PERFORMANCE EVALUATION OF DIFFERENT NETWORK TOPOLOGIES BASED ON ANT COLONY OPTIMIZATION

Saiteja Perumbuduru[1] and Dr. Joydip Dhar[2]

[1]Masters in Technology, Indian Institute of Information Technology and Management, Gwalior, India
tejapv@gmail.com
[2]Dr. Joydip Dhar, Associate Professor, Indian Institute of Information Technology and Management, Gwalior, India
jdhar@iiitm.ac.in

ABSTRACT

All networks tend to become more and more complicated. They can be wired, with lots of routers, or wireless, with lots of mobile node. The problem remains the same, in order to get the best from the network; there is a need to find the shortest path. The more complicated the network is, the more difficult it is to manage the routes and indicate which one is the best. The Nature gives us a solution to find the shortest path. The ants, in their necessity to find food and brings it back to the nest, manage not only to explore a vast area, but also to indicate to their peers the location of the food while bringing it back to the nest. Most of the time, they will find the shortest path and adapt to ground changes, hence proving their great efficiency toward this difficult task. The purpose of this paper is to evaluate the performance of different network topologies based on Ant Colony Optimization Algorithm. Simulation is done in NS-2.

KEYWORDS

Network Protocols, Wireless Network, Mobile Network, Ant Colony Optimization, Adaptive Routing

1. INTRODUCTION

The main source of inspiration behind ACO (Ant Colony Optimization) [1][2][3] and ACO routing is a behavior that is displayed by certain species of ants in nature during foraging. It has been observed that ants are able to find the shortest path between their nest and a food source [4]. This is remarkable because each individual ant is a rather simple creature, with very limited vision and computing power, and finding the shortest among several available paths is certainly beyond its capabilities. The only way that this difficult task can be realized is through the cooperation between the individuals in the colony. This algorithm is implemented in different sectors like manufacturing [5], healthcare etc.

The key behind the colony level shortest path behavior is the use of pheromone. This is a volatile chemical substance that is secreted by the ants in order to influence the behavior of other ants and of itself. Pheromone is not only used by ants to find shortest paths, but is in general an important tool that is used by many different species of ants (and also by a lot of other social animals) for a wide variety of tasks that involve coordinated behavior [6].

In the case of the path finding task we describe here, the ants use pheromone to recruit subsequent ants to the paths they have followed. Ants moving between their nest and a food source leave a trail of pheromone behind, and they also preferably go in the direction of high intensities of pheromone. All moving ants leave a trail of pheromone behind. The ants going over the short path reach the destination earlier than those going over the long path. Moreover, they can return faster. This leads temporarily to a higher pheromone concentration on the shortest path. Subsequent ants leaving the nest are attracted by this higher intensity, and go therefore preferably also over the shortest path. As this process continues, the majority of the ants eventually concentrate on the shortest path. It needs to be pointed out however, that the behavior of the ants is never deterministic, so that there will always remain a minority of ants that explore the longer path.

The shortest path finding process of the ants has a number of interesting properties. First of all, it is highly distributed and self-organized. There is no central control mechanism; instead, the organization of the behavior emerges from the simple rules of stigmergic communication that are followed by the individual ants. Second, it is highly robust. This is related to the property of self-organization: the system has no single point of failure, but instead consists of a high number of individually unimportant agents, so that even significant agent losses do not have a large impact on the performance. Third, the process is adaptive [7]. Since none of the ant behavior is deterministic, and some individuals keep exploring also longer paths, the system can adapt to changes in the environment. Finally, the process is scalable: the process can be scaled to arbitrarily large colonies.

2. PROPOSED WORK

There are many algorithms provided for Ant Colony Optimization but we implemented based on the paper [8]. There are a number of implementations and improvements [9] done for this algorithm but the basic aim is to tweak the memory and make it efficient. The work focuses on implementing the algorithm and evaluating the performance [10] of different Network topologies [11]. The Ant Colony Optimization is described as follows:

2.1. Ant Colony Optimization

ACO [12] is a meta heuristic in which a colony of artificial ants cooperates in finding good solutions to discrete optimization problems. Each ant of the colony exploits the problem graph to search for optimal solutions. An 'artificial ant', unlike natural counterparts, has a memory in which it can store information about the path it follows. Every ant has a start state and one or more terminating conditions. The next move is selected by a probabilistic decision rule [13] that is a function of locally available pheromone trails, heuristic values as well as the ant's memory. Ant can update the pheromone trail associated with the link it follows. Once it has built a solution, it can retrace the same path backward and update the pheromone trails. ACO algorithm is interplay of three procedures as described in [12].

2.1.1. Construct ant solutions

This procedure manages a colony of ants that concurrently and asynchronously visit adjacent states of the considered problem by moving through neighboring nodes of the solution space of the problem's construction graph.

2.1.2. Update pheromones

It is the process by which pheromone trails are modified. The trail value can either increase, as ants deposit pheromone on the components or connections they use, or decrease, due to pheromone evaporation. Net increase/decrease in pheromone value at a given location on trail is determined by difference of deposition and evaporation.

2.1.3. Daemon actions

This procedure is used to implement centralized actions which cannot be performed by single ants.

2.2. Antnet Data structures

AntNet is an ACO algorithm for data network routing proposed by Gianni Di Caro and Marco Dorigo [12]. Mobile agents (artificial ants) act concurrently and independently, and communicate in an indirect way, through the pheromones they read and write locally on the nodes. Each network node k stores two data structures:

2.2.1. Routing table T_k

For each possible destination d and for each neighbor node *n*,

T_k stores a probability value P_{nd} expressing the goodness of choosing n as next node when the destination node is *d*:

$$\boxed{\sum_{n \in N_k} Pnd = 1,\ d \in [1, N],\ N_k = \text{neighbors }(k)}$$

Probability value P_{nd} represents the pheromone concentration along the link from node *k* to neighbor node *n* for destination node *d*.

2.2.2. Traffic model M_k

M_k (μ_d, σ_d^2, W_d) is a statistical model of the traffic situation over the network as seen by node *k*. It is described by the sample mean μ_d and the variance σ_d^2 computed over the trip times experienced by the artificial ants, and by a moving observation window W_d used to store the best value W_{best_d} of the artificial ants' trip time.

2.3. Antnet Algorithm

The AntNet algorithm, as proposed by Di Caro and Dorigo [12], is as follows:

At regular intervals Δt from every network node s, a forward ant $F_{s \to d}$ is launched toward a destination d to discover a feasible, low-cost path to that node and to investigate the load [14] status of the network along the path. If f_{sd} is a measure (in bits or in number of packets) of the data flow s→d, then the probability of creating at node s a forward ant with node d as destination is

$$\boxed{p_{sd} = \frac{f_{sd}}{\sum_{i=1}^{N} f_{si}}}$$

While travelling toward their destination nodes, the forward ants keep memory of their paths and of the traffic conditions found. The identifier of every visited node *i* and the time elapsed since the launching time to arrive at this i-th node are stored in a memory stack.

At each node i, each forward ant headed toward a destination d selects the node j to move to, with a probability Pijd computed as normalized sum of the pheromone τijd with a heuristic value ηij taking into account the length of the j-th link queue of the current node i:

$$\boxed{p_{ij}^{d} = \frac{\tau_{ij}^{d} + \alpha\eta_{ij}}{1 + \alpha(|Ni| - 1)}}$$

The heuristic value ηij is a normalized value function of the length qij of the queue on the link connecting the node i with its neighbor j:

$$\boxed{\eta_{ij} = 1 - \frac{q_{ij}}{\sum_{i=1}^{|N_i|} q_{i1}}}$$

The value of α weighs the importance of the heuristic value with respect to the pheromone values.

If a cycle is detected, the cycle's nodes are removed and all the memory about them is deleted. When an ant reaches a node that is already in its memory, a cycle is detected and all the nodes until this recurrent node are deleted from the ants memory.

When the destination node d is reached, the agent $F_{s\rightarrow d}$ generates backward ant BB $d\rightarrow s$, transfers to it all of its memory, and is deleted.

The backward ant takes the same path as that of its corresponding forward ant, but in the opposite direction.

Arriving at a node i coming from a neighbor node, the backward ant updates the local model of the traffic Mi and the pheromone matrix Ti, for all the entries corresponding to the destination node d.

2.3.1. Update traffic model Mi

The estimated mean [15] and variance [16] are updated as follows:

$$\boxed{\begin{aligned} &\mu_{id} \leftarrow \mu_{id} + \varsigma(o_{i\rightarrow d} - \mu_{id}) \\ &\sigma_{id}^2 \leftarrow \sigma_{id}^2 + \varsigma((o_{i\rightarrow d} - \mu_{id})^2 - \sigma_{id}^2) \end{aligned}}$$

where $o_{i\rightarrow d}$ is the observed ant's trip time from node i to destination d. The factor ς weighs the number of most recent samples that will really affect the average.

2.3.2. Update pheromone matrix Ti

The backward ant $B_{d\rightarrow s}$ moving from node f to node i increases the pheromone values $\tau_{ifd'}$. Pheromones τijd' for destination d' of the other neighboring nodes j, j Є Ni, j ≠ f, evaporate implicitly by normalization.

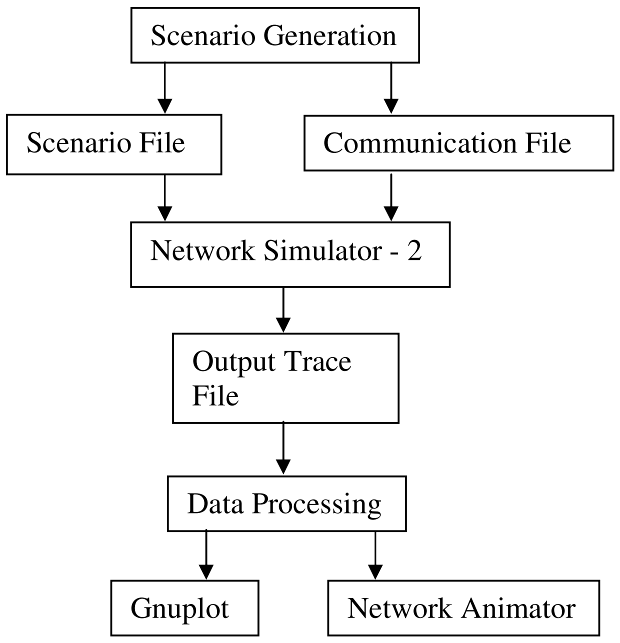
3. SIMULATION ENVIRONMENT

The simulator we have used to simulate the ad-hoc routing protocols in is the Network Simulator 2 [17] from Berkeley. To simulate the mobile wireless radio environment we have used a mobility extension [18] to NS that is developed by the CMU Monarch project at Carnegie Mellon University.

3.1. Network Simulator

Network simulator 2 is the result of an on-going effort of research and development that is administrated by researchers at Berkeley. It is a discrete event simulator targeted at networking research. It provides substantial support for simulation of TCP, routing, and multicast protocols. The simulator is written in C++ and a script language called OTcl. NS -2 [17] uses an Otcl interpreter towards the user. This means that the user writes an OTcl script that defines the network (number of nodes, links), the traffic in the network (sources, destinations, type of traffic) and which protocols it will use. This script is then used by ns during the simulations. The result of the simulations is an output trace file that can be used to do data processing (calculate delay, throughput etc) and to visualize the simulation with a program called Network Animator (NAM). NAM is a very good visualization tool that visualizes the packets as they propagate through the network.

We chose NS-2 [17] for the implementation of AntNet because it is an event driven simulator which enables the generation and forwarding of packets according to the route logic. The C++ classes for agent, packet and routing can be extended to implement new entities. Also trace files provide nice details of packet flow that enables to check the correctness and accuracy of algorithm.



The simulation overview of the proposed model is given in the figure above.

4. ANTNET IMPLEMENTATION

We could successfully extend network simulator NS-2 [17] to implement AntNet simulation. We have simulated artificial ants by implementing ant packets. Ant packets are of two types representing forward ants and backward ants flow through the network according to the algorithm and keep a memory of the path traversed to discover optimal solution. We defined a routing agent called Antnet agent which is responsible to implement route logic and perform the protocols functionality by processing ant packets according to the flow of the algorithm. Routing agent is implemented through class Antnet derived from class Agent. We defined Tcl hooks that enable simulation of AntNet routing algorithm through Tcl script. Hence, it can be simulated for varying simulation time and time interval at which ants are launched.

4.1. Packet type

We have implemented artificial ants by defining a new packet type in NS-2 [17]. Besides information on packet source, destination, sequence number and length, the packet header also has a field that identifies it as forward ant or backward ant. A local data structure represents the memory of a packet, which stores node addresses and trip time to corresponding nodes that the packet traversed.

4.2. Routing agent

We have defined class Antnet inherited from class Agent. This implements the Antnet routing agent to which a node can be associated. The routing table and the traffic of a particular node are stored as local data structures. It is responsible for creating and forwarding forward ant packets and backward ant packets. The modules in the Antnet class implement the functionality of receiving ant packets, processing them according to the flow of the algorithm which includes determining destination node and next hop node, and forwarding them to Antnet agent associated with next hop node. Whenever Antnet agent receives a forward ant, it adds the node address and trip time to the ant's memory, determines next node to which it has to be forwarded and sends it to Antnet agent associated with that node. When Antnet agent receives a forward ant destined to it, it creates a backward ant packet and sends it along the reverse path as stored in ant's memory. When Antnet agent receives a backward ant packet, it updates the routing table and local traffic model and sends it further along the path that it has been retracing. Upon complete traversal of path, the backward ant packet is destroyed. The agent implements cycle detection and elimination, packet forwarding and update of traffic model and routing table according to the AntNet algorithm.

5. NETWORK TOPOLOGIES

After implementing the Antnet algorithm, we tried to apply the algorithm on different network topologies. Some information on network topologies is given below.

Network topology is the *arrangement* or *mapping* of the elements (links, nodes, etc.) of a network, especially the physical (real) and logical (virtual) interconnections between nodes. **Network topology** is sometimes also referred to the study of the *arrangement* or *mapping* of the elements (links, nodes, etc.) of a network. A local area network (LAN) is one example of a network that exhibits both a physical topology and a logical topology. Any given node in the LAN will have one or more links to one or more other nodes in the network and the mapping of these links and nodes onto a graph results in a geometrical shape that determines the physical topology of the network. Likewise, the mapping of the *flow of data* between the nodes in the network determines the logical topology of the network. It is important to note that the physical and logical topologies *might* be identical in any particular network but they also may be *different*. Any particular network topology is determined only by the graphical mapping of the configuration of physical and/or logical connections between nodes. LAN Network Topology is, therefore, technically a part of graph theory. Distances between nodes, physical interconnections, transmission rates, and/or signal types may differ in two networks and yet their topologies may be identical.

There are many network topologies existing at present. But, we mainly tried to concentrate our implementation only to these five network topologies, since they are most commonly used and are easily manageable.

The five network topologies that we concentrated are:

1. Mesh Topology
2. Ring Topology
3. Star Topology
4. Line Topology
5. Random Topology

All these network topologies have certain advantages over one another but they are considered to be the most used network topologies. Here we present the graphs for various network topologies that are used for our simulation.

5.1. Mesh Topology

The figure which we took for our simulation is

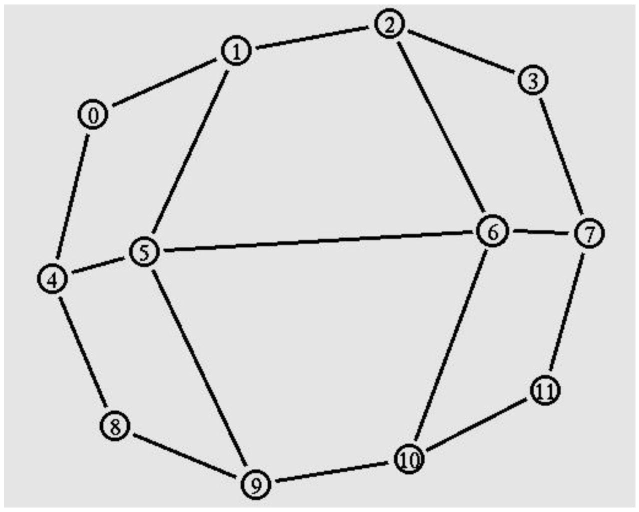


Figure 1. Mesh Topology

5.1.1 Throughput

We present individually three different graphs for a set of 4 nodes.

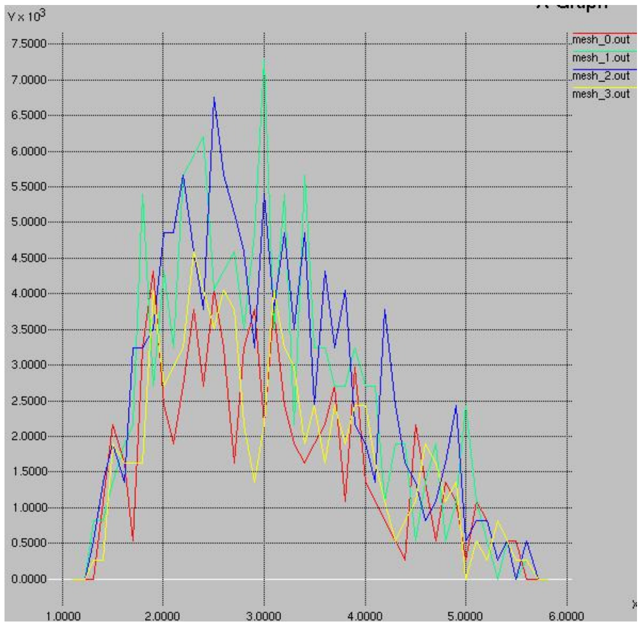


Figure 2. Throughput for the nodes 0,1,2,3 of mesh topology. x-axis denotes time interval and y-axis denotes no of bytes sent.

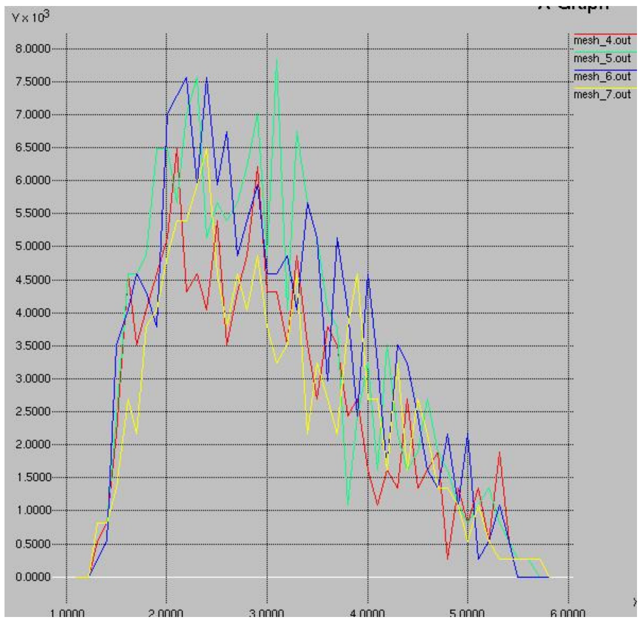


Figure 3. Throughput for the nodes 4,5,6,7 of mesh topology. x- axis denotes time interval and y-axis denotes no of bytes sent.

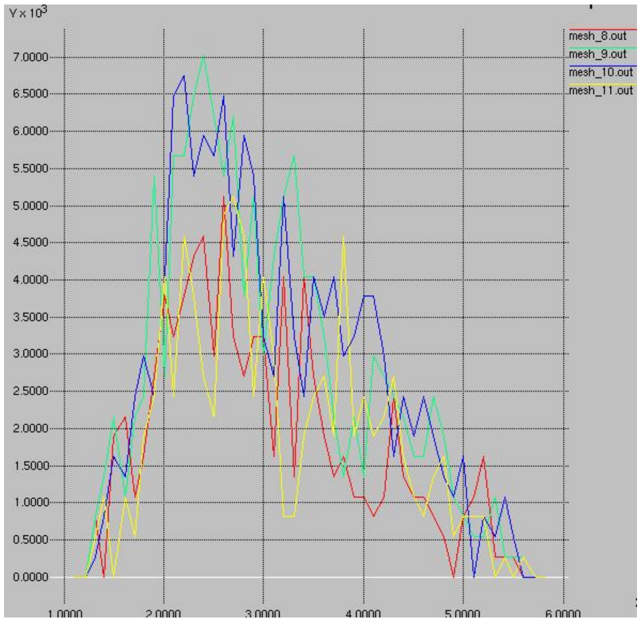


Figure 4. Throughput for the nodes 8,9,10,11 of mesh topology. x- axis denotes time interval and y-axis denotes no of bytes sent.

5.1.2. Delay

We note down the delay of all the nodes at 1.00000 second and 1.03000 second and compare them with all the other topologies. The convergence time can be as big as anything as in this case it is 6 sec, we cannot compare the delays at all the time intervals through table. we can do that in graph but cannot determine correctly due to closeness of delays.

Table 1. Delay at any particular node.

	At 1.00000th second	At 1.03000th second
Node 0	0.310001 ms	0.620002 ms
Node 1	1.240002 ms	1.240003 ms
Node 2	0.620003 ms	1.860005 ms
Node 3	0.310001 ms	1.550004 ms
Node 4	0.620003 ms	2.480007 ms
Node 5	0.620003 ms	0.930003 ms
Node 6	1.240002 ms	0.310001 ms
Node 7	3.410001 ms	1.550004 ms
Node 8	1.240002 ms	3.410010 ms
Node 9	2.170006 ms	1.240004 ms
Node 10	2.480007 ms	1.240004 ms
Node 11	1.240002 ms	1.550004 ms

5.2. Ring Topology

The figure which we took for our simulation is

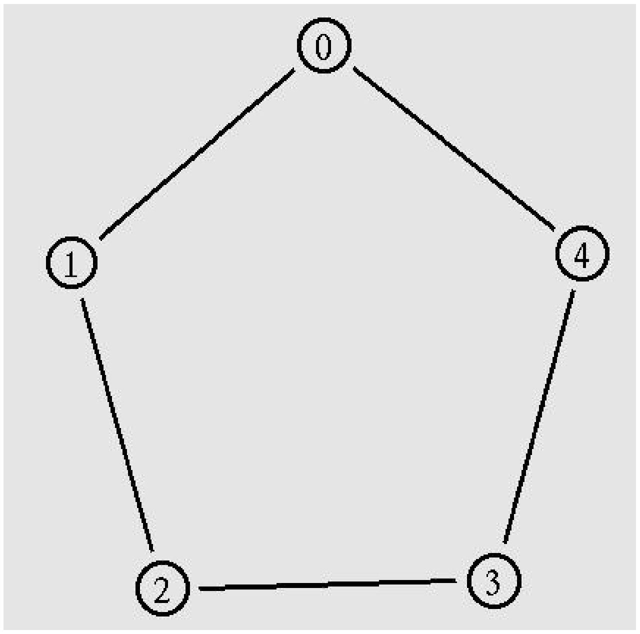


Figure 5. Ring Topology

5.2.1 Throughput

Throughput is given for all the nodes in a single graph as below

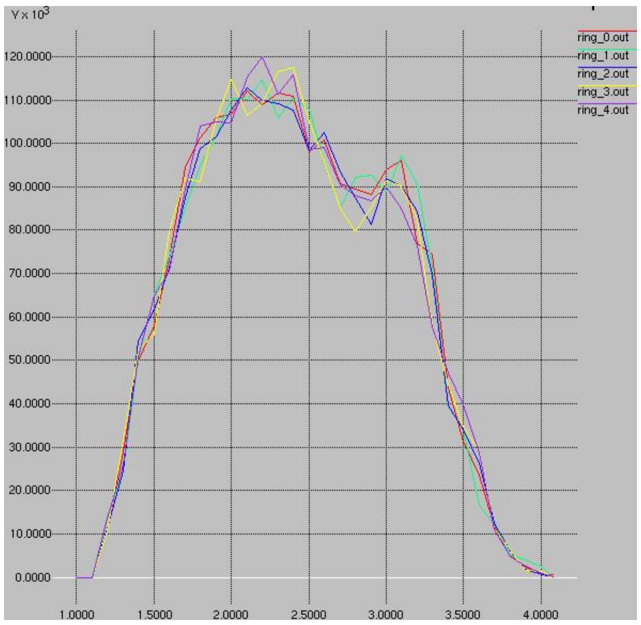


Figure 6. Throughput for all the nodes of ring topology. x- axis denotes time interval and y-axis denotes no of bytes sent.

5.2.2 Delay

We note down the delay of all the nodes at 1.00000 second and 1.03000 second and compare them with all the other topologies. The convergence time can be as big as anything as in this case it is 4 sec, we cannot compare the delays at all the time intervals through table. We can do that in graph but cannot determine correctly due to closeness of delays.

Table 2. Delay at any particular node

	At 1.00000 sec	At 1.03000 sec
Node 0	0.620002ms	0.930003 ms
Node 1	1.930003ms	0.930003 ms
Node 2	0.620003 ms	1.240004 ms
Node 3	0.620002 ms	0.930003 ms
Node 4	0.620003 ms	0.620002 ms

5.3 Star Topology

The figure which we took for this topology is

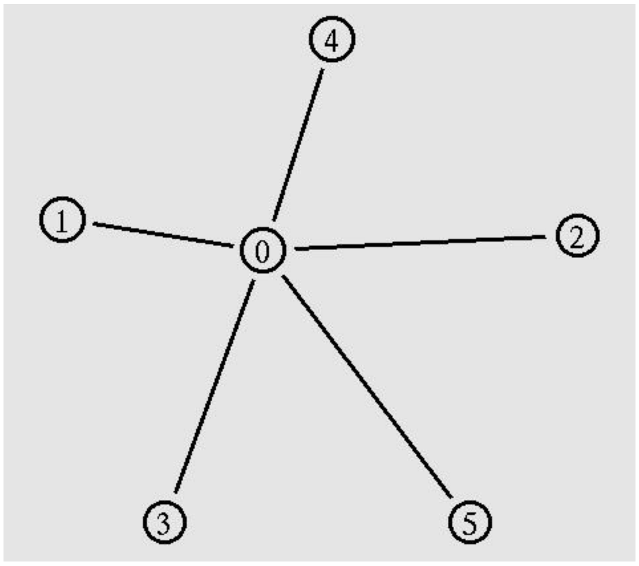


Figure 7. Star Topology

5.3.1 Throughput

Throughput is given for all the nodes is given in single graph as below

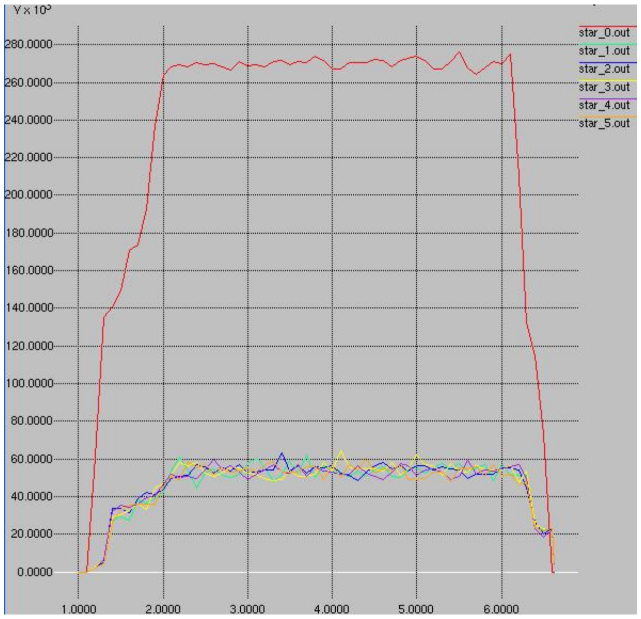


Figure 8. Throughput for all the nodes of star topology. x- axis denotes time interval and y-axis denoted no of bytes sent.

5.3.2 Delay

We note down the delay of all the nodes at 1.00000 second and 1.03000 second and compare them with all the other topologies. The convergence time can be as big as anything as in this case it is 6 sec, we cannot compare the delays at all the time intervals through table. We can do that in graph but cannot determine correctly due to closeness of delays.

Table 3. Delay at any particular node

	At 1.00000 sec	At 1.03000 sec
Node 0	0.155000 ms	0.155000 ms
Node 1	0.620002 ms	0.310001 ms
Node 2	0.310001 ms	0.310001 ms
Node 3	0.620002 ms	0.310001 ms
Node 4	0.620002 ms	0.310001 ms
Node 5	0.620002 ms	0.620002 ms

5.4 Line Topology

The figure which we took for simulation is

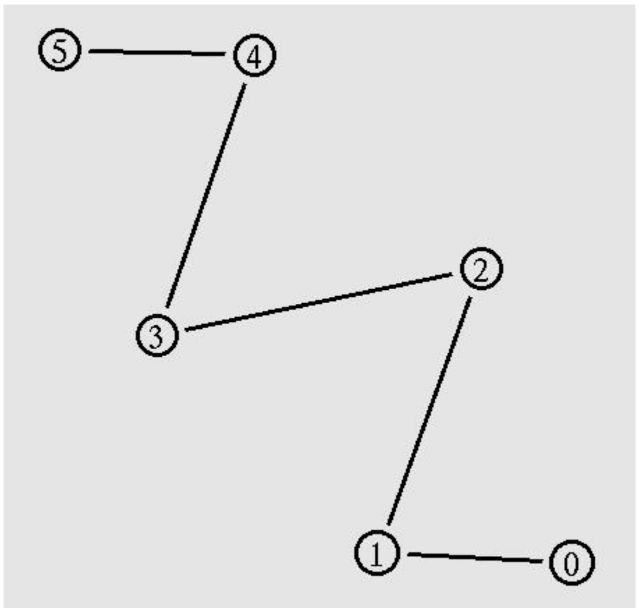


Figure 9. Line Topology

5.4.1 Throughput

Throughput is given for all the nodes is given in single graph as below

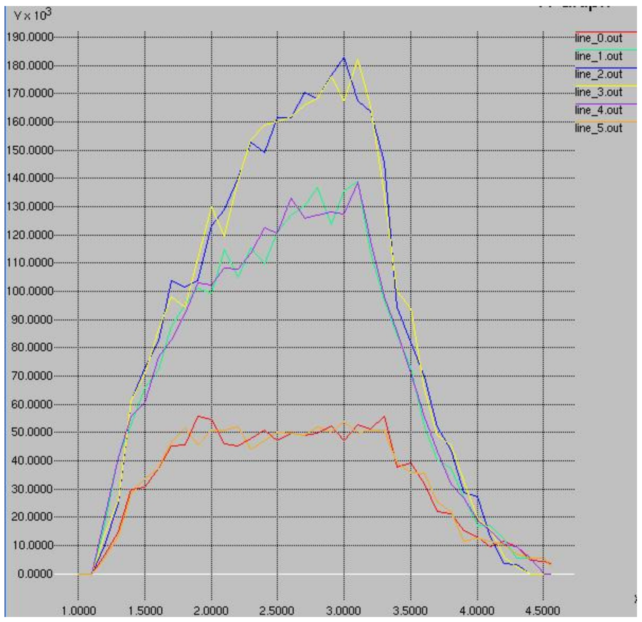


Figure 10. Throughput for all the nodes of line topology. x- axis denotes time interval and y-axis denoted no of bytes sent.

5.4.2 Delay

We note down the delay of all the nodes at 1.00000 second and 1.03000 second and compare them with all the other topologies. The convergence time can be as big as anything as in this case it is 6 sec, we cannot compare the delays at all the time intervals through table. We can do that in graph but cannot determine correctly due to closeness of delays.

Table 4. Delay at any particular node

	At 1.00000 sec	At 1.03000 sec
Node 0	0.310001 ms	1.550004 ms
Node 1	0.155000 ms	0.155000 ms
Node 2	0.310001 ms	0.620002 ms
Node 3	0.310001 ms	0.155000 ms
Node 4	0.155000 ms	0.620002 ms
Node 5	0.310001 ms	0.310001 ms

5.5 Random Topology

The figure which we took for simulation is

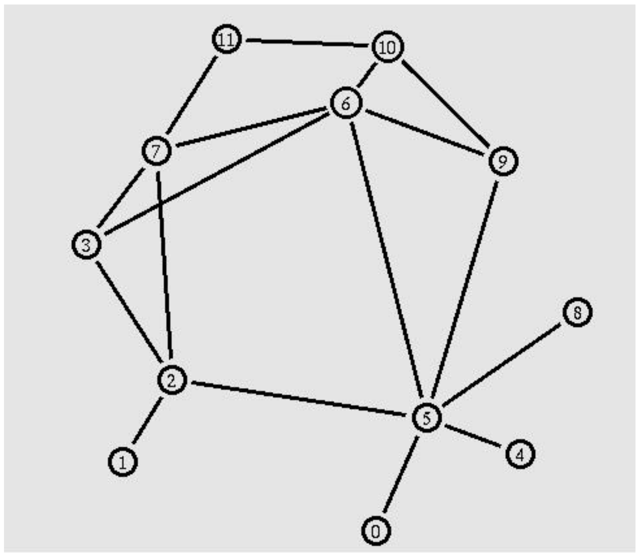


Figure 11. Random Topology

5.5.1 Throughput

Throughput is given in three different graphs for a set of 4 nodes as below

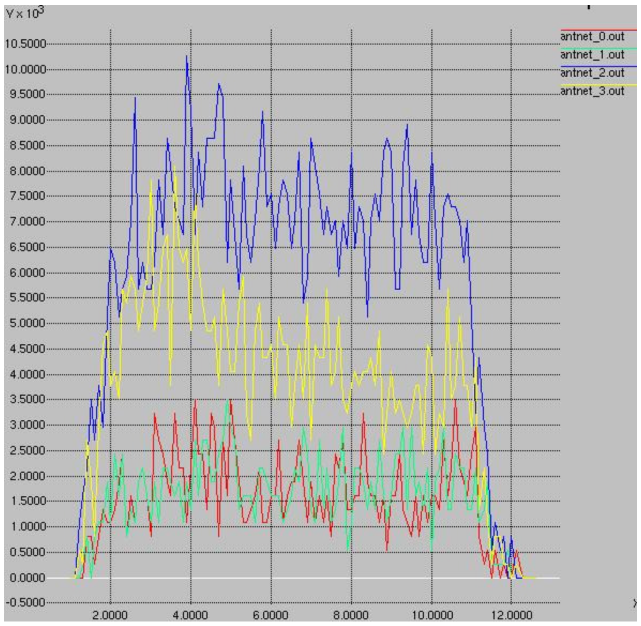


Figure 12. Throughput for the nodes 0,1,2,3 of random topology. x- axis denotes time interval and y-axis denoted no of bytes sent.

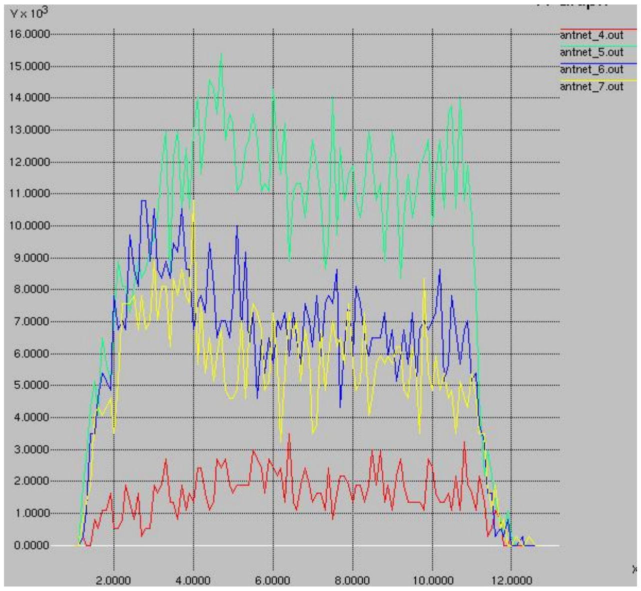


Figure 13. Throughput for the nodes 4,5,6,7 of random topology. x- axis denotes time interval and y-axis denoted no of bytes sent.

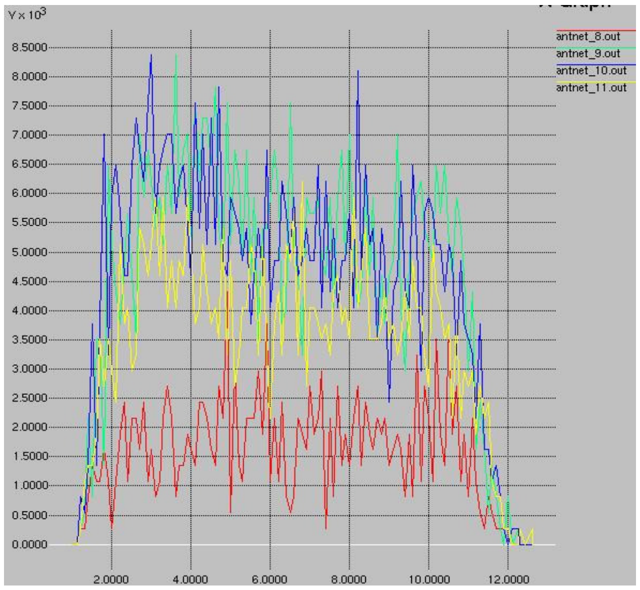


Figure 14. Throughput for the nodes 8,9,10,11 of random topology. x- axis denotes time interval and y-axis denoted no of bytes sent.

5.5.2 Delay

We note down the delay of all the nodes at 1.00000 second and 1.03000 second and compare them with all the other topologies. The convergence time can be as big as anything as in this case it is 6 sec, we cannot compare the delays at all the time intervals through table. We can do that in graph but cannot determine correctly due to closeness of delays.

Table 5. Delay at any particular node

	At 1.000 sec	At 1.030 sec
Node 0	0.930003 ms	0.310001 ms
Node 1	2.480007 ms	0.930003 ms
Node 2	0.930003 ms	0.620002 ms
Node 3	1.085003 ms	1.085003 ms
Node 4	0.310001 ms	1.860005 ms
Node 5	0.155000 ms	0.155000 ms
Node 6	1.550005 ms	2.170006 ms
Node 7	1.240004 ms	1.550004 ms
Node 8	0.620002 ms	0.620002 ms
Node 9	0.930003 ms	0.310001 ms
Node 10	1.860005 ms	0.620002 ms
Node 11	1.860005 ms	0.930003 ms

6. CONCLUSION AND FUTURE WORK

We simulated our assumptions with the help of NS-2 by writing tcl scripts for each topology such that they use ant packets and antnet routing agent. We took different simulation times for different topologies based on their convergence time.

We analyzed that incase of Mesh topology and Ring topology the throughput at all the nodes is maximum but nodes in Ring topology have highest delay compared to Mesh topology. In case of star topology throughput of one node is at maximum. Same is the case with Line topology the nodes that are present in between have the highest throughput compared to the other ones. We can also infer that Antnet algorithm gives suitable results with any kind network. We conclude by saying that Antnet algorithms can be used when working with different types of Static Networks. The only problem with ACO algorithm is it cannot be used effectively in Dynamic Environment.

We analyzed that there are many [19] Routing Algorithms to find shortest path, such as Artificial Potential Field [20], Fuzzy Logic [21], Neural Networks [22], Genetic Algorithm [23] and so on. However these Algorithms can’t reach an ideal solution separately in dynamic environment. Our future works involve identifying the combination of any two algorithms and simulate them across different topologies. We would also like to consider Quality of Service [24] using efficient energy parameters. Ultimate objective would however be to find the best possible combination of algorithms that can also be used in dynamic environment.

7. REFERENCES

[1] Lucas Lessing, Irina Dumitrescu and Thomas Stützle, A Comparison Between ACO Algorithms for the Set Covering Problem, Ant Colony, Optimization and Swarm Intelligence 4th International Workshop, ANTS 2004, Brussels, Belgium, September 5-8, 2004

[2] Adnan Acan, An External Memory Implementation in Ant Colony Optimization, Ant Colony Optimization and Swarm Intelligence 4th International Workshop, ANTS 2004, Brussels, Belgium, September 5-8, 2004

[3] Daniel Merkle and Martin Middendorf, Competition Controlled Pheromone Update for Ant Colony Optimization, Ant Colony Optimization and Swarm Intelligence 4th International Workshop, ANTS 2004, Brussels, Belgium, September 5-8, 2004

[4] S. Goss, S. Aron, J. L. Deneubourg, and J. M. Pasteels. Self-organized shortcuts in the Argentine ant. Naturwissenschaften, 76:579–581, 1989.

[5] Adil Baykasoğlu, Türkay Dereli, Two-sided assembly line balancing using an ant-colony-based heuristic, The International Journal of Advanced Manufacturing Technology, Volume: 36 , Issue: 5-6 (2006)

[6] B. H¨olldobler and E.O. Wilson. The Ants. Springer-Verlag, Berlin, Germany, 1990

[7] Emmanouil A. Panaousis, Tipu A. Ramrekha, Grant P. Millar and Christos Politis, Kingston University London, United Kingdom, Adaptive and Secure Routing Protocol for Emergency Mobile Ad Hoc Networks , IJWMN 2010, Volume 2. Number 2

[8] Gianni Di Caro and Marco Dorigo, AntNet: Distributed Stigmergic Control for Communication Networks. Journal of Artificial Intelligence Research, 9, 1998.

[9] Ying Zhang, Lukas D. Kuhn and Markus P.J. Fromherz, Improvements on Ant Routing for Sensor Networks, Ant Colony Optimization and Swarm Intelligence 4th International Workshop, ANTS 2004, Brussels, Belgium, September 5-8, 2004

[10] Jyotsna Sengupta^1 and Er. Gurpreet Singh Grewal^2, ^{1}Punjabi University,India and ^{2}GNDEC, India, Performance evaluation of IEEE 802.11 MAC layer in supporting delay sensitive services , IJWMN 2010, Volume 2. Number 1

[11] ATIS committee PRQC. "network topology". *ATIS Telecom Glossary 2007*. Alliance for Telecommunications Industry Solutions. http://www.atis.org/glossary/definition.aspx?id=3516.

[12] Marco Dorigo and Thomas Stutzle, Ant Colony Optimization, 3rd~ed. Harlow, England: Addison-Wesley, 1999.

[13] Arkes Hal R, Dawes Robyn M and Christensen Caryn, Factors influencing the use of a decision rule in a probabilistic task, Organizational Behavior and Human Decision Processes, 1986, Volume 37

[14] R. Schoonderwoerd, O. Holland, J. Bruten and L. Rothkrantz, Ant-based load balancing in telecommunication networks. Adaptive Behaviour, 5(2), 169-207, 1997.

[15] http://en.wikipedia.org/wiki/Mean, Mean reference document

[16] http://en.wikipedia.org/wiki/Variance, Variance reference document

[17] http://www.isi.edu/nsnam/ns/doc/index.html: Reference Manual for Network Simulator

[18] http://www.monarch.cs.rice.edu/cmu-ns.html : Reference manual for mobility extensions for NS-2

[19] M.Rajesh Babu, S.Selvan, A Lightweight and Attack Resistant Authenticated Routing Protocol for Mobile Adhoc Networks , IJWMN 2010, Volume 2, Number 2.

[20] Zhiye Li, Xiong Chen and Wendong Xiao. A New Motion Planning Approach Based on Artificial Potential Field in Unknown Environment. PDCAT 2004, LNCS 3320, pp. 376-382

[21] Hee Rak Beom and Hyung Suck Cho, A Sensor-Based Navigation for Mobile Robot Using Fuzzy Logic and Reinforcement Learning, IEEE Trans. On SMC, 1995,25(3),464~477

[22] Ni Bin, Chen Xiong, Zhang Liming, and Xiao Wendong. Recurrent Neural Network for Robot Path Planning. PDCAT 2004, LNCS 3320, pp. 188–191, 2004

[23] Shuhua Liu, Yantao Tian, Jinfang Liu. Multi Mobile Robot Path Planning Based on Genetic Algorithm. Proceedings of the 5th World Congress on Intelligent Control and Automation, 2004, 4706-4709

[24] Dr. Shuchita Upadhayaya and Charu Gandhi, Quality of Service Routing in Mobile Ad Hoc Networks Using Location and Energy Parameters, IJWMN 2009, Volume 1. Number 2