An Efficient Bee-inspired Auto-configuration Algorithm for Mobile Ad Hoc Networks

Filomena de Santis
Dipartimento di Informatica
University of Salerno
84081 Fisciano (Salerno), Italy

ABSTRACT
The infrastructure-less and dynamic nature of mobile ad hoc networks (MANETs) requires the implementation of a new set of networking technologies in order to provide efficient end-to-end communication according to the principles of the standard TCP/IP suite. Routing and IP address auto-configuration are among the most challenging tasks in the ad hoc network domain. Swarm intelligence is a relatively new approach to problem solving that takes inspiration from the social behaviors of insects, such as ants and bees. Selforganization, decentralization, adaptivity, robustness, and scalability make swarm intelligence a successful design paradigm for routing and IP address distribution for MANETs. In this paper it is proposed *BeeAdHocAutoConf*, a new IP address allocation algorithm based on the bee metaphor. Both the protocol operation and the simulation experiments are presented showing that *BeeAdHocAutoConf* guarantees an even address distribution in large scale MANETs at the cost of low complexity, low communication overhead, and low latency with respect to other known algorithms. Eventually, future research suggestions are outlined with the aim to extend the use of swarm intelligence paradigms for the redefinition or modifications of each layer in the MANET TCP/IP suite.

General Terms
Wireless networking, combinatorial optimization, network control algorithms, non conventional computing.

Keywords
Mobile ad hoc network, routing algorithms, IP auto-configuration algorithms, swarm intelligence.

1. INTRODUCTION
A mobile ad-hoc network is a set of mobile nodes which communicate over radio and do not need any infrastructure. The limited transmission range of wireless interfaces makes the communication multi-hop. Nodes accomplish the functionality of hosts, as well as that of routers forwarding packets for other nodes, [1]. MANETs are very flexible and suitable for several situations and applications since they allow to establish temporary communication without preinstalled infrastructure. Remarkable installations for mobile ad-hoc networks are made in calamity and military areas; with the increasing diffusion of radio technologies, e.g., IEEE 802.11a and Bluetooth, many multimedia applications take also advantages from running over mobile ad hoc networks. MANETs suffer from a variety of problems: routing and IP (Internet Protocol) address auto-configuration are among the most challenging ones. In the literature many different approaches dealing with these problems do exist, even though there are not algorithms which fit in all cases.
In this paper a new approach for an auto-configuration algorithm based on swarm intelligence is presented. More precisely the proposal is inspired by a bee colony behavior involved with a food site search when simple individuals show great ability to solve a complex problem by cooperation. The interesting point is that the bees do not need any direct communication: they use a form of visual communication expressed by meaningful and differentiating *dances*. Such an event corresponds to the notion of *stigmergy*, that is the indirect communication of individuals through changes in the environment. Several algorithms based on bee colony behavior have been introduced in recent years to solve optimization problems in the domain of multi-hop ad hoc network routing; in none of them the auto-configuration problem was considered.
The remainder of this paper is organized as follows. In section 2 the basics of bee colony optimization meta-heuristic and *BeeAdHoc*, a well known routing algorithm derived from it, are presented. In section 3 the necessary literature about IP address allocation for MANET are briefly reviewed. In section 4 *BeeAdHocAutoConf*, the new proposal for the solution of the problem at hand is introduced. In section 5 some simulation results to validate the quality of the algorithm are discussed. Eventually, in section 6 conclusions and ideas for future works are drawn.

2. ROUTING WITH THE BEE MODEL
A challenging task in the MANET domain is the routing where a path between a source and its destination must be found, possibly in an efficient way. *Proactive* routing, *reactive* routing and *hybrid* routing, [2], are the most popular classes of MANET routing protocols. In a *proactive* routing protocols (e.g. Destination Sequenced Distance Vector, DSDV, [3]) nodes continuously evaluate routes towards all reachable destinations and maintain consistent, up-to-date routing information even though network topology changes occur. In a *reactive* routing protocol (e.g. Dynamic Source Routing, DSR, [4]), routing paths are searched only when needed by means of a route discovery operation established between the source and destination node. *Hybrid* routing protocols (e.g. Core Extraction Distributed Ad Hoc Routing, CEDAR, [5]) combine the merits of both proactive and reactive protocols and overcome their shortcomings. While referring to the specialized literature for an exhaustive coverage of the topics, in the sequel a short description of the bee labor in a hive, and of *BeeAdHoc*, one of the most efficient swarm inspired routing algorithms for MANETs is given; *BeeAdHoc* will have, indeed, a fundamental role in the auto-configuration algorithm that is going to be introduced.

Bee colonies (*Apis Mellifera*) and the majority of ant colonies *(Argentine ant, Linepithema humile)* [6] show similar structural characteristics, such as the presence of a population of minimalist social individuals, and must face analogous problems for what is concerned with distributed foraging, nest building and maintenance. A honey bee colony consists of

morphologically uniform individuals with different temporary specializations. The benefit of such an organization is an increased flexibility to adapt to the changing environments. Thousands of worker bees perform all the maintenance and management jobs in the hive. There are two types of worker bees, namely *scouts* and *foragers*. The scouts start from the hive in search of a food source randomly keeping on this exploration process until they are tired. When they return back to the hive, they convey to the foragers information about the odor of the food, its direction, and the distance with respect to the hive by performing dances. A *round dance* indicate that the food source is nearby whereas a *waggle dance* indicate that the food source is far away. Waggling is a form of dance made in eight-shaped circular direction and has two components: the first component is a straight run and its direction conveys information about the direction of the food; the second component is the speed at which the dance is repeated and indicates how far away the food is. Bees repeat the waggle dance again and again giving information about the food source quality. The better is the quality of food, the greater is the number of foragers recruited for harvesting. The Bee Colony Optimization (BCO) meta-heuristic has been derived from this behavior and satisfactorily tested on many combinatorial problems [7], [8].

BeeAdHoc is a reactive source routing algorithm based on the use of four different bee-inspired types of agents: *packers, scouts, foragers, and bee swarms*. [9], [10], [11]. *Packers* mimic the task of a food-storekeeper bee, reside inside a network node, receive and store data packets from the upper transport layer. Their main task is to find a forager for the data packet at hand. Once the forager is found and the packet is handed over, the packer will be killed. *Scouts* discover new routes from their launching node to their destination node. A scout is broadcasted to all neighbors in range using an expanding time to live (TTL). At the start of the route search, a scout is generated; if after a certain amount of time the scout is not back with a route, a new scout is generated with a higher TTL in order to incrementally enlarge the search radius and increase the probability of reaching the searched destination. When a scout reaches the destination, it starts a backward journey on the same route that it has followed while moving forward toward the destination. Once the scout is back to its source node, it recruits foragers for its route by *dancing*. A dance is abstracted into the number of clones that could be made of the same scout. *Foragers* are bound to the bee hive of a node. They receive data packets from packers and deliver them to their destination in a source-routed modality. To attract data packets foragers use the same metaphor of a waggle dance as scouts do. Foragers are of two types: delay and lifetime. From the nodes they visit, delay foragers gather end-to-end delay information, while lifetime foragers gather information about the remaining battery power. Delay foragers try to route packets along a minimum delay path, while lifetime foragers try to route packets in such a way that the lifetime of the network is maximized. A forager is transmitted from node to node using an unicast, point-to-point modality. Once a forager reaches the searched destination and delivers the data packets, it waits there until it can be piggybacked on a packet directed to its original source node. In particular, since TCP (Transport Control Protocol) acknowledges received packets, *BeeAdHoc* piggybacks the returning foragers in the TCP acknowledgments. This reduces the overhead generated by control packets, saving at the same time energy. *Bee swarms* are the agents that are used to explicitly transport foragers back to their source node when the applications are using an unreliable transport protocol like UDP (User Datagram Protocol). The algorithm reacts to link failures by using special hello packets and informing other nodes through Route Error Messages (REM). In *BeeAdHoc*, each MANET node contains at the network layer a software module called *hive*, which consists of three parts: the *packing floor*, the *entrance floor*, and the *dance floor*, (see Figure 1), The entrance floor is an interface to the lower MAC layer; the packing floor is an interface to the upper transport layer; the dance floor contains the foragers and the routing information.

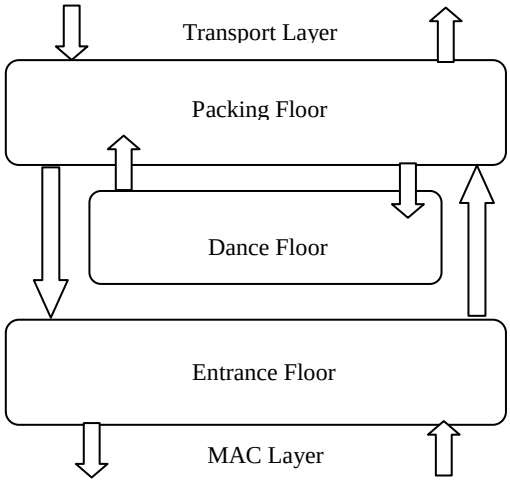

Fig 1: The network layer architecture of BeeAdHoc

BeeAdHoc has been implemented and evaluated both in simulation and in real networks. Results demonstrate a very substantial improvement with respect to congestion handling, for example due to hello messages overhead and flooding, and prove the algorithm far superior to common routing protocols, both single and multipath.

3. MANET ALLOCATION

Before a path between the nodes can be found, the nodes must be identified according to an uniform address scheme, and an unique address assignment policy in sight of an IP correct operation [12], [13]. The major requirement of ad hoc addressing schemes, indeed, is ensuring the uniqueness of node addresses so that no ambiguity appears when they try to communicate. This is not as trivial task because of the dynamic topology of an ad hoc network. A MANET can be split into several parts, and several MANET can merge into one. A great number of nodes coexisting in a single network may participate concurrently in the configuration process. Moreover, the wireless nature, such as limited bandwidth, power, and high error rate makes the problem even more challenging. Besides handling a dynamic topology, the protocols must take into account scalability, robustness, and effectiveness. Finally, in IPv6, a protocol is expected to tackle not only the local addressing, but also the global addressing since, even though a MANET is basically supposed to work by itself, the Internet connectivity might be useful in many contexts. The strong centralization of DHCP (Dynamic Host Configuration Protocol) and the local broadcast of IPv4 Link-Local Addresses are not suited for MANETs. Several approaches have been proposed to solve this problem, generally classified into categories reflecting the allocation features of protocols. *Stateful*, *stateless*, and *hybrid* approaches are the most popular classes of MANET address assignment protocols. For *stateful* approaches, the state of each address is held in such a way that the network has a vision of assigned and non assigned IPs, and the address

duplication is not a possible occurrence; Agent Based Addressing [14], MANETconf [15] , Prophet [16] , and Buddy [17] are the most popular among the stateful protocols. For *stateless* approaches, each node randomly chooses its own address and performs a duplicate address detection test to ensure that the chosen address is not already used; Strong Duplicate Address Detection (SDAD) [12], Weak Duplicate Address Detection (WDAD) [18], Passive Duplicate Address Detection (PDAD) [19] are the most popular among the stateless protocols. *Hybrid* approaches combine mechanisms from both stateful and stateless approaches, in order to improve reliability and scalability; Hybrid Centralized Query-based Auto-configuration (HCQA) is one of them [20]. The swarm approach was also tried for MANET auto-configuration, [21]. It is worthy noticing that the literature is very lacking in swarm-based auto-configuration algorithms, whereas it has plentiful of algorithms based on traditional approaches. In the sequel the Buddy protocol in subsection 3.1 and the ant based protocol in subsection 3.2 will be described, soon after a brief resume of the ant characterizing behavior.

3.1 The Buddy protocol

Buddy is a stateful protocol where every node stores a disjoint set of IP addresses which it can assign to a new node without consulting any other node in the network. At the beginning, only one node in the network has the entire pool of IP addresses; this node detects no neighbors, thus it auto-assigns itself with the first IP of the pool, entitles the network with an ID (Identifier), and becomes the *network initiator*. A new node, that wants to join the network, periodically sends broadcast messages reclaiming an IP address. The initiator assigns an address to it, divides the pool of IP addresses into two sets, gives one half to the requesting node, and keeps the other half with itself; the protocol agreement makes the requesting node to auto-assign itself with the first address in the received set. This process continues and eventually all the nodes in the network have a set of addresses to assign to other nodes. As a consequence, a requesting node can also receive one or more responses; in such a case, it will choose the first node that replies. If a node receives a request and has no available addresses, it should request its neighbors. Three different scenarios are possible: it searches its IP address table for possible one hop neighbor candidates and increment by one the radius of search if it finds no address availability; it sends a broadcast message to its one hop neighbors and a 2 hop broadcast if it receives no reply; it searches its IP address table for the node with the biggest block and contacts it directly. The synchronization of the address tables makes each node to periodically broadcast its address table. The detection of address leaks is accomplished by buddy nodes: if one node detects that another is missing, it merges its IP pool with its own IP pool. When networks merge, conflicting nodes have to give up their address space and acquire a new set of addresses. The protocol guarantees address uniqueness, does not generate unnecessary address changes, and is distributed, but produces a scarce balanced address assignment, and requires a consistent flooding that strongly increases the network overhead [17].

3.2 The ant-based protocol

Many ant species *(Argentine ant, Linepithema humile)* are able to discover the shortest path to a food source and to share that information with other ants through *stigmergy* [22]. In ant colonies, indeed, an odor substance, the pheromone, is used as an indirect communication medium. When a source of food is found, the ants lay some pheromone to mark the path. The quantity of the laid pheromone depends upon the distance, quantity and quality of the food source. While an isolated ant that moves at random detects a laid pheromone, it is very likely that it will decide to follow its path. This ant will itself lays a certain amount of pheromone, and hence enforces the pheromone trail of that specific path. Accordingly, the path that has been used by more ants will be more attractive to follow. The local intensity of the pheromone field, which is the overall result of the repeated and concurrent *path sampling* experiences of the ants, encodes a spatially distributed *measure of goodness* associated with each possible move. This form of distributed control based on indirect communication among agents which locally modify the environment and react to these modifications is called *stigmergy*. These basic ingredients have been reverse-engineered in the framework of Ant Colony Optimization (ACO) , which exploits the ant behavior to define a nature-inspired metaheuristic for combinatorial optimization. ACO has been applied with success to a variety of combinatorial problems, such as traveling salesman, routing, scheduling, [22], [23 [24], [25], [26], [27], showing to be an effective tool in finding good solutions.

The ant-based protocol presented in [21] is stateful and relies on the Ant Colony meta-heuristic. Every node creates and propagates through the network at least one *originator ant*. The node may destroy, reproduce or duplicate the originator ant that, on its own, has the exclusive right to initiate any change involving its parent IP address when a conflict is detected. The ants, usually identified by means of the Medium Access Control (MAC) of their originator nodes, spread their own node information, collect other node information, and induce feedback within the network using the environment as interchange means. The environment is usually realized as a small segment of memory that nodes and ants hold and employ during their mutual updating interactions. Basically, the memory segments contain the MAC address, the IP address and a timestamp for each of the currently known nodes. Timestamp reflects the time elapsed since the node initialization; in order to deal with a totally distributed control, nodes do not need synchronization. When the process begins, each memory segment would have only one entry pointing to itself; as the algorithm progresses information about other nodes will be brought in, and the environment will be dynamically built. At the boot time, a set of IP addresses is available for auto-configuration; each node randomly picks up a unique address, and creates its originator ant that starts its journey through the network. At each step the next hop is chosen with respect to the optimization criterion suggesting to reach the least recently updated node. The exchange of information between a node and an ant is based on the timestamps the ants carry on a per entry basis. On a network with n nodes, the ants carries n IP addresses, one for each node, usually the most recent ones according to its knowledge. When information exchange between the node and the arriving ants takes places, either of them updates itself based on the timestamps. Whenever an ant during the process of its journey detects a conflict for the node it has originated from, it takes responsibility to inform it and have it changed. A conflict is detected when two or more nodes have chosen the same IP address. Conflict resolution mechanism is based on mechanisms followed in Zero-Configuration networks. The node that has the least MAC address takes the responsibility to have its node changes its IP address to a different one. This is not a one step process but the result of various interactions among the swarms. The conflict resolution mechanism will continue until a state wherein all

the nodes have unique IP address is reached. Due to the completely distributed control and feedback flow, the swarm based system guarantees that, even in case of node or link failure, only a partial component of information is lost so that the system can quickly recover from it. An important feature of the swarm based model is concerned with partitions which do not need to be considered as special cases. On the contrary, when partitions merge, there is a sudden increase in the number of IP address conflicts and the system has to make a large effort to respond to the new environmental change.

4. BEEADHOCAUTOCONF

Each node in the MANET has a hive, which consists of three parts: packing floor, entrance floor, and dance floor. The architecture is defined in [7],(see Figure 1), and it is the base for the operation of the *BeeAdHoc* routing algorithm whose services are supposed to be asked from *BeeAdHocAutoconf* as far as possible. *BeeAdHocAutoconf* follows a decentralized stateless approach and is made up by two components: the address assignment and the Duplicate Address Detection (DAD) procedure. Based on a predefined conflict probability, an estimation of the number of nodes and a widespread allocation table, the first component randomly selects an address from this space. The selected address is assigned immediately implying a fixed node configuration time. However, the address might be duplicate. Such a circumstance will be detected and resolved by the DAD component using a bee-swarm approach. We will refer to it as Bee Swarm Duplicate Address Detection (BSDAD) (see Figure 2).

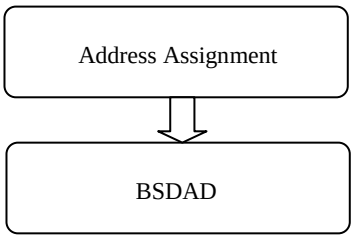

Fig 2: Components of BeeAdHocAutoconf

Protocol operation. When a node wishes to join a network (*source node* in the sequel), it randomly picks up an address, starts setting up a local allocation table, and broadcasts a scout to all neighbors in its range using an expanding TTL. The task of such a TTL is to control the number of times a scout may be re-broadcasted. Each scout is uniquely identified with a key based on its source node identifier (ID) and a sequence number. The task of the scout is twofold: it checks whether or not other nodes on its route are using the same address of its source node, and brings back useful information either if it finds a duplicate address occurrence or not. The source node broadcasts the scout after assigning a small TTL to it and setting up a timer for itself. When the TTL expires, the scout might increment it in order to enlarge the search radius and increase the probability of reaching a node that might use a duplicate address. A maximum TTL is also established with respect to a reasonable size for an ad hoc network. Scouts with exceeded TTL might be killed or not depending on the information they have gathered until then. This mechanism helps ensuring the address uniqueness when the TTL expires and useful address information has not been collected meaning that the source node is a network initiator. Scouts that on their route have been seen already are deleted in order to limit the overhead. More precisely, the algorithm proceeds as follows.

1. The source node assigns to itself a random IP address, broadcasts the scout with a small TTL, sets up a local allocation table as well as a timer till the maximum fixed TTL value.
2. In each node on its route the scout updates a list *L* containing the just visited node and the nodes which have already routes to destinations. In order to get these routes, the scout asks the local hive to look for them by demanding foragers from the dance floor. Eventually, it compares the entries of *L* with the source node address. If there is a match,
   - 2.1. the scout goes immediately back to the source node by means of the reverse route. The source node must pick up a new address, update the allocation table and restart from step 1 unless the new tentative address is founded in the updated allocation table.
   - 2.2. otherwise, its TTL is checked. If the TTL has not expired yet,
     - 2.2.1. the scout is rebroadcasted and continues as in step 2.
     - 2.2.2. otherwise, it is checked if TTL= max. If it has not reached such a maximum yet,
       - 2.2.2.1. its TTL is incremented and the scout continues as in step 2.
       - 2.2.2.2. otherwise, it checks *L*. If $L = \Phi$
         - 2.2.2.2.1. it kills itself. The source node may assume to be the network initiator and, consequently, its allocation table will have just one entry.
         - 2.2.2.2.2. otherwise, it brings *L* to the source node. The source node is not the network initiator, but there is not address conflict. Its allocation table will have numerous entries.

When a network has been configured by means of *BeeAdHocAutoconf*, each node will have a partial knowledge about the network address distribution rising from the allocation table it stores. Such a knowledge might be recursively used by scouts when gathering as much information is possible to bring back to their source node. At the moment, the implementation of *BeeAdHocAutoconf* does not provide for such a service. When a node leaves the network, address reclamation is not trivially needed. When a network becomes partitioned, the existing addresses are different; thus the newly allocated addresses will still be different inside the new partitions. The problem occurs when different networks merge. Since there is not guarantee that the addresses in the merged networks are different, address duplicates might exist. The solution we have implemented is the idea behind WDAD [18], that is duplicate addresses may be tolerated as long as packets reach the destination node intended by the sender, even if the destination node address is being used by another node also. Thus each node selects an identification key to make routing capable of differentiating between potential duplicate IPs. Each node generates a key at initialization phase, and distributes it with its IP address in all routing messages. This key will be used to detect duplicate IP addresses. Each node maintains keys along with IP addresses in its routing table. When a node receives a routing message with an IP address that exists in its table, it checks whether the keys are different or not. If they are different, a duplicate address is detected and the entry is marked as invalid; beacon messages will inform other nodes about this duplication.

5. SIMULATION

BeeAdHocAutoConf has been evaluated comparing its performances with those of *Buddy* and the ant approach. We have used a MASON (Multi-Agent Simulator of Neighborhood…or Network…or something) [28] based simulator. MASON does not allow to vary among different routing protocols, but it "*is a fast discrete-event multi-agent simulation library core in Java, designed to be the foundation for large custom-purpose Java simulation, and also to provide more than enough functionality for many lightweight simulation needs*". That has made it possible to design a suitable environment for the scenarios which were needed. Experiments were carried out varying parameters such as simulation area, network size, mobility pattern, coverage range, and simulation number among values reported in the Table 1. As it is well known, the random walk mobility is an individual mobility model indicating that a random walk with very small steps gets an approximation to Brownian motion. Despite the apparent limitations of a Brownian-type model, it has been widely used in the MANET scenarios because of its effectiveness in aggregating node movements in a very large ad hoc networks.

Table 1. Parameters and related values used in the simulation

Parameters	Values
Simulation Area	35 m x 35 m - 200 m x 200
Network size	50 - 1600
Mobility Pattern	Random Walk 2d Mobility
Coverage Range	30 m
Simulation Number	288

Node and link failures were considered during burst intervals. Every node was given a set of neighbor nodes to which it can directly communicate in a duplex manner. Comparisons about the connection numbers and relative operation times have been made with a binary exponential increment of the node number step by step as shown in Table 2 and 3, where each result is the average of 8 simulations grouped by number of nodes. As Tables 2 and 3 show, *BeeAdHocAutoConf* performances appear promising with respect to the ant approach and *Buddy* , both for the number of connected nodes and the requested time to converge as the network size increases. The ant-based algorithm holds good with respect to the execution time suffering yet for the number of configured nodes. *Buddy* behaves well with respect to configured nodes suffering yet for the execution time as compared with both the swarm-like algorithms.

Table 2. Network sizes and connection numbers

Size	BeeAdHocAutoConf	Ant-based	Buddy
50	97,80	100,00	100,00
100	101,00	99,40	96,40
200	97,80	92,60	89,50
400	95,90	90,00	94,70
800	93,90	81,70	95,50
1600	93,80	74,80	92,60

Table 3. Network sizes and connection times

Size	BeeAdHocAutoConf	Ant-based	Buddy
50	59,40	57,00	119,90
100	109,10	109,30	209,10
200	224,00	214,60	399,00
400	461,00	433,60	798,10
800	1141,50	1217,40	1994,40
1600	2442,50	3552,20	5857,10

6. CONCLUSIONS

BeeAdHocAutoConf has been presented; its simulation showed that ideas inspired from natural systems provide a sufficient motivation for designing and developing algorithms for not only scheduling and routing problems, but for auto configuration also. According to [7] an reengineering approach has been followed that allowed to map concepts from a bee colony to an IP address auto-configuration algorithm. The algorithm has been evaluated in a simulation environment; however, the simulation model was developed in such a way that the constraints of a real network would be taken into account. Extensive testing and evaluations under various environmental parameters that represent real network conditions have been done. The results from all experiments reveal that the performance of *BeeAdHocAutoConf* is of the order of the best auto-configuration algorithms known in literature, even though it is achieved at a much less energy expenditure. Future works would consider the extension of the protocol to deal with the improvement of the network merging management, the global connectivity with Internet, security issues, the TCP congestion control, the exploration of the honey bee colony behavior for its reengineering in other problem frameworks as well as the exploration of different swarm intelligence forms to be used in problem solving.

A last consideration about the amount of things that nature has still to teach to everybody is strictly due. It has very recently been discovered by two Stanford researchers that *Pogonomyrmex barbatus* colonies, a species of harvester ants, determine how many foragers to send out of the nest in much the same way that TCP discovers how much bandwidth is available for the transfer of data in Internet in order to avoid or recover from network congestion. The researchers are calling them the *anternet*. According to Prabhakar it is worthwhile to conclude by saying "Ants have discovered an algorithm that we know well, and they've been doing it for millions of years", [29].

7. ACKNOWLEDGMENTS

Our thanks to doctors L. Caputo, C. Davino, V. Ferri, and S. Piscitiello who have contributed towards the development of the MASON-based simulator and relative results.

8. REFERENCES

[1] Siva Ram Murthy, C and Manoj. 2004 Ad Hoc Wireless Networks: Architecture and Protocols. Prentice Hall.

[2] Royer, E. and Toh, C.K. 1999 A Review of Current Routing Protocols for ad hoc Mobile Wireless Networks. IEEE Personal Communications. **6** , 46-55.

[3] Perkins C. and. Bhagwat P. 1994 Highly dynamic destination-sequenced distance vector routing (DSDV) for mobile computers. Proceedings of SIGCOMM, 234–244.

[4] Johnson D. B. and Maltz D. A. 1996 Dynamic source routing in ad hoc wireless networks. Mobile Computing, 153–181.

[5] Sinha P., Sivakumar R. and Bharghavan V. 1999 CEDAR: a core extraction distributed ad hoc routing protocol. IEEE INFOCOM, 202-209.

[6] Bonabeau E., Dorigo, M. and Theraulaz, G. 1999 Swarm Intelligence. From Natural to Artificial Systems, Oxford University Press, 0-19-513159-2.

[7] Wedde H.F. and Farooq M. 2005 Beehive: New ideas for developing routing algorithms inspired honey bee behavior. Handbook of Bioinspired Algorithms and Applications, 21, 321–339.

[8] Wedde H.F. and Farooq M. 2005 The wisdome of the hive applied to mobile ad-hoc networks. Proceedings IEEE Swarm Intelligence Symposium, 341–348.

[9] Wedde H.F. and Farooq M. 2005 A performance evaluation framework for nature insired routing algorithms. Applications of Evolutionary Computing, LNCS (3449), 136–146.

[10] Wedde H.F. and Farooq M. 2004 BeeAdHoc–An Energy-Aware Scheduling and Routing Framework,.Technical report-pg439, LSIII, School of Computer Science, University of Dortmund.

[11] Farooq M. 2006 Intelligent Network Traffic Engineering through Bee-inspired Natural Protocol Engineering. Natural Computing Series, Springer.

[12] Perkins C., Malinen J. T., Wakikawa R., Belding-Royer E. M. and Sun Y. 2001 IP address auto-configuration for ad hoc networks, IETF Draft.

[13] Jeong J., Park J., Kim H., Jeong H.and Kim D., 2005 Ad Hoc IP Address Autoconfiguration. IETF draft.

[14] Günes M.and Reibel J. 2002 An IP Address Configuration Algorithm for Zeroconf Mobile Multihop Ad Hoc Networks. Proceedings Broadband Wireless Ad Hoc Networks and Services.

[15] Nesargi S. and Prakash R. 2002 MANETconf: Configuration of Hosts in a Mobile Ad Hoc Network. Proceedings IEEE INFOCOM

[16] Zhou H., Ni L. M. and M. W. Mutka Prophet Address Allocation for Large Scale Manets. Proceedings IEEE INFOCOM.

[17] Mohsin M., Prakash R. 2002 IP Address Assignment in a Mobile Ad Hoc Network. Proceedings IEEE MILCOM.

[18] Vaidya N. H. 2002 Weak Duplicate Address Detection in Mobile Ad Hoc Networks. Proceedings ACM MobiHoc, 206–16.

[19] Weniger K. 2003 Passive Duplicate Address Detection in Mobile Ad Hoc Networks. Proceedings IEEE WCNC.

[20] Sun Y. and Belding-Royer E. M. 2003 Dynamic Address Configuration in Mobile Ad Hoc Networks. UCSB tech. rep. 2003-11.

[21] Ring S., Kumar V., Cole M. E., 2004 Ant Colony Optimization Based Model for Network Zero.Configuration. Proceedings SPCOM, 423-427.

[22] Dorigo M. and Stützle T. 2004 Ant Colony Optimization. MIT Press, 0-262-04219-3.

[23] Dorigo M., Maniezzo V. and Colorni A. 1996 Ant system: optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics-Part B, 26(1), 29-41.

[24] Di Caro G.A., 2004 Ant Colony Optimization and its application to adaptive routing in telecommunication networks. PhD thesis, Facultè des Sciences Appliquèes, Universitè Libre de Bruxelles.

[25] Di Caro G.A. and Dorigo M. 1997 *A mobile agents approach to adaptive routing*, Technical Report 97–12, IRIDIA, Universitè Libre de Bruxelles.

[26] Di Caro G.A., Ducatelle F .and Gambardella L.M. 2004 AntHocNet: an ant-based hybrid routing algorithm for mobile ad hoc networks. Proceedings PPSNVIII, LNCS (3242), 461–470.

[27] Di Caro G.A., Ducatelle F., and Gambardella L.M., 2005 AntHocNet: an adaptive nature-inspired algorithm for routing in mobile ad hoc networks. European Transaction on Telecommunications, 16(5), 443–455.

[28] Luke S., Cioffi-Revilla C., Panait L., Sullivan K., and Balan G., 2005 MASON: A Multiagent Simulation Environment. Simulation (81), 517-527.

[29] Prabhakar B., Dektar K. N., and Gordon D. M. 2012 The Regulation of Ant Colony Foraging Activity without Spatial Information. PLOS Computational Biology.