AutoBeeConf

A swarm intelligence algorithm for MANET administration

Luca Caputo, Cristiano Davino, Filomena de Santis,Vincenzo Ferri
Dipartimento di Informatica
Facoltà di Scienze, Università degli Studi di Salerno
Fisciano (SA), Italy

***Abstract*— In a mobile ad-hoc network (MANET) nodes are self-organized without any infrastructure support: they move arbitrarily causing the network to experience quick and random topology changes, have to act as routers as well as forwarding nodes, some of them do not communicate directly with each other. Routing and IP address auto-configuration are among the most challenging tasks in the MANET domain. Swarm Intelligence is a property of natural and artificial systems involving minimally skilled individuals that exhibit a collective intelligent behavior derived from the interaction with each other by means of the environment. Colonies of ants and bees are the most prominent examples of swarm intelligence systems. Flexibility, robustness, and self-organization make swarm intelligence a successful design paradigm for difficult combinatorial optimization problems, such as routing and IP address allocation in MANET. This paper proposes *AutoBeeConf*, a new IP address auto-configuration algorithm based on a bee swarm labor that may be applied to large scale MANET with low complexity, low communication overhead, even address distribution, and low latency. Both the protocol description and the simulation experiments are presented to demonstrate the advantages of AutoBeeConf over two known algorithms, namely *Buddy* and *Antbased* protocols. Eventually, future research directions are established, especially toward the principle that swarm intelligence paradigms may be usefully employed in the redefinition or modifications of each layer in the TCP/IP suite in such a way that it can efficiently work even in the infrastructure-less and dynamic nature of MANET environment.**

Keywords—MANET; Routing protocols; IP address auto-configuration; Swarm intelligence.

I. INTRODUCTION

Advances in wireless communication technology have strongly encouraged the use of low-cost and powerful wireless transceivers in mobile applications. As compared with wired networks, mobile networks exhibit unique features: recurrent network topology changes, link capacity fluctuations, critical bounds to their performances. Mobile networks can be classified into infrastructure networks and mobile ad-hoc networks, [1]. In an infrastructure mobile network, mobile nodes communicate through wired access points that work in the node transmission range and create the backbone of the network. In a mobile ad-hoc network (MANET) nodes, acting potentially both as routers and hosts, are generally equipped with either omnidirectional or directional antennas for sending and receiving information. They have a packet-forwarding capability in order to communicate via shared and limited radio channels. Communication may be performed by one-to-one transmissions (single-hop) or using other nodes as relay stations (multi-hop). In both cases each sender node must adjust its emission power in order to reach the respective receiver node. In cases where energy is supplied by batteries, the network lifetime is limited by the batteries of the wireless devices. Therefore, energy saving is critical in all network operations. Ad-hoc networks are suitable for situations where only temporary communication is needed, and establishing a communication infrastructure is either not possible or not desirable. As an example for an ad-hoc network, we can imagine a meeting in which the members want to interchange data. The participants do not want to make high efforts for the network configuration since; perhaps, the users are not technically skilled. Notwithstanding, users wish a convenient way for their cooperation.

A challenging task in the MANET domain is 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 protocol nodes continuously evaluate routes towards all reachable nodes and maintain consistent, up-to-date routing information even though network topology changes occur (e.g. Destination Sequenced Distance Vector, DSDV, [3]) . In a *reactive* routing protocol, routing paths are searched only when needed by means of a route discovery operation established between the source and destination node (e.g. Dynamic Source Routing, DSR, [4]). *Hybrid* routing protocols combine the merits of both proactive and reactive protocols and overcome their shortcomings (e.g. Core Extraction Distributed Ad-Hoc Routing, CEDAR, [5]). However, 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 (Internet Protocol) correct operation [6]. The strong centralization of DHCP (Dynamic Host Configuration Protocol) and the local broadcast of IPv4 Link-Local Addresses are not suited for MANETs, where topology changes, network partitioning and merging cannot assure that every mobile node will be connected at a given time neither predict the topology or size of the network. 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 *statefull* approaches, the state of each address is held in such a way the network have a vision of assigned and non assigned IPs, so address duplication could be avoided. 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. *Hybrid* approaches combine mechanisms from both stateful and stateless approaches, in order to improve reliability and scalability. The price is a more complex protocol.

Swarm Intelligence (SI) is an novel distributed paradigm for the solution of hard problems taking insight from biological examples such as colonies of ants, bees, and termites, schools of fish, flocks of birds, [7]. The most interesting property of SI is the involvement of multiple individuals that interact with each other and the environment, exhibit a collective intelligent behavior, and are able to solve complex problems. Many applications, mainly in the contexts of computer networks, distributed computing and robotics are nowadays being designed using SI, [8], [9]. The basic idea behind this paradigm is that many tasks can be more efficiently completed by using multiple simple autonomous agents instead of a single sophisticated one. Regardless of the improvement in performance, such systems are usually much more adaptive, scalable and robust than those based on a single, highly capable, agent. An artificial swarm can generally be defined as a decentralized group of autonomous agents having limited capabilities. Due to the adaptive and dynamic nature of MANETs, the swarm intelligence approach is considered a successful design paradigm to solve the routing and the IP address auto-configuration problems.

The rest of this paper is organized as follows. Section 2 briefly reviews references on the swarm paradigm, specifically based on ant and bee behaviors, with a glance at their use for the solution of the MANET routing problem. Section 3 first defines the IP address auto-configuration problem for ad-hoc networks, then describes two well kwon protocols, such as the *Buddy* protocol and the *AntConf* protocol, developed with a stateful approach based on the *binary split* idea of [10], and with a swarm intelligence based model targeted at network administration [11], respectively. Section 4 contains the description of the *AutoBeeConf* protocol, our proposal for the IP address auto-configuration for MANET that integrates the advantages deriving from the classical approaches with the benefits arising from the most typical activities of a bee swarm. Section 5 presents the simulations carried on to test and compare the performances of the three before mentioned protocols. Eventually, section 6, after reviewing the main features of AutoBeeConf, sketches potential future extensions to the work.

II. THE SWARM PARADIGM

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* [12]. 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 modifies the environment and reacts 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 meta-heuristic for combinatorial optimization. ACO has been applied with success to a variety of combinatorial problems, such as traveling salesman, routing, scheduling, and has been shown to be an effective tool in finding good solutions.

Bee colonies (*Apis mellifera),* show structural characteristics similar to those of ant colonies, such as the presence of a population of minimalist social individuals, and must face analogous problems such as 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 distance with respect to the hive by performing dances. A *round dance* indicates that the food source is nearby whereas waggle dances indicate that the food source is far away. Waggling is a form of dance in eight-shaped circular direction. It is repeated again and again; its intensity and direction gives information about the food source quality and location, respectively. The better is the quality of food; the greater is the number of foragers recruited for harvesting. In analogy with ACO, the Bee Colony Optimization (BCO) meta-heuristic has been defined and satisfactorily tested on many combinatorial problems [13].

While referring to the specialized literature for an exhaustive coverage of swarm-inspired algorithms, in the sequel we will limit our attention to a short description of a few routing algorithms, namely modeled on both ant and bee behaviors, which can help in appreciating equivalent solutions in the IP address auto-configuration domain.

A. AntNet and AdHocNet

The first ACO routing algorithm, *AntNet* [14], [15] was designed for wired packet-switched networks. It is a proactive algorithm where each node periodically sends a *forward ant* to a random destination. The forward ant records its path as well as the time needed to arrive at each intermediate node. The timing information recorded by the forward ant, which is forwarded with the same priority as data traffic, is returned from the destination to the source by means of a high priority *backward ant*. Each intermediate node updates its routing

tables with the information from the backward ant. Routing tables contain per destination next hop biases so that faster routes are used with greater likelihood. The algorithm exhibits a number of interesting properties which are also desirable for MANET: it can work in a fully distributed way, is highly adaptive to network and traffic changes, uses mobile agents for active path sampling, is robust to agent failures, provides multipath routing, and automatically takes care of data load spreading. However, the fact that it crucially relies on repeated path sampling can cause significant overhead.

AntHocNet is a hybrid multipath algorithm *for* routing in mobile ad-hoc networks consisting of reactive and proactive components, [16], [17]. It does not maintain routes to all possible destinations at all times (like *AntNet*), but only sets up paths when they are needed at the start of a data session. This is done in a *reactive route setup* phase, where the *reactive forward ants* are launched by the source in order to find multiple paths to the destination, and the *backward ants* return to the source to set up the paths. According to the common practice in ACO algorithms, the paths are set up in the form of pheromone tables indicating their respective quality. After the route setup, data packets are routed stochastically over the different paths following these pheromone tables. While the data session is going on, new ants, the *proactive forward ants, monitor*, maintain and improve paths. This allows to adapt to changes in the network, and to construct a mesh of alternative paths between source and destination. The proactive behavior is supported by a lightweight information bootstrapping process. Link failures, detected by unicast transmissions or expected hello messages crashes, and are coped with either a local route repair or by warning preceding nodes on the paths.

Antnet and *AntHocNet* have been evaluated on the basis of a relatively large number of simulation experiments using a custom network simulator. The algorithms have been tested on a variety of different scenarios based on different topologies with a variable number of nodes, and considering UDP traffic patterns with different geographical and generation characteristics. The reported experiments show that they robustly outperform several different dynamic state-of-the-art algorithms in terms of throughput and delay.

B. BeeHive and BeeAdHoc

BeeHive is a proactive algorithm that models bee agents in packet switching networks for routing purposes, [18], [19]. Since in nature the majority of forager's exploits food sources nearby the hive whereas a minority visits food sites far away from it, the algorithm provides for two types of agents: short distance bees and long distance bees which collect and disseminate routing information in the neighborhood of their source and in the entire network, respectively. They differ in their life time that is the number of hops they can travel across. Nodes periodically send a bee agent, by broadcasting replicas of it to each neighbor. When a replica of a particular bee agent arrives at a site, it updates routing information before being flooded again. This process continues until the life time of the agent expires, or if a same replica had been received already at a site. Short and long distance bees allow to partition the network in *foraging zones* and *foraging regions* so that each node maintains current routing information to reach all nodes in its zone and only the address of a *region representative node* to reach nodes located outside its zone. The next hop for a data packet is selected in a probabilistic manner according to a quality measures assigned to the current node. As a result, not all packets follow "best" paths. This will help in maximizing the system performance though a data packet may not follow a best path, a concept directly borrowed from a principle of bee behavior: a bee could only maximize her colony profit if she refrains from broadly monitoring the dance floor to identify the single most desirable food.

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*. [20], [21]. *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 a 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 bounded for 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*. 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.

Beehive and *BeeAdHoc* have 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 proved both the algorithm far superior to common routing protocols, both single and multipath.

III. IP Address Auto-Configuration

The most important constraint of ad-hoc addressing schemes is to guarantee the uniqueness of node addresses so that no uncertainty appears in communication. This is not a trivial task because of the dynamic topology of ad-hoc networks. A MANET, indeed, can be split into several parts, and several MANETs can merge into one, and an indefinite 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 make 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 not only to deal with the local addressing, but also the global addressing. Since 1998, several address auto-configuration protocols for IPv4 and IPv6 have been proposed, each of them attempting to achieve a level of optimization for a particular aspect [6], [22].

In the sequel we will describe two well known IP auto-configuration protocols, namely *Buddy* and *AntConf*, that we implemented with the aim to compare their performances with those of the proposed algorithm *AutoBeeConf*.

A. 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 it is complex to implement, produces a scarce balanced address assignment, and requires a consistent flooding that strongly increases the network overhead, [10].

B. The AntConf Protocol

AntConf is a stateful protocol based on the Ant Colony meta-heuristic, where 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 carry 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 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 change 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, [11], [23].

IV. AUTOBEECONF

Auto-BeeConf is the new auto-configuration algorithm for efficient ad-hoc network administration presented in this paper. The algorithm is inspired by the foraging principles of honey bees and it is supposed to share the services of the IP layer; more precisely, *Auto-BeeConf* is supposed to relay on the services of the *BeeAdHoc* routing protocol placed in the TCP/IP suite of any network node. The main features of *Auto-BeeConf* are two: first, the acquirement of the controlled multicast, and second, the intelligent division of the labor force which is done proportionally to the available food resources. The controlled multicast allows to limit the information to propagate only through a node subset in such a way that the network is poorly flooded inducing a noticeable overhead and energy saving. The labor division allows nodes to manage a number of addresses proportionally to their battery charges in such a way that the address losses are reduced when nodes leave the network because of a battery discharge. *Auto-BeeConf* is a hybrid algorithm that works through two phases. In the first one, a node that wants to join a MANET tries to get an IP address by means of state-full policy that allows it to look for an address among its neighbor nodes; in case of an iterated number of failures with respect to its request, it assumes that none of its neighbors has free addresses and starts trying with a stateless policy. In such a way the incoming node has the chance to look for conflicts as well as for a valid available address. The two phases strongly balance themselves inducing a promising improvement in performances as compared to existing state-of-the-art auto-configuration algorithms due to the reduced use of control packets.

A. Protocol Operation

A node that wants to join a MANET senses its neighbors by means of *Hello Messages* and sends an *IP Assignment Request Bee Agent* to the best of them soon after the initializations of two variables, *my_back-off* and *my_patience*. A node is better than another when its battery charge is larger; the incoming node gets such information from the *Hello Messages*.

Phase 1: The neighbor nodes receiving the request look for a free IP address in their tables. If they do not have free addresses, they only discard the request; otherwise, they divide their tables proportionally to the requiring device life-time and type , send a part to it by using the just received bee agent, and start waiting for an acknowledgment (ACK). In the case where the ACK does not return within a certain amount of time, neighbor nodes restore their original IP address table. The requiring node might receive various address blocks depending on the number of neighbors which captured its bee; it will only retain the first one acknowledging the owner in such a way that the other blocks it received may be released. However, the requiring node might also not receive any response from its neighbors. Thus it is necessary to enlarge the search radius. The *IP Assignment Request Bee Agent* must be now flooded to all neighbors in range using a number of iteration (*my_patience*) and a back-off time (*my_back-off*). Phase 1 will be iterated until the maximum value of *my_patience* will be reached. If the requiring node has not still be configured, it must enter the Phase 2 of the algorithm.

Phase 2: The node generates an IP address that is coherent with the address class, the network mask and the *my_patience* value using a MAC address based function. In order to verify the uniqueness of such an address, the node auto-assigns it to itself, resets the my*_patience* value, and generates a *BeeARP* according to the specification of the Address Resolution Protocol (ARP) in the TCP/IP suite, and the setting of a TTL. The bee-agent is sent to its best neighbors, and, each time it reaches a node, it verifies whether or not it is the destination node. In the former case, it asks the destination for a free IP address that, if available, quickly is brought back to the requiring node. In case a free IP address is not available, the bee-agent starts its journey back toward the source trying to get a free IP address from each intermediate node. In the latter case, when the bee-agent has reached a node that is not the destination, it is tried to be forwarded to the destination by means of the *BeeAdHoc* algorithm until its TTL expires. Thus, from the requiring node point of view, a *BeeARP* might come back or not. If it does, the next step is to verify whether or not it has certified the absence of an address conflict since in one case the node can begin its network activities whereas in the other it is still in lack of an address. The requiring node might also consider itself configured when the *BeeARP* TTL expires before returning home. Phase 2 is allowed to be iterated a *my_patience* maximum value number of times.

In case of failure, a *max_try* value bounds possible iteration of both phase1 and phase 2; after that the access to the MANET is forbidden to the requiring node since it is reasonable to think that IP addresses are all over, or the node is in a hotspot or dead-zone.

Network partitions do not affect the protocol operation. Network merging might create conflicts. In this case, as soon as the merging is detected by some node via the ID network, a bee swarm might be quickly broadcasted through the network with the task to resolve conflicts according to phase 2.

V. SIMULATION FRAMEWORK

The performance of *AutoBeeConf* has been evaluated as compared to *AntConf*, and *BuddyConf* using a MASON (Multi-Agent Simulator Of Neighborhoods. ….or Networks...or something), [24], [25], based simulator. Even though MASON "*is a fast discrete-event multiagent simulation library core in Java, designed to be the foundation for large custom-purpose Java simulations, and also to provide more than enough functionality for many lightweight simulation needs",* it does not allow to vary among different routing protocols. Nevertheless, MASON is Java based so that this has made it possible to design a suitable environment for the necessary scenarios.

Simulations were carried out for the set of parameters reported in TABLE I. 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.

TABLE I. SIMULATION PARAMETERS

Parameters	*Values*
Simulation Area	35 m x 35 m to 200 m x 200 m
Mobile Node Number	50 to 1600
Mobility Pattern	Random Walk 2d Mobility Model
Node Range or Coverage	30 m
Simulation Number	288

Comparisons have been made both in discharging and not discharging modalities with a binary exponential increment of the node number step by step as shown in TABLES II and III, where each result is the average of 8 simulations grouped by number of nodes.

As TABLES II and III show, *AutoBeeConf* performances appear promising with respect to *AntConf* and *BuddyConf* , 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. *BuddyConf* behaves well with respect to configured nodes suffering yet for the execution time as compared with both the swarm-like algorithms. *AutoBeeConf* takes advantages from the cooperation of the two phases it uses: when the number of devices that want to join the network increases, and thus the probability of the address space depletion increases, the second phase of the algorithm allows to quickly recovering all the lost addresses. TABLES IV and V simply synthesize results of TABLES II and III.

VI. CONCLUSIONS AND FUTURE WORKS

A new auto-configuration algorithm for wireless ad-hoc networks, *AutoBeeConf*, has been presented. Its simulation showed that ideas inspired from natural systems provide a sufficient motivation for designing and developing algorithms for scheduling and routing problems as well as for auto-configuration. According to the literature a reverse engineering approach has been followed that has allowed mapping concepts from a bee colony to an auto-configuration problem. 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 evaluation under varying environmental parameters that represent a real network conditions have been done. The results from all experiments reveal that the performance of AutoBeeConf is of the order of the best auto-configuration algorithms, even though it is achieved at a much less energy expenditure.

Future works could consider extension of the protocol to deal with:

- network merging,
- global connectivity with Internet,
- security issue,
- TCP congestion,
- exploration of the honey bee colony behavior for its reengineering in other problem framework,
- Exploration of the different swarm intelligence forms.

A last consideration about the amount of things that nature has still to teach to everybody is 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", [26].

REFERENCES

[1] C. Siva Ram Murthy and B. S. Manoj, "Ad-hoc wireless networks: architectures and protocols", Prentice Hall, 2004

[2] E. Royer and C.K. Toh, "A review of current routing protocols for ad-hoc mobile wireless networks", IEEE Personal Communications, vol. **6,** 1999, pp. 46-55.

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

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

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

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

[7] E. Bonabeau, M. Dorigo and G. Theraulaz, "Swarm intelligence. From natural to artificial systems, 1999, Oxford University Press, 0-19-513159-2.

[8] D. Chrysostomou and A. Gasteratos, "Optimum multi-camera arrangement using a bee colony algorithm", Proceedings of IEEE Int. Conf. on Imaging Systems and Techniques, 2012, pp. 387 – 392.

[9] D. Chrysostomou, A. Gasteratos, L. Nalpantidis and G. Ch. Sirakoulis, "Multi-view 3D scene reconstruction using ant colony optimization techniques", Measurement Science and Technology, 2012, vol. 23, 11.

[10] M. Mohsin and R: Prakash, "IP address assignment in a mobile ad-hoc network, Proceedings IEEE MILCOM, 2002.

[11] S. Ring, V. Kumar and M. E. Cole, "Ant colony optimization based model for network zero-configuration", Proceedings SPCOM, 2004, 423-427.

[12] M. Dorigo and T. Stützle, "Ant colony optimization", MIT Press, 2004, 0-262-04219-3.

[13] M. Farooq, "Intelligent network traffic engineering through bee-inspired natural protocol engineering", Natural Computing Series, Springer, 2006.

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

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

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

[17] G.A Di Caro, F. Ducatelle and L.M. Gambardella, "Anthocnet: an adaptive nature-inspired algorithm for routing in mobile ad-hoc

networks", European Transactions on Telecommunications, 2005, 16 (5), pp. 443–455.

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

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

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

[21] H.F. Wedde and M. Farooq, "Beeadhoc: an energy-aware scheduling and routing framework, Technical report-pg439, LSIII, School of Computer Science, University of Dortmund., 2004.

[22] J. Jeong, J. Park, H. Kim, H. Jeong and D. Kim, "Ad-hoc IP address auto-configuration", IETF draft, 2005.

[23] V. Kumar and E. Cole, "An ant colony optimization model for wireless ad-hoc auto-configuration", Proceedings of IEEE Int. Conf. on Systems, Man and Cybernetics, 2005, vol. I, pp. 103-108.

[24] http://cs.gmu.edu/~eclab/projects/mason/

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

[26] B. Prabhakar, K. N. Dektar and M. Gordon, "The regulation of ant colony foraging activity without spatial information, PLOS Computational Biology, 2012 , vol. VIII, 8, e1002670

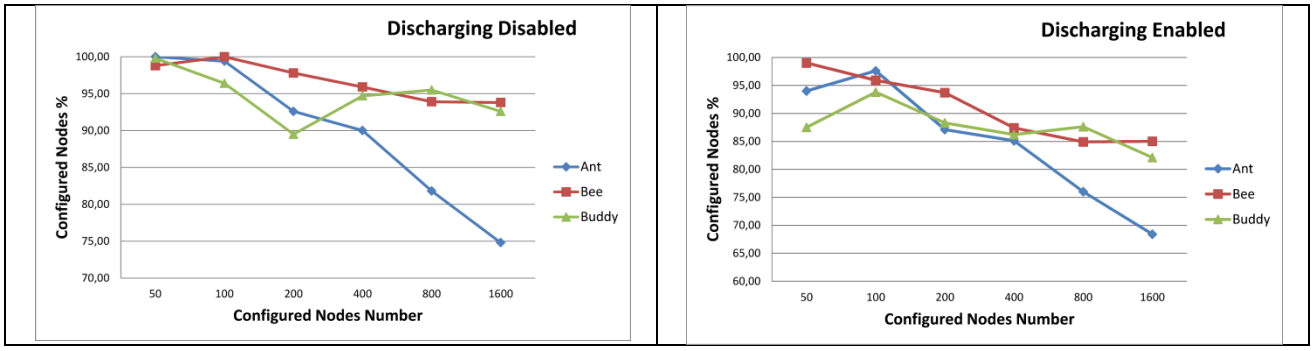
TABLE II. NUMBER OF CONNECTED NODES BY AUTOBEECONF, ANTCONF AND BUDDYCONF

AutoBeeConf			*AntConf*			*BuddyConf*		
Connected nodes			*Connected nodes*			*Connected nodes*		
Nodes	**No Discharging**	**Discharging**	**Nodes**	**No Discharging**	**Discharging**	**Nodes**	**No Discharging**	**Discharging**
50	98,80	99,00	50	100,00	94,00	50	99,80	87,50
100	100,00	95,90	100	99,40	97,60	100	96,40	93,80
200	97,80	93,70	200	92,60	87,10	200	89,50	88,30
400	95,90	87,40	400	90,00	85,10	400	94,70	86,20
800	93,90	84,90	800	81,80	76,00	800	95,50	87,60
1600	93,80	85,00	1600	74,80	68,40	1600	92,60	82,10

TABLE III. CONNECTION TIMES FOR AUTOBEECONF, ANTCONF AND BUDDYCONF

AutoBeeConf			*AntConf*			*BuddyConf*		
Time			*Time*			*Time*		
Nodes	**No Discharging**	**Discharging**	**Nodes**	**No Discharging**	**Discharging**	**Nodes**	**No Discharging**	**Discharging**
50	59,40	61,00	50	57,00	59,40	50	119,90	93,60
100	109,10	111,30	100	109,30	108,00	100	209,10	199,60
200	224,00	223,90	200	215,50	216,60	200	399,00	450,00
400	461,00	429,00	400	433,60	428,40	400	798,10	717,10
800	1142,50	931,70	800	1217,40	1061,20	800	1994,40	1935,30
1600	2442,50	2159,20	1600	3652,20	3183,40	1600	5958,10	5930,80

TABLE IV.



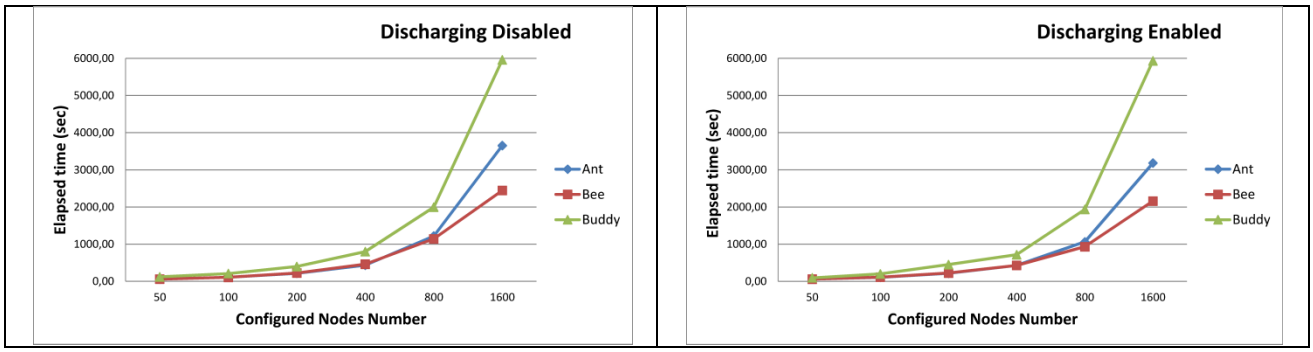
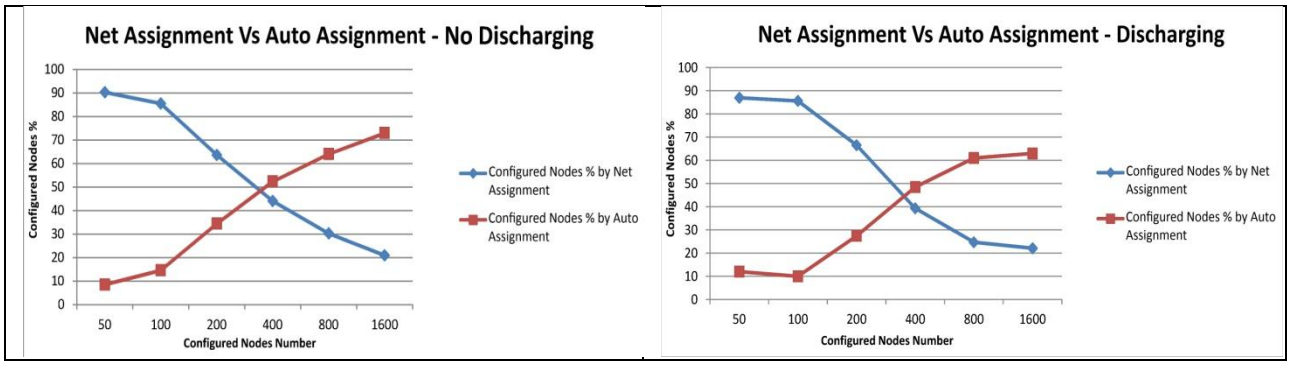

Discharging Disabled
Elapsed time (sec)
6000,00
5000,00
4000,00
3000,00
2000,00
1000,00
0,00
50
100
200
400
800
1600
Configured Nodes Number
Ant
Bee
Buddy
Discharging Enabled
Elapsed time (sec)
6000,00
5000,00
4000,00
3000,00
2000,00
1000,00
0,00
50
100
200
400
800
1600
Configured Nodes Number
Ant
Bee
Buddy

TABLE V.


Net Assignment Vs Auto Assignment - No Discharging
Configured Nodes %
100
90
80
70
60
50
40
30
20
10
0
50
100
200
400
800
1600
Configured Nodes Number
Configured Nodes % by Net Assignment
Configured Nodes % by Auto Assignment
Net Assignment Vs Auto Assignment - Discharging
Configured Nodes %
100
90
80
70
60
50
40
30
20
10
0
50
100
200
400
800
1600
Configured Nodes Number
Configured Nodes % by Net Assignment
Configured Nodes % by Auto Assignment