Trinity College Dublin
Journal of Postgraduate Research

Published by the Graduate Students' Union of the University of Dublin,
Trinity College, in partnership with the Trinity Annual Fund.
This issue represents submissions from the 2008-2009 academic year.

Published by:
Brunswick Press Ltd.,
Unit B2,
Bluebell Ind. Est.,
Dublin 12,
Ireland.

ISBN: 0-9550547-2-9, 978-0-9550547-2-3

This issue of the TCD Journal of Postgraduate Research may be cited as: (2009) 8 JPR.

Printed in Ireland.

Ant Navigation-Based Adaptive Routing Technique for Mobile *Ad Hoc* Networks

Niaz Morshed Chowdhury,
Syed Murtoza Baker and Ershadhul H. Choudhury

Introduction

A Mobile *Ad Hoc* Network (MANET) is a collection of mobile nodes operated over wireless technologies such as IEEE 802.11 with or without having an infrastructure-based access point. MANETs are comprised of moving objects and create frequent changes in network topology. Moreover, due to the limited transmission range of wireless interface, MANET requires multiple hops to establish end-to-end communication between nodes. Protocols operating in the network layer of MANET are responsible for finding paths between two end points of a communication. Existing network layer protocols, both proactive and reactive, are not sufficient to handle frequent changes in network topology and show poor performances due to link failure.

Furthermore, the NP complex nature of *ad hoc* network topology makes it very difficult to find a perfect solution even for a single end-to-end communication. Difficulty increases when the number of hops between two end points becomes significantly large. In those cases obtaining a better solution without increasing bandwidth or packet overhead is almost impossible. All those shortcomings inherently increase the importance of an adaptive approach to be used in design of network protocols for MANETs.

In this paper, we focus on the design of an adaptive algorithm that works based on the principle of Ant Colony Optimization (ACO). The main contributions of this work are threefold: 1) we have provided an analytic design of different adaptive parameters required to facilitate ACO into a routing protocol, 2) we have demonstrated a whole routing technique in the form of an organized pseudo-code representation and 3) through simulation we have shown that the chance of identifying the best path is very high and requires a significantly shorter period. The proposed algorithm is a probabilistic adaptive technique that changes its routes with the change of network topology over time. Like other adaptive approaches, this algorithm learns from the environment and identifies appropriate paths using feedbacks of previously travelled packets. A self-made simulator implemented on C++ is used to evaluate the performance of this algorithm on the basis of diverse adaptive issues such as change of probability, growth of pheromone intensity, randomness of selection and packet sending rate through different paths.

Related Studies

It was the mid 90's when MANET started becoming popular[1] and demand for ubiquitous devices significantly increased. Hundreds of routing protocols arrived over the period of 1994 – 2004, though a large number of those protocols failed to survive due

to their weak proactive nature.[2] During this period, four specific protocols became very popular and established themselves as key elements for MANETs.

In 1994, Perkins proposed his first routing protocol "Dynamic Destination Sequenced Distance Vector (DSDV)".[3] Two years later, in 1996, Johnson proposed his famous *ad hoc* routing protocol "Dynamic Source Routing (DSR)."[4] In 2002, Beijar proposed another protocol "Zone Routing Protocol (ZRP)",[5] though it maintains few restrictions and also lacks in its generalized nature. Soon after, Perkins, working in the Nokia Research Centre, released his newly developed protocol "*Ad hoc* On-demand Distance Vector (AODV)".[6] AODV was accepted as a Request for Comments (RFC)[7] in July 2003, and since then this has been an era for DSR and AODV to dominate the ubiquitous and *ad hoc* network world.

Though DSR and AODV are currently the two most popular routing protocols, they still have many limitations. A major drawback of DSR is its packet overhead as it sends large control information with each packet. On the other hand, AODV broadcasts enormous control data or more specifically route request (RREQ) packets to its neighbouring nodes while establishing a route. Those RREQ packets increase its bandwidth overhead and result in poor performance.

A number of routing protocols have been proposed in recent years to improve performance of existing protocols, either by modifying their current mechanism or replacing them with a new set of approaches. One of the most noteworthy approaches recently used is the adaptive technique. In 2001, a routing protocol was proposed called Adaptive Distance Vector Routing,[8] that tries to convert conventional Distance Vector Routing into an adaptive self-organized protocol for MANETs. That same year, Adaptive Demand Driven Multicast[9] routing arrived to improve multicast facilities in *ad hoc* networks. There was another algorithm entitled Fuzzy Sighted Link State Algorithm,[10] a modified approach of the classical version used in wired networks. In July 2006, Masillamani *et al.* proposed a genetic algorithm (GA)-based routing protocol for conventional distance vector routing and they are currently working to convert this protocol to a GA-based adaptive AODV.[11]

Ant Colony Optimization

Ant Colony Optimization (ACO),[12] proposed by Dorigo in 1992, is an adaptive method to find partial solutions for the problems where identifying an exact solution is either difficult or impossible.[13] The basic idea of this approach is taken from the food searching behaviour of real ants in nature. When ants are searching for food, they initiate their journey from their nest and gradually move towards the food. While walking, ants deposit a special type of biological chemical called pheromone. The concentration of pheromone on a certain path is an indication of its usage. Ants follow the path where it is densely concentrated. When an ant reaches an intersection point of a path, it decides which branch to be taken next based on pheromone concentration of the paths. With time, concentration of pheromone on less travelled paths decreases due to diffusion effects, and eventually the best path is established.

In 1997, Dorigo, in his work "Ant Colony System: A comparative learning approach to the Travelling Salesman Problem",[14] hinted that network routing can be implemented based on an ant algorithm. The very next year he published another research work[15] where he discussed how packets and ants are typically identical for an adaptive system and a routing algorithm can be formed or improved based on this technique. In 1999, in another paper, "Ant Algorithm for Discrete Optimization"[16] he proposed the same idea. Despite the fact that he always recommended his algorithm to be used in computer networks, neither he nor other researchers showed enough interest in this regard.

Basic Framework of the Proposed Algorithm

Adjusting ACO into a routing protocol is a critical modification and requires extensive transformation in the basic model. Our proposed algorithm considers each packet as an ant and each connected path as a food searching track. This algorithm maintains two different types of packets namely 'forward packet and 'backward packet.' These are typically ants on the way to food and nest respectively.

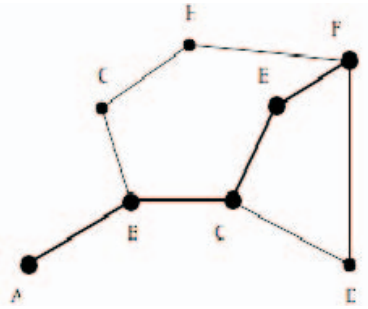

Figure 1. A network topology with travelled path in dark black. A-H are mobile nodes in the network. Each black line indicates a path between nodes. The dark path is the selected path to deliver a packet for a particular instance. Forward packet keeps track of this path and encapsulates this information into backward packet so that it can get back through this path and deposit pheromone on the tracks.

Forward packet is the original data packet and responsible for delivering data from the source to the destination. Each source node initiates a forward packet, encapsulates data and sends it to a specific destination. When a packet reaches a new node, it identifies its future path based on the selection parameter of that respective node. On the way to the destination, this packet keeps track of paths it has travelled for future use. Backward packet is an acknowledgement and responsible for depositing pheromone on the paths. For each forward packet, destination creates a corresponding backward packet and encapsulates the travelled path list into its header so that it can follow the track way back to the source and deposit pheromone on the paths.

Core contributions of the proposed algorithm lay in the design of different adaptive parameters. These parameters are the key elements in this algorithm that make diverse decisions at different phases. Success of the research is highly dependent on those parameters; hence it is imperative to design them carefully and effectively.

Mathematical Design of Adaptive Parameters

ACO is a generalized adaptive technique[17] like Fuzzy Logic,[18] Genetic Algorithm[19] or Simulated Annealing,[20] having different phases such as initialization, probability calculation, selection, pheromone deposition and pheromone evaporation. Each of those phases is problem-dependent[21] and needs to be designed based on the purpose and nature of specific problems. The following sub-sections will talk about the mathematical design of different adaptive parameters to facilitate ACO in a network routing protocol.

Initialization

A meta-heuristic method like ACO requires initialization at the beginning of the procedure.[22] Conventionally in ACO all the ant tracks are initialized with the numerical value zero as it is assumed that initially there is no pheromone on the tracks. But the proposed algorithm initializes its tracks or more appropriately routing paths in a different way, with a numerical value of one instead of zero. The rationale behind this modification is quite straightforward: in this algorithm a non-zero value of intensity represents a connection. Thus, pheromone intensity indicates connectivity of nodes, and paths having zero intensity indicate there is no connection between two particular nodes.

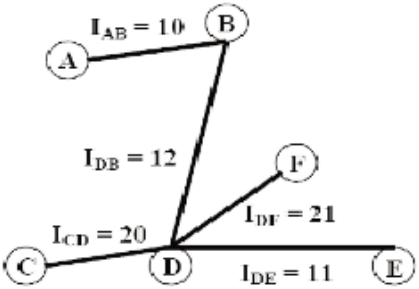

Figure 2. A network topology with a stable pheromone intensity of the paths.

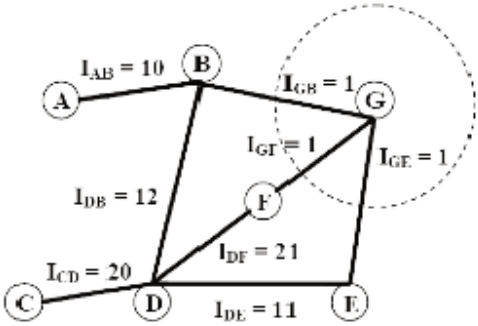

Figure 3. Changed topology when a new node G joined the network.

In Figure 2, a stable network topology has been shown where pheromone intensities of the paths are I_{AB}, I_{CD}, I_{DB}, I_{DF} and I_{DE}. Let us assume that a new node G has just joined the network (shown in Figure 3), hence creates three new paths GB, GF and GE. Conventional approach of ACO would initialize these three paths with zero intensity as those new paths have not been travelled by any ant (packet) yet. But, as per this proposed algorithm, initialization will be performed with unit values in order to indicate connections between B and G, F and G, and E and G.

Probability Calculation

Probability calculation is one of the most vital subjects for an adaptive algorithm. Success of the proposed technique is highly dependent on appropriate calculation of a particular probability called path probability. While sending a packet from one node to another, this algorithm determines the appropriate path based on this parameter. Path probability is further reliant on the calculation of pheromone intensity of different paths. In this proposed algorithm, path probability is defined as:

$$P_i = \frac{I_i}{I_t}$$

Equation 1. Path probability. Here, P_i is the probability of the i_{th} path connected with the neighbouring nodes whilst I_i and I_t are the intensity of i_{th} and $\sum I_{connected}$ (all connected) paths, respectively.

Selection

In this proposed algorithm, selection is maintained through the single spin roulette wheel method. Space allocation for each path on the wheel is proportional to its probability. For example, if there are three paths A, B, and C connected with a node having probability PA=0.50, PB=0.25 and PC=0.25, arrangement on the wheel for this particular case would be as in figure 4:

Figure 4. Roulette Wheel arrangement for PA=0.50, PB=0.25 and PC=0.25.

Another important issue in selection is pseudo random number generation. Weakness in random number generation may create an initial fluctuation problem,[23] resulting in unexpected performance by the algorithm. Most of the compilers in practice use current time (in second) as seeds. As a result, built-in random number generator func-

tions return a close value if they have been called within a short duration of time (*e.g.* 1 millisecond interval). A method for well-distributed random number generation has also been proposed with this algorithm. The following function takes a value as a parameter and returns a well distributed random number between zero and value.

Random Number Generator Function

```
Int random(int highest)
{
  srand((unsigned)time(0));
  int random_integer;
  int lowest= 0;
  int range= (highest-lowest) + 1;
  random_integer= lowest + int(range*rand()/(RAND_MAX + 1.0));
  return random_integer;
}
```

Pheromone Deposition

In this proposed algorithm, pheromone deposition is performed by backward packets. After a successful delivery, each backward packet increases the intensity of the path that forward packet or typically data packet has travelled to deliver data from the source to the destination. This process is formulated as follows:

$$I_{new} = I_{old} + \frac{Q}{Length\_Ratio}$$

Equation 2. Pheromone deposition formula. Here, Q is a unit quantity of pheromone to be deposited on the paths.

But the most notable point in pheromone deposition is that not all paths will get the full portion of Q, rather, deposition of Q will be inversely proportional to the travelled path. Length_Ratio varies from node to node. For each node on the path, it is measured as the path length between the source and the calculated node.

The rationale behind this design approach lay in quick identification of best paths. In this technique, backward packet will deposit more pheromone on the shorter path compared to the longer path while coming back from the destination. As probability calculation is dependent of the pheromone density, it ultimately increases the chance of the best path to be selected.

A pictorial example is shown below to describe the above literature. Consider the following network topology shown in figures 5 and 6. In figure 5, it shows that A has made a successful delivery of a packet to C. During this travel, this packet marked the paths that it has travelled.

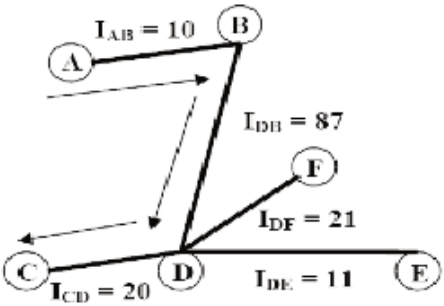

Figure 5. A is sending a packet to C.

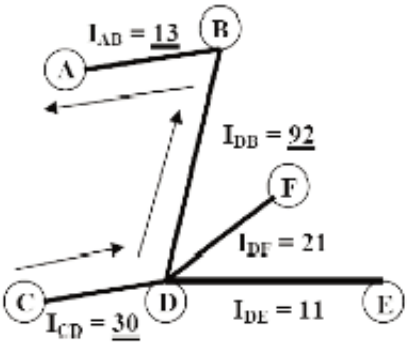

Figure 6. Intensity of the paths is being changed by the backward packet.

Let us consider $Q = 10$. Based on Equation 2, pheromone of each path will be updated as follows:

$$I_{AB(new)} = 10 + \frac{10}{3} = 13$$

$$I_{DB(new)} = 87 + \frac{10}{2} = 92$$

$$I_{CD(new)} = 20 + \frac{10}{1} = 30$$

Figure 5 shows backward packet sent by C to A. This backward packet will increase the intensity of the paths (underlined) that it has travelled while coming from A to C.

Pheromone Evaporation

Pheromone evaporation is another tricky challenge in the proposed algorithm. It decreases pheromone from less travelled paths. In this algorithm, pheromone evapo-

ration is computed based on a parameter called evaporation parameter, λ. Value of λ depends on the traffic density of the network. If intensity of any path at a junction crosses the value of λ, intensity for all the paths of that junction will be updated as follows:

$$I_{i(new)} = \frac{I_{i(old)}}{GCD(I_{1(old)}, I_{2(old)}, \ldots, I_{k(old)})}$$

Equation 3. Pheromone evaporation formula

Here, $I_{i(new)}$ and $I_{i(old)}$ are the intensity of i_{th} path before and after the evaporation respectively and k is the number of connected paths with that junction. GCD is the abbreviation of Greatest Common Divisor.

Operation

In general, a routing methodology needs to perform three basic operations, namely route establishment, route discovery and route maintenance.[24] These are the three most important and key tasks for any routing technique. Nature and performance of a particular routing algorithm is determined by how it deals with these three tasks. Our proposed algorithm is no different and follows these basic operations while operating in a network.

At the very beginning of the network, or when a new node enters into the network, it finds neighbouring nodes and initializes local intensity value of the corresponding paths with unit cost. Initially it also assumes that all the nodes in that network are reachable through any of those connected paths. It maintains a vector table like distance vector routing where it keeps intensity of different paths. This table is arranged in the conventional form of 'from A to B via C.'[25] But instead of distance value, it holds the intensity of a particular link. As time passes, the best outgoing link for a particular node gradually becomes stable in this intensity table.

In this proposed technique, nodes do not send route request like AODV or broadcast control information like DSR, though it still maintains a reactive on-demand nature. Before sending any packet to a particular destination, it finds the appropriate path based on a probabilistic selection mentioned earlier. Paths having been travelled most to reach a destination will receive higher priority in the selection. This route discovery is an adaptive method and eventually becomes stable by learning its environment over a period of time.

There might be a case when a node leaves the network. In such a situation, no packet can be delivered successfully through the paths that go over that particular node. As a result of this failure, pheromones of that path will be evaporated and other paths will start getting priority in selection. If the leaving node returns to the network, this path will be established again. Otherwise, due to periodical pheromone evaporation, it will become a dead link and be removed permanently.

Pseudo-Representation of the Proposed Algorithm

This pseudo-code (see Appendix) is a virtual representation of the proposed algorithm that works on each node of the network. Execution of this proposed algorithm is also associated with a set of control variables, which are unique for each packet. These variables hold information regarding status and route of different packets and are being encapsulated within the packet header. Table 1 describes detail about necessary control variables.

Name	Size (bit)	Description
packet_status	1	Keeps track of forward and backward packet. 1 and 0 are the value for forard and backward packet, respectively.
Delivery_status	1	Keeps track of whether a packet has been delivered successfully or not. 1 for successful delivery, otherwise 0.
Total_length	32	Stores number of hops travelled from the source to the destination (total length of the path).
Path_length	32	Stores number of hops traveled from the source on the way to the destination with respect to current node.
Path	32 x n	This is a list storing address of the traveled path by a specific packet. Length of this field will be determined by the path_length.

Table 1. Control variables.

When a packet arrives, the adaptive_routing() procedure is being called and handed over the responsibility to perform the required job. This root procedure gradually makes decisions and calls necessary sub-procedures to perform further actions. It extracts control variable from the packet headers and takes one of the following actions:

Action A

If the current node is found as the destination, it receives the packet, extracts control variables, constructs a backward packet, assigns packet_status as 'backward' and delivery_status as 'successful' and sends this packet back to the immediate last node from where it has arrived.

Action B

If the current node itself is the source and the packet is a backward packet with delivery status successful, it indicates end of the journey for that particular packet. On

receiving such packets, receiver node destroys them and deposits pheromone on the proper path.

Action C
If it is identified that the current node address already exists in the path list of the packet, it indicates this packet has already been forwarded through this node, *i.e.,* it is creating a cycle. In such a situation, this packet will be sent back to the previous node by changing its packet_status to 'backward' and delivery_status to 'failed.' This special approach also solves the 'Count to Infinity' problem[8] that occurs in the conventional Distance Vector Routing.

Action D
If none of the above situations match with a received packet, root procedure calls next_router_selection() procedure to select next node. This procedure classifies each packet as one of the followings:

Class A: A forward packet whose destination is reachable from the current node.
Class B: A backward packet with successful delivery
Class C: A backward packet with delivery failed
Class D: No more paths to go

For a forward packet whose destination is reachable from the current node, this procedure finds probabilities for each outgoing path excluding the one this packet has already used to arrive. Based on calculated probabilities and with the help of the decide_next_router() procedure, it finds the next appropriate node for the considering packet. Then it includes the next node address in the path, increases path length and returns selected next node address. For a backward packet with successful delivery, it selects the last address of the path as next node, discards this address from the path, decreases path length and deposits pheromone on the path through which this packet has arrived. For a backward packet with delivery status 'failed,' it calculates corrected probability and sends the packet accordingly. If there is no path left to send the packet, it changes packet status to 'backward' and delivery status to 'failed' and sends back this packet to the previous node.
Other procedures, such as calculate_probability(), decide_next_router(), deposite_pheromone() and evaporate_pheromone() work as described earlier in the mathematical design section. These are mainly responsible for incorporating ACO into a routing algorithm.
The pseudo code for this algorithm is presented in the Appendix to this paper.

Results and Evaluation
A self-made simulator implemented on C++ is used to evaluate performance and behaviour of the proposed algorithm. This simulation mainly concentrates on meas-

uring and evaluating adaptive natures of the algorithm. A 50-node scenario is considered, taking two nodes as source and destination. Three different and complex paths are available to reach the destination from the source node. Among the paths, path 2 is the best one while path 3 and path 1 are the second and third best, respectively. Measurements of the following four issues have been taken into account to evaluate the proposed algorithm:

- Change of probability
- Growth of pheromone intensity
- Randomness of the selection
- Packet sending performance

Change of Probability

Change of probability is measured over the period of 900 packet send time from the source node to the destination. Initially, when the very first packet was sent, probability of each path was equal, *i.e.* 0.33. As time increased, gradually the best path was established and the remaining two almost reduced to zero. The main rationale behind this behaviour is implicated in the special pheromone deposition mechanism by the backward packet on its travelled paths. Backward packet deposits more pheromone on the shortest paths compared to other paths, which are relatively larger. As intensity of a path is proportional to its selection probability, eventually the probability of getting selected for best path also increases. Figure 7 shows change of probability of different paths within an interval of 100 packet send period.

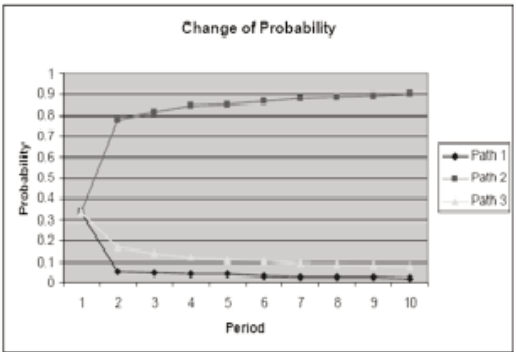

Figure 7. Change of probability in the test topology

Growth of Pheromone Intensity

In this phase of evaluation, again 900 packets were sent. Measurement was taken within an interval of each 100 packet send time. Initially, pheromone intensity of all three paths was 1. Figure 8 shows the change of intensity due to pheromone deposition on the experimental paths. The result clearly demonstrates that the best path received more pheromone deposition and its intensity increased drastically. Initially, intensity of the other two paths also increased but gradually their rate reduced to zero, which also lowered their chance of being selected.

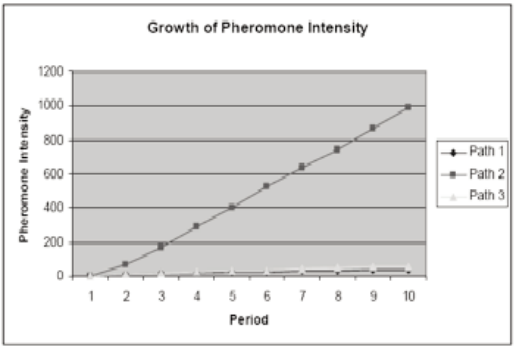

Figure 8. Growth of pheromone intensity

Randomness of the Selection

Randomness of the selection has been measured for the same network topology over five different runs. Instead of using default functions, a new well-distributed random number generator is proposed with this work. This random number generator is responsible for maintaining a stable selection for this algorithm. Figure 9 shows the result where it is clearly noticeable that for five different experiments, variation of path selection is very marginal.

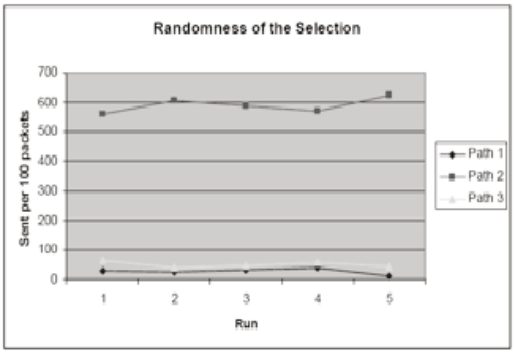

Figure 9. Randomness of the selection

Packet Sending Performance

Although change in selection probability, intensity of different paths or stability of selection function are key elements of this algorithm, in terms of evaluation they are subordinate results. Ultimate success of this algorithm mainly depends on how effectively and efficiently it uses best paths to send packets from the source node to the destination. In this final phase of evaluation, the total number of packets sent through each path is measured for the same topology discussed earlier. Like previous experiments, in this phase measurement is taken over the period of 100 packet send time. Figure 10 clearly shows that as time increases, more packets are being sent through the best path *i.e.* path 2. More than 90% of packets are delivered through this path whilst the other two paths receive less than 10% of packets together. It also proves

success of the proposed algorithm and its proper facilitation with Ant Colony Optimization.

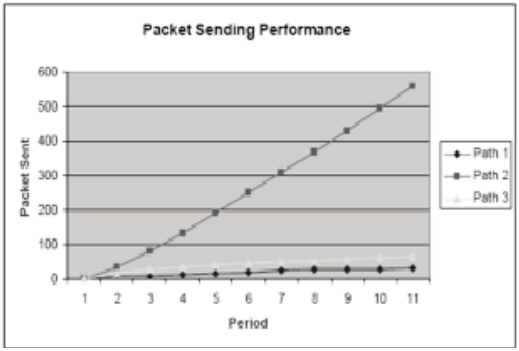

Figure 10. Packet sending performance

Conclusion and Future Work

In this work we proposed an adaptive routing protocol for MANETs. This protocol incorporates the principle of Ant Colony Optimization to discover routes from the source node to the destination. Design of different adaptive parameters was the main concern of this work, although a complete overview of the proposed technique was also presented. We used a self-made simulator implemented on C++ as the test bed to evaluate and observe performance of different parameters.

The evaluation section clearly shows that probability of suitable paths quickly converge towards a higher value which eventually makes the sender select best paths in most occasions. This special nature of the technique reduces bandwidth overhead of the network by not sending frequent control information. It has the intelligence to decide the suitable path depending on its parameters. Through the final part of the evaluation we have shown that most of the packets always go through the best paths.

This work mainly deals with design issues, but its success on self-made simulator also indicates its potential to provide adequate services for wireless networks. Our future objective is to implement this algorithm for MANETs using OPNET[26] and compare its performances with AODV and DSR on simulator as well as on real world.

NOTES

[1] Larry L. Peterson and Bruce S. Davie, *Computer Networks – A Systems Approach* (San Francisco, Morgan Kaufmamm Publishers Inc., 2007)

[2] Dimitri Bertsekas and Robert Gallager, *Data Networks* (Prentice Hall, New Jersey, 1992); Scott Corson *et al.*, "An Internet MANET Encapsulation Protocol (IMEP) Specification," Internet Draft, draft-ietf-manet-imepspec01.txt, August 1998. Work in progress; Santashil Palchaudhuri and Sridhar Lavu, "A Performance Comparison of *Ad Hoc* Networks Routing Protocols," Scientific Commons, http://en.scientific-commons.org/42711255

[3] Charles E. Perkins and Pravin Bhagwat, "Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for mobile Computers", in *Proceedings of SIG-COM '94 Conference on Communications Architecture Protocols and Applications*, 234-244

[4] Josh Broch *et al.*, "A performance Comparison of Multi-hop Wireless *Ad hoc* Network Routing Protocols," Mobicom '98, Dallas Texas, 25-30 October, 1998; David A. Maltz and David B. Johnson, "Protocols for adaptive wireless and mobile computing," *IEEE Personal Communications* 3 (1996); David A. Maltz, David B. Johnson, "Dynamic source routing in ad hoc wireless networks," in *Mobile Computing*, ed. Tomasz Imielinski and Henry F. Korth (Kluwer Academic Publishers, 1996), 153-181

[5] Zygmunt J. Haas and Marc R. Pearlman, "The Zone Routing Protocol (ZRP) for Ad Hoc Networks," Internet Draft, draft-ietf-manet-zone-zrp-01.txt, August 1998. Work in progress.

[6] Charles E. Perkins, "Ad hoc On Demand Distance Vector (AODV) Routing," Internet Draft, draft-ietfmanet-aodv-01.txt, August 1998. Work in progress.

[7] Charles E. Perkins. Ad Hoc On-Demand Distance Vector Routing (RFC - 3561)

[8] http://www.comp.brad.ac.uk/~sburuha1/routingprot.htm (accessed Jul 25, 2008)

[9] Elizabeth M. Royer, Chai-Keong Toh, "A Review of Current Routing Protocols for Ad Hoc Mobile Wireless Networks," *IEEE Personal Communications* 6 (1999): 46-55

[10] *Ibid.*

[11] M. Roberts Masillamani *et al.*, "Genetic Algorithm for Distance Vector Routing Technique," *AIML Journal* 6 (2006): 59-62

[12] Marco Dorigo, "Optimization, learning and natural algorithms," (Unpublished PhD dissertation, Politecnico di Milano, Dipartimento di Elettronica, Italy, 1992); M. Dorigo, V. Maniezzo, and A. Colorni, "Positive feedback as a search strategy," Tech. Rep. 91-016. Milan, Italy: Politecnico di Milano, Dipartimento di Elettronica, 1991

[13] E. Bonabeau, M. Dorigo, and G. Theraulaz, *From natural to artificial swarm intelligence* (New York: Oxford University Press, 1999); D. Corne, M. Dorigo and F. Glover, eds., *New ideas in optimization* (Maidenhead, UK: McGraw-Hill, 1999)

[14] M. Dorigo and L. M. Gambardella, "Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem," *IEEE Transactions on Evolutionary Computations* 1 (1997): 53-66

[15] Gianni Di Caro and Marco Dorigo, "AntNet: Distributed Stigmergetic Control for

Communications Networks," *Journal of Artificial Intelligence Research* 9 (1998): 317-365

[16] M. Dorigo, G. Di Caro and L. M. Gambardella, "Ant Algorithms for Discrete Optimization," *Artificial Life* 5 (1999): 137-172

[17] D. B. Fogel, *Evolutionary computation: toward a new philosophy of machine intelligence* (New York : IEEE Press, 1995); J. H. Holland, *Adaptation in natural and artificial systems* (Cambridge, Mass.: MIT Press, 1992); I. Rechenberg, *Evolutionsstrategie* (Stuttgart, Germany: Frommann-Holzboog, 1973); H. P. Schwefel, *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie* (Basel, Switzerland: Birkauser, 1977)

[18] R. Krishnapuram, H. Frigui and O. Nasraoui, "Fuzzy and Possibilistic Shell Clustering Algorithms and Their Application to Boundary Detection and Surface Approximation. I," *IEEE Transactions on Fuzzy Systems* 3 (1995): 29-43; Patrick K. Simpson, "Fuzzy Min-Max Neural Networks - Part 2: Clustering", *IEEE Transactions on Fuzzy Systems* 1 (1993): 32-45

[19] Masillamani, "Genetic Algorithm"

[20] H. L. Tan, S. B. Gelfand and E. J. Delp, "A Cost Minimization Approach to Edge Detection Using Simulated Annealing," *IEEE Transaction on Pattern Analysis and Machine Intelligence* 14 (1992): 3-18

[21] Dorigo, "Discrete Optimization"

[22] J. M. Pasteels, J. L. Deneubourg and S. Goss, "Self-organization mechanisms in ant societies. I: Trail recruitment to newly discovered food sources," *Experientia Supplementum* 54 (1987): 155–175

[23] Dorigo, "Discrete Optimization"

[24] Royer, "Routing Protocols"

[25] *Ibid.*

[26] www.opnet.com (accessed Jul 25, 2008)

Appendix

```
Procedure adaptive_routing()
   source = extract_source()
   destination = extract_destination()
   previous = extract_previous()
   total_length = extract_total_length()
   path_length = extract_path_length()
   Path = extract_path()
   packet_status = extract_packet_status()
   delivery_status = extract_delivery_status()
```

```
   if(destination == current_router)
      packet_status = backward
      delivery_status = successful
      construct_backward_packet(packet_status, deliv-
ery_status, Path, path_length)
      exchange_packet()
      next = previous

   else if(source = current_router & packet_status =
backward & delivery_status = successfull)
      length_ratio = total_length - path_length + 1
      deposit_pheromone(current_router, previous,
length_ratio)
      if(I_previous > λ)
         evaporate_pheromone()
      end_of_the_journey()

   else if(current_router ∈ Path)
      packet_status = backward
      delivery_status = failed
      next = previous

   else
      next = next_router_selection(source, previous,
destination, Path, packet_status, delivery_status)
   update_header(packet_status, delivery_status,
total_length, path_length, Path)
   send_packet(next)
end procedure

procedure next_router_selection(source, previous, desti-
nation, Path, path_length,packet_status,delivery_status)

   A = load_local_routing_table()
   if (packet_status = forward & E_current router >1)
      P = calculate_probability(A, previous, destina-
tion)
      next_router = decide_next_router(P)
      Path = Path U {current_router}
      path_length = path_length + 1
```

```
   else if (packet_status = backward & delivery_status
= successful)
       length_ratio = total_length - path_length + 1
       deposit_pheromone(current_router, previous,
length_ratio)
       if(I_previous > λ)
           evaporate_pheromone(current_router)
           Path = Path \ {previous_router}
           path_length = path_length - 1
           next_router = second_last_of_the_path(Path)

   else if (packet_status = backward & delivery_status
== failed)
       P = calculate_corrected_probability(A, previous,
destination, Path)
       if (P has at least one value)
           next_router = decide_next_router(P)
           Path = Path U {next_router}
           path_length = path_length + 1
           packet_status = forward
   else
       next_router = Path ∩ {outgoing_edges_current router}
   return next_router
end procedure

procedure calculate_probability(A, Source, Previous,
Destination)
   for all i in A except Previous
       Find I_t =  Σ I_i
   for all i in A except Previous
         Pi = I_i / I_t
    return P
end procedure

procedure decide_next_router(P)
   for all i in probability list P
   construct roulette_wheel based on the probability P
   selected = spin(roulette_wheel)
   return selected
end procedure
```

```
procedure deposite_pheromone(current_router, previous,
length_ratio)
    from the routing table A
    I_previous = I_previous + Q / length_ratio
end procedure

procedure evaporate_pheromone(current_router)
    for all link i in the current_router
    I_i(new) = I_i(old)/GCD (I_1(old), I_2(old), ... , I_k(old))
end procedure
```