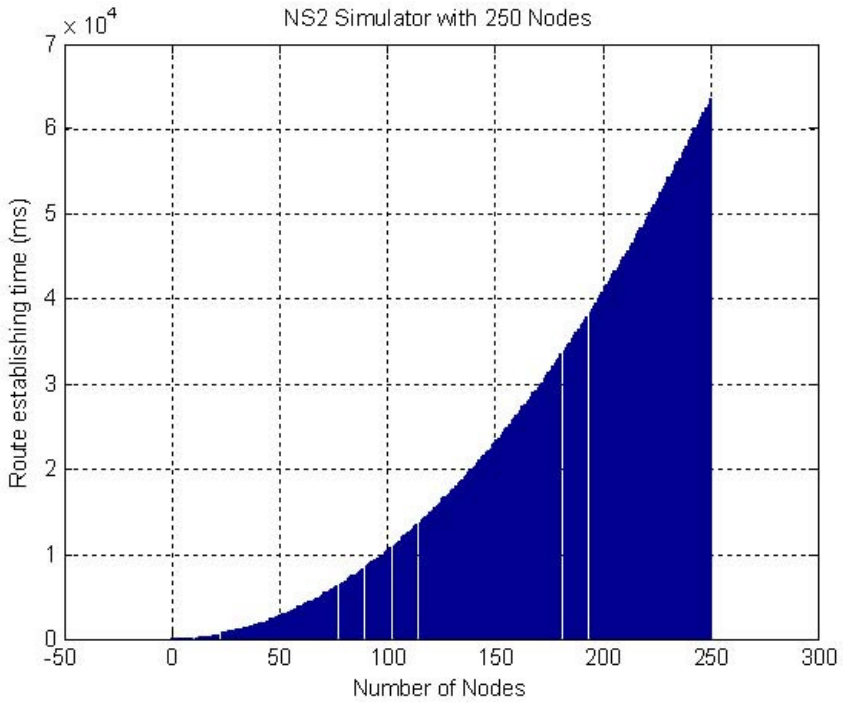


Figure 8. Result of NS2 with 250 Active Nodes

5. Conclusions:

A careful study of the above graphs shows a superior performance of HyperNet over NS2 in establishing route over shorter periods of time. The results are encouraging enough to allow us to study the possibility of refining the NFG Simulator in a larger environment.

The shape of the graphs shows a remarkable profile both for HyperNet and NS2. In fact the shape of the graphs can be approximated by a quadratic polynomial: $x^2 + px + q$ where p and q are constants for a given environment. Further investigation on this polynomial pattern should probably lead us to interesting outcomes.

6. References :

[1] F. Baker, "An outsider's view of MANET," Internet Engineering Task Force document (text file), 17 March 2002.

[2] C. Barrett et al., "Characterizing the Interaction Between Routing and MAC Protocols in Ad-hoc Networks," *Proc. MobiHoc 2002* , pp. 92-103

[3] J. Broch et al., "A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols," *Proc. Mobicom '98*.

[4] K.-W. Chin, et al., "Implementation Experience with MANET Routing Protocols," *ACM SIGCOMM Computer Communications Review*, Nov. 2002, pp. 49-59. Available online.

[5] C. Elliott and B. Heile, "Self-Organizing, Self-Healing Wireless Networks," *Proc. 2000 IEEE Int'l Conf. on Personal Wireless Comm.*, pp. 355-362.

[6] Z. J. Haas, et al., eds., Special Issue on Wireless Ad Hoc Networks, *IEEE J. on Selected Areas in Communications*, Vol. 17, No. 8 (August 1999).

[7] P. Johansson et al., "Scenario-based Performance Analysis of Routing Protocols for Mobile Ad-hoc Networks," *Proc. Mobicom '99*, pp. 195-206.

Authors Short Biography

Siddesh.G.K. is an Assistant Professor in the Department of Electronicsand Communication Engineering, Sri Bhagawan Mahaveer Jain College of Engineering, Jain University, Belgaum. He obtained his Bachelor degree in Electronics and Communication Engineering from Dayananda Sagar College of Engineering ,Bangalore University and Master degree in Digital Electronics and Advanced Communication from University Manipal Institute of Technology, Manipal University. He is pursuing Ph.D. in Electronics and Communication Engineering, Visvesvaraya Technological University, Belgaum. His research interest includes Computer Networks and Wireless Communication.

K N Muralidhara obtained his BE degree in Electronics and Communication from University of Mysore in 1981. He completed the ME and Ph.D. degrees in 1990 and 1998 respectively from Indian Institute of Technology, Roorkee (formerly known as University of Roorkee, Uttaranchal , India). At present, he is working as Professor and Head of the Dept. of Electronics and Communication Engineering, PES College of Engineering, Mandya, Karnataka, India, Visvesvaraya Technological University. He is guiding for research scholars for award of PhD degree from VTU, Karnataka.He has altogether 20 intenational/ national journals/ conferences to his credit. His research interests includes in the areas of Electronic Devices, VLSI, Microprocessor and Microcontroller applications, Embedded systems and Wireless Communication

.