International Journal of Research in Engineering and Science (IJRES)
ISSN (Online): 2320-9364, ISSN (Print): 2320-9356
www.ijres.org Volume 3 Issue 4 ǁ April. 2015 ǁ PP.49-58

Optimal Data Collection from a Network using Probability Collectives (Swarm Based)

Abdulkadir Ahmed[1], Olalekan Ogunbiyi[2], Tahir Aduragba[3]
[1](Electrical and Computer Engineering, Kwara State University, Malete, Nigeria)
[2](Electrical and Computer Engineering, Kwara State University, Malete, Nigeria)
[3](Electrical and Computer Engineering, Kwara State University, Malete, Nigeria)

ABSTRACT*: This paper contains the implementation of the BeeAdhoc algorithm for data routing in mobile Ad Hoc Network (MANet). The algorithm was inspired by the foraging behaviour of honey bees and its implementation mimics this behaviour. The integration was done on Network Simulator version 2 (NS-2.34) where different scenarios were considered in comparison with other existing state-of-the-art routing algorithms that have been implemented in the chosen simulator. The comparison was carried out between DSR, DSDV, AOMDV which are all multipath routing algorithms as the BeeAdhoc; this gave a better insight to the different behaviour of the algorithms on a common application environment. Throughput, end-to-end delay and routing overhead constitute the indices used for the performance evaluation. Experimental results showed the best performance of BeeAdhoc over, DSDV and AOMDV algorithms.*
Keywords -*BeeAdhoc, Network simulator, Probability collective, Routing, Swarm*

I. INTRODUCTION

The Collective Intelligence (COIN) emerged in the technical report submitted to *National Aeronautics and Space Administration* (NASA) by Wolpert and Tumer and in which they referred to it as any combination of large, distributed collection of interacting computational processes in which there is little or no centralized communication/control, together with a 'global utility' function that rates the possible dynamic histories of the collection [1].

Collective can be described as a group of self-motivated agents that maximise overall system performance through improving on their local objectives [2, 3]. Probability Collectives (PC) is a framework of COIN used in the modelling and control of distributed systems, its concept has been linked to Game theory, statistical physics and optimization [4].

The approach of PC is an efficient means of sampling the joint probability space in order to convert the problem under consideration into a convex space of probability distribution [2]. Approach of COIN is to design a collective whereby every section is seen as an agent which gives an overall view of the system as a Multi-Agent-System (MAS) [5].

Probability Collective (PC) as implemented in the COIN framework, allows each of the agents to select actions from a group of available actions and receive reward based on the achieved objective due to the taken action. The approach is an iterative one and reaches equilibrium in which at some point the agent's reward do not increase any more for taking any action further. This equilibrium concept is known as Nash Equilibrium [3, 5, 6, 7]. According to [6, 7, 8] the advantages that could be derived from the use of PC include: It can be used to solve problems with large number of variables, it can be used to handle constrained problems, it is a distributed solution approach in which agents independently updates their probability distribution at any time instance and can be applied to continuous, discrete or mixed variables, a failed agent can just be considered as one that does not update its probability distribution and this do not have any effect on the other agents, the minimum value of the global cost function can be derived by considering the Maxent Lagrangian equation for each agent. In view of the above, a swarm-based system approach which focuses on honey bee behaviours was implemented in this research.

The focus area for this research was on Ad-Hoc wireless network with mobility (MANet); an ad-hoc network could be described as a network without any form of central control among the nodes, that is, no installed infrastructure like routers are required. In this kind of setup the nodes serve as partial router and aid in routing of information. This research implemented a swarm based system in routing data and comparing with existing approaches.

The problem to be addressed in this work is that of routing and information collection in a network. This includes the execution time of algorithms and its accompanying protocols, propagation delay, throughput and energy consumption.

In response to the issues identified above, objectives were: to identify an appropriate modification to be made to the algorithm, to implement the algorithm with an appropriate network protocol for simulation. It also

includes the incorporation of one or combination of the following features: improved resilience (i.e. faster recovery from node/link failure), reduced energy consumption, higher throughput, and minimised execution time.

1.1 Swarm intelligence

Swarm intelligence is the study of computational systems inspired by the 'COllective INtelligence' (COIN). COIN emerges through the cooperation of large numbers of homogeneous agents in the environment [9]. Literally, Swarm systems are those which mimic the behaviours of animals in optimising/solving real life problems through simulations. Examples include schools of fish, flocks of birds, and colonies of ants. These systems are *decentralized, self-organizing and distributed* in a problem domain [10]. Examples include Particle Swarm Optimisation, Ant Colony Optimisation, Bacterial Foraging Optimisation and Bee Colony Optimisation.

Swarm based systems have been used to solve optimisation problems ranging from salesman problem to routing of packets in data network. This research focused on the Honey Bee behaviour in the routing of packet in a Mobile Ad Hoc Network.

The study of bee behaviour for optimisation processes did not kick off early enough because researchers do not understand how information is being disseminated in the beehive. This became history when Nobel Laureate 'Karl Von Frisch' broke the jinx and structured it into a language in his book *The Dance Language and Orientation in Bees*. He elaborated and explained the meanings of the dances given by the bees after each flight back to the hive and after this, several works relating to the bee behaviour have been embarked upon.

BeeHives is one of the earliest works described in [11] that uses the honey bee behaviour to optimise the energy consumption in routing of data in a wired data network. The work was compared with existing swarm based system (AntNet, Distributed Genetic Algorithm (DGA)) using the *Japanese Internet Backbone* (NTTNET) in OMNeT++ network simulator and was found to outperform others in most of the simulated scenarios [11]. In the work, it was said that "Honey bees evaluate the quality of each discovered food site and only perform a waggle dance for it on the dance floor if the quality is above a certain threshold" [11]. The dance is abstracted into a routing table and it is used to keep track of the information received through all bees sent out that arrives from different neighbours. Two types of bee agents are defined are *short distance bee agent* and *long distance bee agent*; this was based on the study which revealed that more bees explore areas closest to the hive and few going farther from the hive for exploration [11].

Short distance bee agent are only allowed to traverse few hops away from it node in gathering and disseminating information to neighbouring nodes while the long distance bee agent can travel to all parts of the network. The implementation assume network to be in partitions which results from the network topology as foraging zones and foraging regions. Based on this, each node maintains information in its routing table about routes that allow it communicate with all its zone members and a path to the representative node in the region where it belongs for data meant for destinations beyond its coverage.

This mechanism allows the algorithm to reduce routing overhead and aid in efficient routing of data in the network. The implementation on OMNeT++ which was compared with AntNet, Distributed Genetic Algorithm (DGA) and Open Shortest Path First (OSPF), focused on energy consumption in routing of data in a wired network. Beehive outperformed others in most of the simulated scenarios [11].

II. THE BEEADHOC ALGORITHM

This was inspired by the foraging behaviour of honey bees and its implementation is to optimise the routing of data in a mobile Ad Hoc network. There are several existing algorithms such as DSR, DSDV, AODV; designed for this type of environment and their respective performances would be compared.

BeeAdhoc routing algorithm is a reactive type of routing protocol in that paths/routes to a destination are only discovered when there is a data to be delivered to that destination. It also uses the *source* routing options of IP, in that the paths to a destination are embedded in the header of the packet which get reviewed as the packet traverses the network.

This is implemented as a layer 3 protocol of the ISO/OSI standard and the idea of abstraction in the standard makes the algorithm independent of lower or upper layer in addition to the ease of integration over any platform. All nodes in its implementation are considered to be a hive and packets sent out also to be a bee. The major mechanisms of the algorithm are the entrance, packing floor and the dance floor and also three major types of bees are implemented.

2.1. Bee Types

The bee names are absorbed from the real honey bee colony; actually they refer to control packets and other types used in the implementation. Three types of bees are used in this algorithm. These are the scout (for route discovery), the forager (to transport data) and the packer (for data collection from the upper layer).

Packers: These are created at the packing floor whenever a packet/data arrives from the upper transport layer (TCP/UDP) to hold the data pending when a forager to the desired destination is found. They remain in the packing floor throughout their life time and are deleted immediately once the appropriate forager is found.

Scouts: This is similar to the route request packet used in other algorithms; it is also created in the packing floor whenever a route to a destination is not available and it's used to find routes. It is a broadcast type of packet to all neighbours, it has in the header the destination address and time to live (TTL) which are part of regular IP header. The header option of BeeAdHoc also appends the route traversed so far and an ID to uniquely identify each scout. All nodes that receives the scout will rebroadcast it if the destination address does not match their address and also if the TTL has not expired. Once the scout arrives at the destination, it will be sent back to the source using the reverse route. At the source it will be passed to the dance floor where a forager will be created from it.

Foragers: These are the bees that transport actual packets in the network from the source node's hive to the destination node's hive. They are kept in the dance floor. They also have an age tag attached and basically this tag is used to note the age of the forager and this is decreased anytime it transport data until it gets to zero when a new paths/routes will have to be requested if there's need to send data to the destination.

2.2. Algorithm Design and Operations

As stated earlier, each node on the network is seen as a bee hive through which the routing information is generated and stored. Again, the nodes are independent of one another in that no control packets are exchanged for routing to be possible.

The design focused on the ISO/OSI layer 3 (network layer) and as such interfaces to the upper transport and lower MAC layer were part of the design. The *packing floor* interacts with the upper layer while the *entrance* interacts with the lower layer. In between these two is the *dance floor* which contains the routing information. The architectural overview is illustrated in Fig. 1.

III. Algorithm Implementation in NS-2.34

As earlier stated, the algorithm here was based on the design from [12]; the focus area in the work discussed there was on energy consumption of various algorithms in comparison with BeeAdhoc. The authors of the work in [12] were contacted and the source code for their implementation was made available for use. Their implementation was on NS-2.29, an older version compared to NS-2.34 used in this work.

On receipt of the source code, there were several compilation errors into NS-2.34 during the integration stage; these were due to the upgrade in the library files present in NS-2.34 compared to NS-2.29. There were also different types of special bees (throughput bee, energy bee, swarm bee etc) declared to enhance its energy consumption which was the focus area of their work.

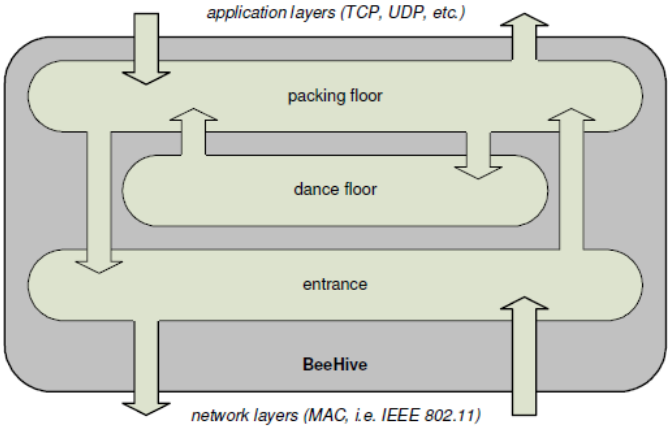

Fig. 1: Architecture Overview

In this implementation, all the library issues that gave compilation errors were resolved and missing variables clearly identified and declared appropriately. Also the special bees usage was disabled to change the focus area of the work presented here.

For this implementation, some of the simulator files need to be modified slightly in other for the algorithm to be integrated. The modification involves in most cases a line of code defining the algorithm's variable and at most a function section.

For the success of this research, we were able to integrate the BeeAdhoc algorithm in the chosen simulator with appropriate modification to make it work. All the modifications made to the simulator files were

written by us and also the library issues mentioned above was debugged by us. A shell script was written to automate the multiple runs of simulations of different scenarios. I also wrote a java program to parse the trace files for analysis. The program was made to compute the throughput, end-to-end delay and routing overhead for the different scenarios in a .csv file which was then used to generate all the graphs. Fig. 2 shows the flow of event that led to the completion of this project.

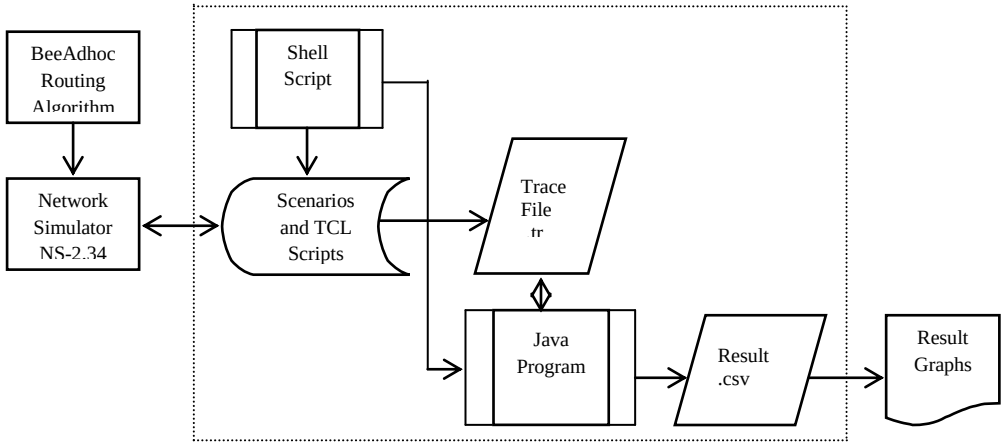

Fig. 2: Project Tasks

3.1. Simulation Scenarios

The Beeadhoc algorithm was evaluated in NS-2.34 and results compared with other state-of-the-art routing algorithms that already exist in the simulator. This section explains briefly the simulation scenarios, performance metric and results.

The type of traffic that was simulated was Constant Bit Rate (CBR) over User Datagram Protocol (UDP). The choice of this was made to aid in determining the actual routing packets used instead of using Transmission Control Protocol (TCP) which could increase the overhead against our wish.

The random waypoint model feature of the simulator was used to generate node and their properties. The nodes were generated with an initial random position and the mobility throughout the simulation run time was made random as their respective position switching was randomised.

The simulator has several topology types that could be used; but for this work the simulation topology was a flat grid that provides a flat surface area, which implies that the surface was free of any object that could negatively affect the radio transmission power of the nodes. The topology area was made up of a square of 1000 x 1000 m^2 for all simulation.

Apart from the scenarios where the number of nodes were varied and mobility speed, all other experiments have the same number of nodes and uses same mobility speed. The nodes moves randomly to a different location from the initial point at a fixed speed throughout the experiment and stay there based on the pause time specified and then moves again.

The wireless radio antenna used was an Omni-antenna (transmitting to all direction) and it was centrally place on the node with a height of 1.5m and the wireless technology adopted was the WaveLan DSSS which operates with 915MHz frequency. This decision was also made because WaveLan operates only with one frequency as stated above which ensures equity in radio transmission frequency of nodes with the same power. Other parameters as used for the experimental simulations are as shown in Table 1.

It is worthy of note to say that different protocols were examined along with the Beeadhoc algorithm and in few simulations DSDV and DSR were not used. This was because DSDV and DSR protocols were part of the oldest available in the simulator and as such it gave segmentation faults during some of the simulations.

The fault was traced to NS-2.34 file named as ns-packet.tcl located in ns-lib and common folders. Further study showed that the mentioned file has the packets structures of most algorithms defined in it; and modifying it could affect the performance of other algorithms or even cause compilation error in NS-2.34.

The observed effect of the segmentation faults on the two algorithms (DSDV and DSR) was basically transmission of lower number of packets than expected in some instances. This effect was seen to have partial effect on the comparison; thereby all instances where the segmentation fault was observed were deleted from the data taken for analysis and another instance ran to bring up the samples to the same number with other algorithms.

3.2. Metrics for Performance Evaluation

Properties of the simulation that was used to evaluate performance of the various algorithms in comparison to one another were defined to include throughput, end-to-end delay and routing overhead.

Throughput: This was defined in percentage as the number of received packets to the number of sent packets in the application layer. The algorithm that has got the highest percentage value is rated the best performed one for that particular scenario.

End-to-End Delay: This was defined as the average of the time it takes all sent packets to be received at the destination. This time is stamped at the moment the packet leaves the sender to include all the delay in the queue up to when it gets to the destination. Only the times spent by the received packets are considered and the total sum of the time spent by all received packet is divided by the number of received packets. The algorithm with the least time is evaluated to be the best performing one for the particular scenario.

Table 1: Simulation Parameters

Parameter	Value
Protocols Examined	AOMDV, BeeAdHoc, DSDV and DSR
Channel Used	Wireless Channel
Network Interface	Wireless Physical
MAC Type	IEEE 802.11
Queue Type	Drop-Tail or Priority Queue
Link Layer Type	Used ARP to resolve IP addresses to MAC address
Antenna Type	Omni Antenna
Default Wireless Physical Setting	914MHz Lucent WaveLAN DSSS
Queue Length	50 Packets
Number of Nodes	10, 20, 30, 40 and 50
Maximum Area	1000 X 1000 meters
Simulation Time	Maximum of 20s
Pause Time	5s
Node Mobility Speed	20, 40, 60, 80, and 100 meters/s
Node Transmitting Range	150, 200, 250, 300 and 350 meters
Packet Size	512 Kb/s
Propagation Type	Two Ray Ground
Node Movement Model	Random Way Point

Routing Overhead: This was defined as the number of packets generated at the network layer which was tagged RTR packets in ensuring that the packets get to the destination. This packets include route request, scouts etc. that are used to find routes. The algorithm with the minimum number of routing overhead is rated the best performing one again in the particular scenario.

IV. RESULTS ANALYSIS

In the simulated experiments, the traffic type explained above was setup. The source node was made constant for all experiments and the destination nodes were randomized in multiple runs.

A shell (bash) script was used to aid in automation of multiple runs of each of the simulated scenarios and generated the required trace files for analysis.

A java program was used to analyse the trace files generated from each runs of the respective scenarios. It calculated the total number of sent packets, received packets, routing overhead, and the average end-to-end delay and create a .csv file in which all the values were written from which the graphs were generated.

The points on the graphs are average of multiple runs ranging from 10 – 20 in most cases; this is aimed at finding out the stochastic behaviour of the algorithm or environment.

4.1 Varying Number of Nodes

In this experiment, the numbers of nodes were varied from between 10 – 50 in different simulations, aimed at observing the behaviour of the algorithms as the number of nodes increases with reference to the performance metrics. It was expected that the routing overhead will increase and possibly with increased end-to-end delay as the number nodes increases but the throughput was envisaged not to be affected by this variation.

From Fig. 3, it was observed that the throughput of Beeadhoc, DSR and AOMDV increases steadily on the average as the number of nodes increases. DSDV had the worst performance in this regards.

The routing overhead are the control packets used by the algorithms to find routes/paths to the required destination as based on their working mechanisms. It was expected that the routing overheads would increase as the number of nodes increases as there would be more nodes to communicate with in the flooding processes. Fig.4 shows the behaviour of the respective algorithms. All had experienced an increase in the routing

overheads, but DSR and DSDV had the best performance in this case. Beeadhoc also outperforms AOMDV on the average.

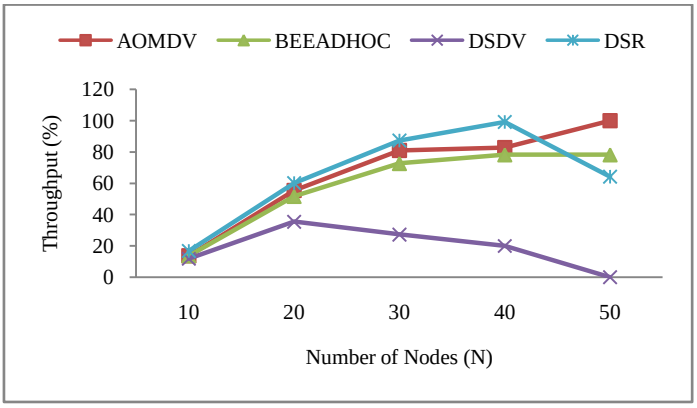

Fig.3: Number of Nodes Vs Throughput

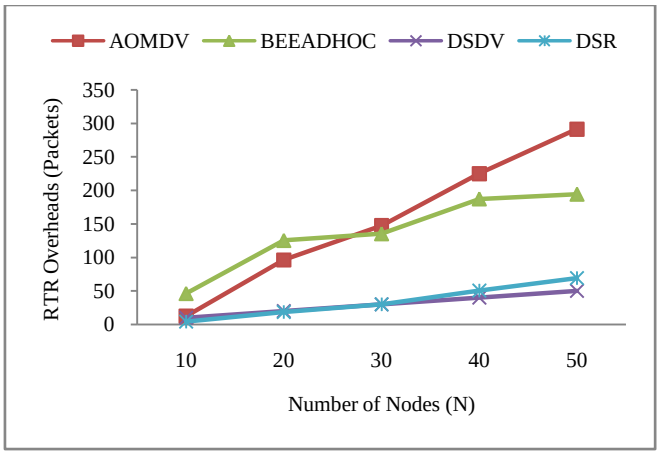

Fig.4: Number of Nodes Vs Routing Overhead

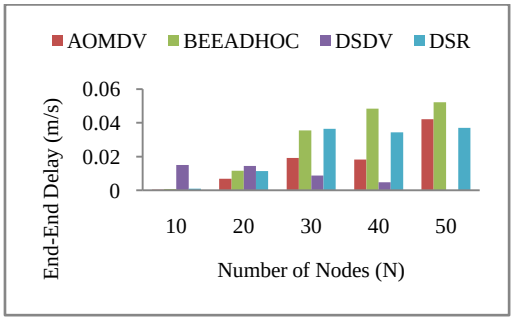

Fig.5: Number of Nodes Vs End-to-End Delay

Fig. 5 shows the behaviours of the algorithms when the average time it takes packets to be delivered at the destinations was considered. Beeadhoc competed well with other state-of-the-art algorithms, though the time is seen to increase as the number of nodes increases as expected at a steady pace. DSDV has an opposite behaviour as the time reduces as the number of nodes increases, this could be tied down to the fact that it already stored multiple routes to all nodes at the beginning and can easily switch on which paths to use as soon as there are packets to be sent out instead of just searching for the routes as others would do.

4.2 Varying Nodes Mobility Speed

Node mobility changes network topology frequently and the aim of this experiment is to observe the behaviour of the algorithms to changing topology. This is aimed at studying the adaptability of the algorithms. Ordinarily, it would be expected that the throughput of the algorithms be affected negatively as the mobility speed increases; this is because the topology changes and more packets would be expected to be dropped.

All the algorithms exhibit different behaviours in this regards, DSDV was seen not to be stable as it goes up and down as the speed increases. Beeadhoc is the most adaptive algorithm to topology changes as the speed had little impact on its throughput. Again, all the algorithms converge to between 18% - 25% when the speed was 80m/s and Beeadhoc and AOMDV was seen to have an improved throughput at higher speed beyond this point as shown on Fig.6. This was repeated for higher speed to be sure of the reaction and the throughput actually increases. This could be the instance of the simulator or that the algorithms actually adapts quickly to changes.

Routing overhead was expected to increase as the speed increases because known paths are to change as the nodes moves around randomly with an increased speed and control packets for route discovery are expected to increase on the overall.

DSDV seem not affected by this as shown in Fig. 7. Beeadhoc experiences an increased routing overhead as the nodes mobility speeds increases as expected. AOMDV has a reverse behaviour compared to Beeadhoc, this again could be tied to the fact that AOMDV uses routing table to store multiple paths to a destination and alternate paths might be found without having to launch new route discovery control packets.

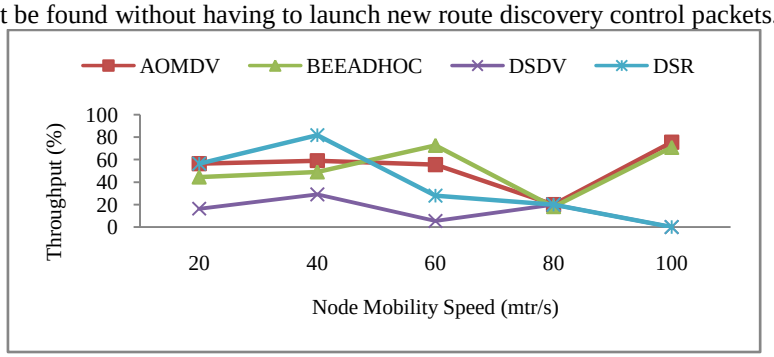

Fig.6: Node Mobility Vs Throughput

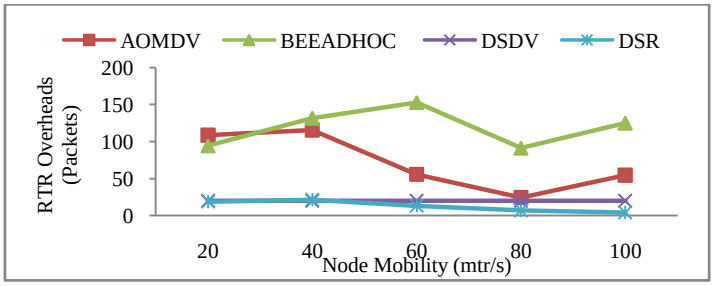

Fig.7: Node Mobility Vs Routing Overhead

End-to-End delay graph in Fig. 8 in comparison with the throughput graph in Fig. 7, it could be deduced that Beeadhoc delayed the packets longer while it searches for new routes to the destination which gave it an edge over others in better throughput but made it the worst performed in the delay chat.

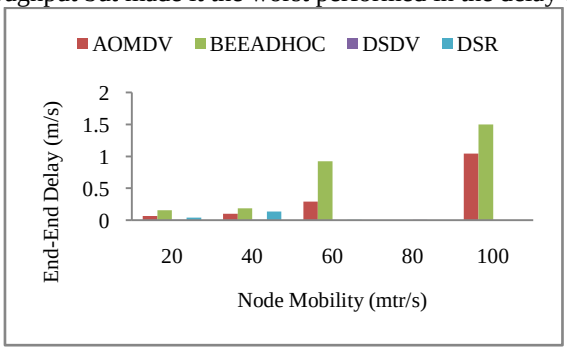

Fig.8: Node Mobility Vs End-to-End Delay

4.3 Varying Number of Failed Nodes

Network failure was another way the algorithms adaptability features to network changes was verified. In these experiments, nodes were randomly disabled from participating in any activity in the network after some time. The number of failed nodes was varied and the results are shown in Fig. 9.

DSDV had the worst throughput over the range of failed nodes while the reaction of DSR, DSDV and Beeadhoc competitively decreases along the failed node axis as shown in Fig. 9.

This was the expected results because the failed nodes might have been actively involved in the routing of packets before they went down coupled with the nodes mobility which remained constant. This implies the network topology changing was affected by node failure only in this experiment.

AOMDV was the best performed algorithm in this regards and it maintains a very good throughput before it eventually decreases when the number of failed nodes increases to 25 and 30 respectively

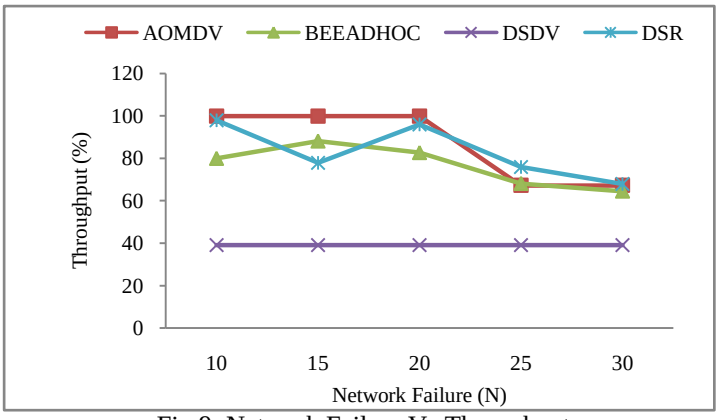

Fig.9: Network Failure Vs Throughput

4.4. Varying Radio Wireless Transmitting Range

In this experiment the radio transmission range of the nodes was varied, which in turns varies their respective coverage in the simulated area. The varied transmission range was plotted against throughput and routing overhead as depicted in Fig. 10 and 11 respectively.

It was expected that the throughput would be poor if the nodes transmission range could not allow them exchange data as the case with when the transmission range was made 100m as shown in Fig. 10, and that the effect of the range would not have any significant impact on the throughput once the nodes could communicate with each other. This was evident from the outcome of Fig. 10; in this, all the algorithms converged at 0% throughput when the nodes transmission range did not establish a connection between them and a drastic positive improvement recorded immediately connection was established and this was constant afterwards on the average for all protocols.

The outcome shown in Fig. 11 was the routing overhead against nodes radio transmission range. As expected, at smaller coverage area more hops would be required to get to the destination which was randomly selected and also exhibiting random movement within the simulated area.

This directly implies that more routing control packets would be required at smaller coverage radius which is expected to reduce as the coverage radius of nodes increases. This assumption was true of Beeadhoc, DSR displayed a fluctuating behaviour while AOMDV obeyed the assumption partly.

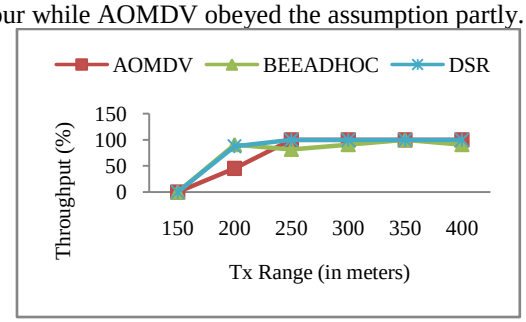

Fig.10: Radio Transmission Range Vs Throughput

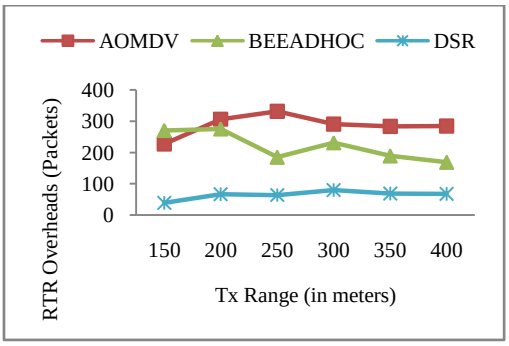

Fig.11: Radio Transmission Range Vs Routing Overhead

V. CONCLUSION

In this project, Beeadhoc routing algorithm has been implemented in network simulator NS-2.34 for a mobile Ad Hoc Network (MANet). Comparisons were made with other state-of-the-art routing algorithms, varying different features which include radio transmission range, number of nodes, nodes mobility speeds and number of failed nodes in several of the simulation instances considered.

The metrics used in the evaluation and analysis of the performance of the algorithms were throughput, end-to-end delay and routing overheads. Optimal data collection from a network using BeeAdhoc Routing Algorithm was successfully implemented and inferences from the experimental results indicate that: BeeAdhoc algorithm generated greater *throughput* than 2 of the 4 existing algorithms (DSDV) when the nodes were increased, BeeAdhoc algorithm yielded reduced *overhead* than 2 of the 4 existing algorithms (AOMDV), when the nodes were increased, BeeAdhoc exhibited an increased *End-to-Enddelay* as the node number increased, BeeAdhoc experienced decrease in its *throughput* as the number of failed nodes increases but still performed better than DSDV.

Overall, the BeeAdhoc Routing performance was comparable to that of DSR and DSDV in *throughput* and *overhead* but worse in delay. From the results obtained in all simulated experiments, BeeAdhoc could be used for routing packets in Ad Hoc network whenever interest is on *throughput*, *delay* and *overheads*. This is because its performance based on those metrics was better and compete reasonably with other algorithms.

REFERENCES

[1] D. H. Wolpert, K. Tumer, An Introduction to Collective Intelligence, *Technical Report, NASA ARC-IC-99-63, NASA Ames Research Centre, 1999.*

[2] I. Kroo, Collectives and Complex Systems Design, *VKI Lecture Series on Optimisation Methods and Tools for Multicriteria/Multidisciplinary Design, 2004.*

[3] D. H. Wolpert, Collective Intelligence, *Computational Intelligence: The Experts Speak, Edited by D.B. Fogel and C. J. Robinson (IEEE), 2003.*

[4] H. A. Mohammed, H. K. Babak, A Distributed Probability Collectives Optimisation Method for Multicast in CDMA Wireless Data Networks, *Proc. 4th IEEE International Symposium on Wireless Communication Systems, Article* No. 4392414, pp. 617 – 621, *2007.*

[5] A. J. Kulkarni, K. Tai, Probability Collectives: A Distributed Optimisation Approach for Constrained Problems, *IEEE Congress on Evolutionary Computation (CEC), pp. 1 – 8, 2010.*

[6] A. J. Kulkarni, K. Tai, Probability Collectives: A Multi-Agent Approach for Solving Combinatorial Optimisation Problems, *Applications of Soft Computing, vol 10, no. 3, pp. 759 – 771, 2010.*

[7] A. J. Kulkarni, K. Tai, Probability Collectives for Decentralised, Distributed Optimisation: A Collective Intelligence Approach, *Proc. IEEE International Conference on Systems, Man, and Cybernetics, pp. 1271 – 1275, 2008.*

[8] D. Subramanian, P. Druschel, J. Chen, Ants and Reinforcement Learning: A Case Study in Routing in Dynamic Networks, *International Joint Conference on Artificial Intelligence (IJCAI), 1998.*

[9] J. Brownlee, Clever Algorithms – Natured Inspired Programming Recipes, 2011

[10] C. E. Perkins and P. Bhagwat, Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers, *ACM-SIGCOMM*, 1994.

[11] H. F. Wedde, M. Farooq, and Y. Zhang, BeeHive: An Efficient Fault-Tolerant Routing Algorithm Inspired by Honey Bee Behaviour, 2004

[12] H. F. Wedde, M. Farooq, T. Pannenbaecker, B. Vogel, C. Mueller, J. Meth and R. Jeruschkat, BeeAdHoc: An Energy Efficient Routing Algorithm for Mobile Ad Hoc Networks Inspired by Bee Behaviour, In: GECCO ACM, 2005.

[13] T. White, B. Pagurek, D. Deugo, Collective Intelligence and Priority in Networks, *IEA/AIE '02 Proceedings of the 15th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems: Developments in Applied Artificial Intelligence, 2002.*

[14] J. Kil-Woong, Meta-Heuristic Algorithms for Channel Scheduling Problem in Wireless Sensor Networks, *International Journal of Communication Systems, John Wiley and Sons, Ltd, 2011.*

[15] D. H. Wolpert, K. Tumer, J. Frank, Using Collective Intelligence to Route Internet Traffic, *Proceedings of the 1998 Conference on Advances in Neural Information Processing Systems II, 1999.*

[16] P. D. Maio, Digital Ecosystems, Collective Intelligence, Ontology and the 2nd Law of Thermodynamics, *2nd IEEE International Conference on Digital Ecosystems and Technologies (IEEE DEST 2008), pp 144 – 147, 2008.*

[17] J. A. Boyan, M. L. Littman, Packet Routing in Dynamically Changing Networks: A Reinforcement Learning Approach, *Advances in Neural Information Processing Systems vol. 6, pp. 671 – 678, 1994.*
[18] C. Chiang, H. Wu, W. Liu, M. Gerla, Routing in Clustered Multihop, Mobile Wireless Networks with Fading Channel,*1997.*
[19] G. S. Ryder, K. G. Ross, "A Probability Collectives Approach to Weighted Clustering Algorithms for Ad Hoc Networks, P*roc. 3rd IASTED International Conference on Communications and Computer Networks, pp. 94 – 99, 2005.*
[20] D. S, P. Volf, M. Pechoucek, N. Suri, D. Nicholson, D. Woodhouse, Optimisation-based Collision Avoidance for Cooperating Airplanes, *IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology-Workshops, 2009.*