International Journal of Engineering and Technology Vol. 1, No. 1, April, 2009
1793-8236

HPRAAM: Hybrid Parallel Routing Algorithm using Ant Agents for MANETs

Ramkumar. KR, Ravichandran, GaneshKumar. M, Hemachandar. N, ManojPrasadh. D

***Abstract*—Mobile Ad Hoc Networks (MANETs) are communication networks in which all nodes are mobiles and they communicate with each other via wireless connections. All nodes are equal and there is no central control or overview. There is no fixed infrastructure for them to interact, so without relying on any existing, preconfigured network infrastructure or centralized control, MANET aims for parallel data transformation, and guaranteed data transaction. Due to the high mobility of the nodes, the topology of the network changes constantly, and hence their paths change. For a solution to the above problem we have devised HPRAAM a novel framework for MANETS which aim for parallel data transformation, and guaranteed data transaction. HPRAAM uses Ant agents to find the optimal path between the source and the destination. Our algorithm also finds the optimal load balancing routes between the two ends. Adding to this HPRAAM tries to reduce the total number of ants in the system. we have also proposed payload with ants so that the data reaches the destination, even in case of a route failure.**

***Index Terms*— Ant Colony Optimization; MANETS; Routing; Mobile Agents**

I. INTRODUCTION

A substantial research effort has gone into the development of routing algorithms for MANETs. A number of routing algorithms have been proposed. Some of these are DSDV, OLSR, AODV, DSR several other routing protocols Ant Net [8], ARA [3], AntHocNet [7] and PERA [6], [13], [14]. These protocols can generally be categorized as either *proactive* or *reactive* protocols. Proactive protocols build routes in the network constantly, even though there might not be packets to be transmitted between a certain set of nodes. Reactive (on-demand) protocols, on the other hand, attempt to establish multi-hop between pairs of nodes only when there are packets to be exchanged between these pairs of nodes. The nodes then forward data stochastically based on Pheromone values calculated from backward ants. [**AntHocNet**] uses equal pheromone values in initially and is updated by backward ants. In path maintenance phase the ants' exploratory behavior is limited around the Current Optimal path. Link failures are found through loss of neighbors or data packet transmission failure.

A. ACO routing

The basic idea behind ACO [1] algorithms for routing is the acquisition of routing information through path sampling using ant agents. These lightweight agents are generated concurrently and independently by the nodes, with the task to try out a path to an assigned destination. An ant going from source *s* to destination *d* collects Information about the quality of the path (e.g. end-to-end delay and number of hops), and, retracing its way back from *d* to *s*, uses this information to update the routing tables at intermediate nodes. Ants always sample complete paths. [2]The routing tables, called pheromone *tables*, contain for each destination vector of real-valued entries, one for each known neighbor node. These entries, the pheromone variables, are a measure of the goodness of going over that neighbor on the way to the destination. They are continually updated according to the quality of the paths sampled by the ants. The repeated and concurrent generation of ants results in the availability at each node of a bundle of paths, each with an estimated. While moving on the path, an ant evaluates this solution and deposits pheromone on its way. This pheromone trail will be used by the future ants to make a routing decision [4].

In this paper, we make a number of contributions to the design of parallel data transmission and path updating while transferring the data. We focus on the guaranteed data transaction in through acknowledgement in mobile ad hoc routing protocols. The rest of the paper is organized as follows. In Section II, we discuss the different types of forward ants and backward ants which ensure the parallel and guaranteed data transaction. Next In Section III we present our proposed algorithms to implement the above specified concept.

II. ROUTE DISCOVERY

A. Forward ant

When a source needs to send a packet to a particular node, it first checks the cache for existing routes. When no routes are known, it broadcasts Forward Request ANTS which are propagated through the network till it reaches the destination [1] [7]. This process can be compared to ants initially spreading out in all directions in search of food source. A forward ant at each intermediate node selects the next hop using the information stored in the routing table of that node or by rebroadcast. The next node is selected with a probability proportional to the goodness of that node which is measured by the amount of pheromone deposited on the link to that node. When a forward ant reaches the destination, it generates a backward ant which takes the same path as the corresponding forward ant but in opposite direction. The backward ant updates pheromone values as it moves on its way to the source node s.

Here F flag will decide the nature of the ant for the value of F=1 it will behave as a Forward ant.

Authrorrs are with Sri Venkateswara College of Engineering

B. *Forward Ant with Payload*

Here forward ant is attached with the payload to discover the new routes whenever required and the AntID is used to avoid the duplicate packets and to avoid the traffic congestion. Here P=1 represents the payload

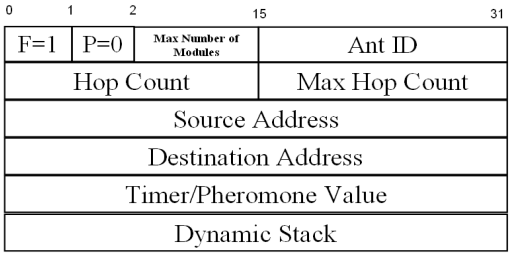

Fig. 1. Forward Ant

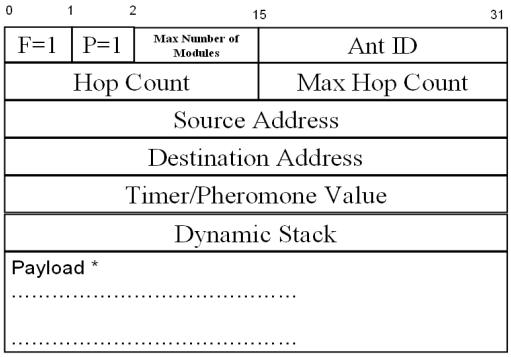

Fig 2. Forward Ant with Payload

C. *Backward Ant*

Upon arrival at the destination d, it is converted into a backward ant, which travels back to the source retracing the path. At each intermediate node i, coming from neighbor n, the ant updates the entry i T_{nd}^{i} which is the pheromone value indicating the estimated goodness of going from i over neighbor n to reach the destination d, in the i's pheromone table. The way the entry is updated depends on the path quality metrics used to define pheromone variables. For instance, if the pheromone is expressed using the number of hops as a measure of goodness, at each hop the backward ant increments an internal hop counter and uses the inverse of this value to locally assign the value τ_d^i which is used to update the pheromone variable T_{nd}^{i} [7]

$$T_{nd}^{i} = \gamma T_{nd}^{i} + (1-\gamma)\tau_d^i \gamma \epsilon [0,1] \qquad (1)$$

D. *Backward Ant structure*

For F = 0 represents BackwardAnt and P bit represents whether the ant has acknowledgment. P = 1 denotes that AntID is the acknowledgment.

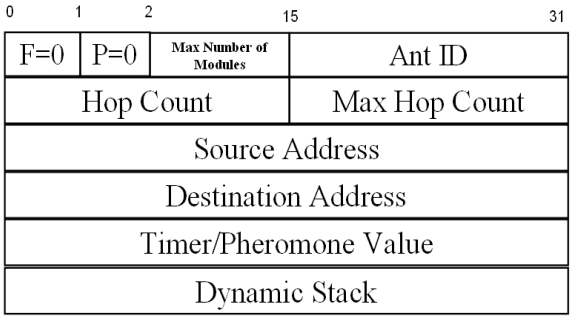

Fig 3. Backward Ant

E. *Route maintenance*

During the course of a communication session, source node s periodically sends out proactive forward ants to update the information about currently used paths and try to find new and better paths. However, the received proactive forward ant's paths are compared with the already stored paths by the destination d for achieving node disjointness [7], [9].The destination d sends proactive backward ants only for the ants with the disjoint paths and others are ignored. The proactive ants follow pheromone and update pheromone tables in the same way as reactive forward ants. Such continuous proactive sampling of paths is the typical mode of operation in ACO routing algorithms.

III. LOAD BALANCING

Load balancing deals with improving the performance of the system by transferring the jobs from overloaded nodes to under loaded or idle nodes. By doing so, the total time to processes all jobs may reduce considerably and also makes it sure that no node sits idle while some jobs are waiting to be processed. Routing protocols are vital for the proper functioning of ad hoc networks. A major drawback of all existing adhoc routing protocols is that they do not have provisions for conveying the load and/or quality of a path during route setup. Hence they cannot balance the load on different routes. Also, both proactive and reactive protocols chose a route based on the metric, the smallest number of hops to the destination. But it may not be the most significant route when there is congestion or bottleneck in the network. It may cause the packet drop rate, packet end-to-end delay, or routing overhead to be increased particularly in the cases when the traffic is concentrated on a special node like a gateway through which mobile nodes from ad hoc network can connect to Internet.

There are various proposed algorithms for load balancing that consider traffic load as a route selector, but these algorithms neither reflect burst traffic nor transient congestion [10]. Due to multi path routing, even if one path fails, data can still be routed to the destination using the other routes. Thus, the cost of discovering new path can be salvaged [11]. While selecting the path set the following issues need due consideration [12].The distribution of load should be even. Mobile nodes with lower traffic load should

be preferred to the heavily loaded mobile nodes. The traffic load in the medium surrounding the mobile nodes on the routes, should be light. If a link is highly reliable, it is advantageous to allow it to be shared by more than one path. Multipath routing will select multiple paths to disperse the data to the target at the same time the forward ant will travel along with the payload to discover new root and backward ant can be used for acknowledgment.

IV. Node Level Algorithm for Forward Ant

A. Route Discovery:

In route discovery process, when we do not have routing information we broadcast the ForwardAnts without payload to all neighbors. The intermediate nodes on receiving the forward ant broadcast the ants in turn if HopCount is less than MaxHopCount. If the forward ant has reached the destination then, the ant is converted into BackwardAnt and unicast by using the path information on the dynamic stack.

```
if
{ {F==1} and {P==0}  and
  Destination Address != NodeIP} and
  HopCount<MaxHopCount  }

{
   BroadCast(ForwardAnt)
   }

else
if{Destination Address == NodeIP}
   and {F==1} and {P==0}
{
    ConvertToBackwardAnt(ForwardAnt)
       Unicast(BackwardAnt)
}
```

B. Ant Conversion

We use the same forward ant packet to create the backward ant by swapping source and destination address and then by setting MaxHopCount to CurrentHopCount of the forward ant received. This makes sure that the ant does not travel through more hops than the hops it took to reach the destination. The current hop count is set to zero.and F bit is set to 0 to denote backward ant and P bit is set to 0 or 1 based on whether it carries Acknowledgment

```
ConvertToBackwardAnt(ForwardAnt)
{
       Swap Destination and Source Address
       Copy HopCount into MaxHopCount
       Set Current Hop Count to Zero
       Change the Flag bits F=0 P=0/1
}
```

C. Forward Ant with Payload

This type of ants remains inactive and becomes active at intermediate node only when the data packet loses its route due to node mobility and helps in better data delivery. NOM represents the number of modules to be sent for the particular data transfer session. We divide the data to be sent into NOM modules and index them. The neighbors through whom we plan the parallel transfer are decided based on the THRSLPHEROMONE. Only the neighbors above this threshold are chosen for data transfer to make sure that the quality of path is not bad. The number of packet for each neighbor which pass this criteria is decided on the formula α *Ceil (PheromoneValue *10) [α=7] to make the more packets to be directed though the strongest path. Once the number of packets NUMOFPACKETS for each packet is decided, then we declare a queue for each neighbor above the THRSLPHEROMONE and keep track of all the indexes of the data modules allocated for each neighbor. Once the data is ready it's attached with ForwardAnt with F=1 and P=1 to denote that it's ForwardAnt with payload. The ForwardAnt also carries with it MaxNumModules representing the sequence number of the last payload. Then the whole payload is sent.Once we receive BackwardAnt with acknowledgement, that particular index is deleted from the queue. We also check the queue every MAXTIMEALLOWED to check that if the queue is empty. if queue is not empty then we resend the data corresponding to the all the indexes in the queue. on successful transfer of all packets the queue is deleted and ready for another transaction.

Forward Ant with Payload algorithm

```
NOM= PAYLOADSIZE/MODULESIZE

Index the Modules with AUTOINDEX

ArrangeinDescending(Neighbour's Pheromone
value)

Minimum THRSLPHEROMONE=VALUE [user given value]

NUMOFPACKETS= α *Ceil(PheromoneValue *10 )
[α=7]

Declare a Queue [NUMOFPACKETS+CONSTANT]

Queue[i]=AUTOINDEX.

AttachForwardAnt(Payload, ForwardAnt]

AntID=Queue[AUTOINDEX]

SourceAddress = Sending Node Address

Destination Address=Receiving Node Address

Set F=1, P=1

ForwardAnt.MaxNumModules = max{AUTOINDEX}

Acquire NOP packets from source and send it
through the neighbour

Wait for ACKID=AndID from neighbour

Delete Queue[ACKID]

If(Empty(Queue)==TRUE and
CurrentTime>=MAXTIMEALLOWED)
       Delete the Queue
       Start a new Transaction

else if (Empty(Queue )==FALSE and
CurrentTime<=MAXTIMEALLOWED)

Resend the modules identified from the
Queue[AUTOINDEX]
```

D. Data Processing at receiver side

On the receiver node, the datapayload dp (forward ant with payload) received is discarded if it's a duplicate else the Data and ForwardAnt are extracted from it. The Data is stored in the buffer sorted in the order of the sequence number. The buffer size will be decided based upon the

```
Receive Payload dp
Declare BufferVector[] Dynamic Buffer to Store
the Received Packets.
Set FirstPacket=0

If(dp.ForwardAnt.DesitnationAddress==CurrentN
odeIP)and(duplicate(dp)==TRUE)
      discard(dp)

else
if(dp.ForwardAnt.DestinationAddress==Current
   NodeIP)
{
   If(FirstPacket ==0)
   {
  Size=MaxNumModules;
  Define BufferVector[size]
  }
     else
  {
    First=1
    Do nothing;
  }

    Data = ExtractData(dp)

    ForwardAnt=ExtractForwardAnt(dp)

    BackwardAnt=ConvertToBackwardAnt
   (ForwardAnt)

    Set BackwardAnt.P=1

    BufferVector[AntID]=Data
  If(BackwardAnt.DestinationAddress is
    NotFound in NodeTable)
    {
        Generate(ForwardAnt)to explore  the
    new Route to the source
        Broadcast(ForwardAnt)
    }
    else
    {
        Unicast(BackwardAnt)}

if { F== 0 } and { P==1 }  and
Destination Address != Node IP}
{
if(!UniCast(BackwardAnt)==Success)
BroadCast(ConvertToForwardAnt(BackwardAnt))

}
```

MaxNumModules received from the first packet, we immediately send the acknowledgement to the data received by converting the ForwardAnt extracted from dp into backward ant with the AntID representing the acknowledgement. This backward ant not only serves as acknowledgement but also to update the routing information. The backward ant path updating will help to strengthen the reliable path.

E. Node Level Algorithm (for BackwardAnt)

In case the backward ant loses the return path due to node mobility, then the ant is converted to ForwardAnt and is broadcasted to find the new route from the intermediate node instead of being discarded as in AntHocNet [7]. We have there by dealt with the better management of BackwardAnt.

V. CONCLUSION

In this study we proposed a modification to Ant Colony for distributing traffic load among nodes in ad hoc networks. The key concept of modification is to provide a metric for load distribution, the selection of light load path and the reduction of congested nodes in high load networks. We are in the process of implementing a simulation study and initial results shows a better performance and modifications can improve average throughput and reduce routing load by keeping track of the aggregate interface queue length. This work will create a new path towards the parallelism and acknowledged data transaction in mobile ad hoc networks.

REFERENCES

[1] M. G. Hinchey, R. Sterritt, and C. Rouff, "Swarms and Swarm Intelligence, " IEEE Computer, Vo. 40, No. 4, pp. 111-1 13, April 2007.

[2] K. M. Sim and W. H. Sun, "Ant Colony Optimization for Routing and Load-Balancing: Survey and New Directions, " IEEE Trans. on Systems, Man, and Cybernetics, Vol. 33, No, 5, Sep. 2003, pp. 560-572.

[3] M. Guine, , U. Sorges, and I. Bouazzi, "ARA-the ant-colony based routing algorithm for MANETs, " in Proc. of IWAHN 2002, pp. 79-85, August 2002.

[4] H. Matsuo and K. Mori, "Accelerated Ants Routing in Dynamic Networks, " 2nd International Conf. on Software Engineering, Artificial Intelligence, Networking & Parallel/Distributed Computing, pp. 333-339, 2001.

[5] S. Marwaha, C. K. Tham, and D. Srinivasan, "Mobile agents based routing protocol for mobile ad hoc networks, " in Proc. Of IEEE ICON, pp. 27-30, August 2002.

[6] J. S. Baras and H. Mehta, "A Probabilistic Emergent Routing Algorithm for Mobile Ad Hoc Networks, " in Proc. of WiOptO3, 2003.

[7] G. Di Caro, F. Ducatelle, and L. M. Gambardella, "AntHocNet: An Adaptive Nature-Inspired Algorithm for Routing in Mobile Ad Hoc Networks, " Tech. Rep. No. IDSIA-27-04-2004, IDSIA/USI-SUPSI, Sep.2004.

[8] Gianni Di Caro, Marco Dorigo, "AntNet: Distributed Stigmergetic Control for Communications Networks" Journal of Artificial Intelligence Research 9 (1998) 317-365.

[9] T. Maekawa, et. al. "An Ant-based Routing Protocol using Unidirectional Links for Heterogeneous Mobile Ad-Hoc Networks, " in Proc. ofICWMC '06, pp. 43 - 43, July 2006. ISBN

[10] Abolhasan, M., T. Wysocki and E. Dutkiewicz, 2001. A Review of Current On-demand Routing Protocols: ICN 2001, LNCS 2094, pp: 186-195.

[11] Perkins, C.E. and P. Bhagwat, 1994. Highly Dynamic Destination Sequenced Distance Vector Routing for Mobile Computers: Proc. Of Computer Communication Rev., 1: 234-244.

[12] Murthy, S. and J.J. Garcia-Luna-Aceves, 1996. An Efficient Routing Protocol for Wireless Networks: ACM/Baltzer Mobile Networks and Applications, 1 (2): 183-197.

[13] D.B. Johnson and D.A. Maltz. "Dynamic Source Routing in Ad Hoc Wireless Networks", in Mobile Computing, T. Imielinski and H. Korth, eds., Kluwer Academic Publishers, Norwell, Mass., 11996, pp. 153-181.

[14] J. Broch, D. Maltz, D. Johnson, Y. Hu and J. Jetcheva, "A Performance Comparison of Multi- Hop Wireless Adhoc Network Routing Protocols", Carnegie Mellon MONARCH Project, October 1998, http://www.monarch.cs.cmnu.edu/.

Ramkumar K.R. Born in Madurai, Tamil Nadu, India at 1977, did the B.E. computer Science in the year of 1999 at Madurai Kamaraj University, Madurai, Tamil Nadu, India and then completed M.E. Computer Science in the year of 2007 at Sri Venkateswara College of Engineering, Anna University, India.

He worked as a Lecturer at Sri Venkateswara College of Engineering, Chennai, Tamil Nadu, India for seven years and now working as a Senior Lecturer in the same college. He has published many papers in international journals.His research interests are Swarm Intelligence, Mobile Ad Hoc Networks and Security issues in routing algorithms.

M. Ravichandran earned his Master's degree in Optical Communication from college of Engineering, Anna University, Chennai in 2002. He acquired his Bachelor's degree in Electronics and Communication Engineering, from Sri Venkateswara College of Engineering, Sriperumbudur in 2000.

He is a lecturer of Computer Science and Engineering at Sri Venkateswara College of Engineering. He has over 6 years of teaching experience where he has guided many Undergraduate and Post graduate research projects. He has published many research papers and journals.

His research areas include Mobile Computing. High Speed Communication Networks, intelligent System and Digital Design. He is a Member of ISTE and IETE.

Hemachandar. N Born on December 9 1987, in Chennai, India. Is a under graduate student in the Department of Computer Science and Engineering Department at the Sri Venkateswara College of Engineering. His research interests are in Ad Hoc Mobile Networks and Swarm intelligence. He has published many research papers and journals.

Manojprasadh. D Born on November 8 1987 in Chennai, India. Is a under graduate student in the Department of Computer Science and Engineering Department at the Sri Venkateswara College of Engineering. His research interests are in Ad Hoc Mobile Networks and Swarm intelligence. He has published many research papers and journals He is a member of IEEE.

GaneshKumar. M Born on June 18, 1988, in Chennai, India. Is a under graduate student in the Department of Computer Science and Engineering Department at the Sri Venkateswara College of Engineering. His research interests are in Networking, Peer to Peer Networks, Ad Hoc Mobile Networks and Swarm intelligence. He has published many research papers and journals. He is a member if the IACSIT an International scientific association.