Evaluating Self-Addressing Protocols for Ad-Hoc Networks

Ricardo de O. Schmidt*, Aiko Pras* and Reinaldo Gomes†
*Design and Analysis of Communication Systems
University of Twente, The Netherlands
{r.schmidt, a.pras}@utwente.nl
†Systems and Computing Department
Federal University of Campina Grande, Brazil
reinaldo@dsc.ufcg.edu.br

***Abstract*—Ad-hoc networks are supposed to operate autonomously and, therefore, self-* technologies are fundamental to their deployment. Many of these solutions have been proposed during the last few years, covering several layers and functionalities of networking systems. Addressing can be considered as one of the critical operations given the support it provides to other operations such as routing in IP-based networks. This paper has the goal of analyzing different strategies of addressing in ad-hoc networks. Five self-addressing protocols were evaluated and their strenghts and drawbacks identified. It is not our goal, in this paper, to come up with a new proposal for addressing autonomous networks, but to evaluate existing ones in a non-isolated manner. We considered critical situations of ad-hoc networks, like partition and merging. Conclusions about the evaluated protocols are drawn at the end of this paper.**

I. Introduction

Autonomous networking systems represent a new paradigm mostly based on an innovative cooperation model of network devices to create and manage communication environments automatically. The idea of creating autonomous networking systems relies heavily on the concept of autoconfiguration. The autoconfiguration problem can be seen as the main reason for the emergence of self-* technologies. Due to its importance for the proper operation of the network, addressing can be seen as one of the most important (and challenging) issues in self-* solutions.

According to documents, like [1], published within the IETF working group AUTOCONF [2], among the goals of autoconfiguration in dynamic networks, like ad-hoc networks, we must consider the configuration of unique address for nodes. The well known DHCP (*Dynamic Host Configuration Protocol*) has a very limited applicability when considering dynamic characteristics found in these networks, such as mobility, unpredictable number of nodes, and undefined topology. Addressing is a critical functionality in IP networks given that it provides basic configuration for other network functionalities such as routing.

Several self-addressing protocols were already proposed for ad-hoc networks, like the ones surveyed in [3] and [4]. These solutions implement different strategies to allow autoconfiguration of nodes interfaces with tentative of valid and unique addresses within the network. Such methodologies range from simple random selection of addresses out of predefined addressing spaces, to mathematical efforts where equations are defined allowing nodes to calculate their own addresses. Self-addressing protocols can be roughly classified as stateless, stateful or hybrid approaches. Each one of these categories is detailed in the next section.

The main goal of this paper is to contribute to the specific area of self-addressing. We implemented and compared five different self-addressing protocols in critical ad-hoc networking situations. Previous works, available in the literature, have already evaluated self-addressing protocols. However, most of them evaluate protocols to very simple network situations that may not reflect situation of real-world scenarios. It is important to state that it is not our intention, in this paper, to propose a new self-addressing protocol, but to evaluate different strategies that we believe to be interesting and promising approaches for such problem. Results from this research, and also from further experiments, will be used in the conception of a new proposal for self-addressing in ad-hoc and temporary networks. To do so, we consider to use, extend and/or combine existent strategies, and also develop new ones when necessary.

The rest of this paper is organized as follow: self-addressing is briefly introduced in the next section; the methodology used in our experiments and a short description of the evaluated protocols are presented in Section III; the results obtained from simulations are presented and discussed in Section IV; and, in Section V, conclusions about the evaluated protocols are drawn.

This is a modified version of the paper presented at the 17th EUNICE Workshop 2011 and published in Springer Verlag LNCS 6955, ISBN 978-3-642-23540-5, 2011.

II. Self-Addressing for Ad-Hoc Networks

Self-addressing is tightly related to the concepts of autoconfiguration. It can be part of a set of technologies that allows a network to operate autonomously. Basically, a self-addressing protocol must provide a node the ability of generating its own address, or the ways for retriving such configuration from another network entity (e.g., addressing server or addressing authority). Self-addressing protocols can roughly be classified in one of the following categories: stateless, stateful or hybrid.

In short, stateless approaches like StrongDAD [5] are those where nodes do not keep track of addresses in use in the network (i.e., the state of addresses is unknon). For instance, with a stateless protocol, a node can randomly select an address from a predefined addressing space, and test this address within the network as an attempt to guarantee uniqueness (i.e., no other node is already configured with the selected address). This testing procedure is generally named Duplicate Address Detection (DAD). On the other hand, stateful approaches like SooA [6], allow nodes to be aware of addresses state. Usually, with a stateful protocol, nodes that keep track of addresses are also responsible for performing addressing tasks on assignment and management of resources. Information about addresses state can, for example, be stored in tables. Alternative stateful protocols, like Prophet Allocation [7], define mathematical approaches that allow nodes to track addresses in use and foresee future allocations. Finally, hybrid approaches mix properties of both stateless and stateful strategies. Hybrid solutions usually implement random address selection, DAD testing, and the address registration within one or more addressing authorities (like in HCQA [8]) or in tables spread through the network (like in MANETconf [9]).

III. Experiments

A. Evaluated Protocols

Five self-addressing protocols were compared in our experiments. Three stateless protocols: StrongDAD, AIPAC and AROD; one stateful protocol: Prophet Allocation; and one hybrid protocol: MANETconf. AROD and AIPAC were fully evaluated, which means that we also considered their mechanisms for handling networks merging and partition. All protocols were implemented in C++ and added to the network simulator ns-2 (Network Simulator 2) [10].

We opted for these five protocols due to the differences in their strategies and also because they implement consistent addressing operations. Aiming a fair comparison among the protocols, we deployed them in the same network scenarios, disregarding their advanced implementations for handling complex network situations. A brief description of each protocol is given next.

StrongDAD [5] is a stateless addressing protocol which implements random selection of addresses followed by DAD procedures. Each starting node is responsible for its own address configuration. On starting, a node randomly selects two addresses form a predefined space: one temporary address and a tentative address. The former is used as the node's identification during the DAD procedure that is executed to test the latter. On succeding in the DAD procedure, the tentative address is used to configure the node's interface. Otherwise, the entire procedure is repeated until the testing procedure is successfully executed. StrongDAD does not handle partition and merging of networks. Consequently, its deployment in more critical scenarios is restricted.

AIPAC [11] operates with requester and initiator nodes. The former is a starting node and the latter an already configured node. The initiator node negotiates an address within the network on behalf of the requester. To communicate with its initiator, the requester uses a temporary address, which is discarded when it receives a negotiated one. For the negotiation procedure, the initiator node randomly selects an address from a predefined space and tests it with the network with DAD procedure (similarly to StrongDAD). AIPAC defines functions to handle partition and merging of networks. A network identification defined by the very first node, is added to the protocol's messages, enabling nodes to identify messages from different networks. AIPAC's main drawback is that it depends on messages and tables of the routing protocol, even requiring modifications in the routing protocol operations. It makes AIPAC unsuitable for networks where routing may also be dynamic.

AROD [12], although being also based on DAD procedure and requester-initiator scheme, focuses on reducing configuration time and control overhead. To do so, it implements address reservation and optimistic DAD. The main difference is that an initiator node may have reserved addresses, previously validated within the network. These addresses can be allocated to a requester node without executing DAD, reducing the configuration time. A DAD procedure is usually executed for more than a single tentative address at the same time, which allow nodes to obtain spare addresses for future configurations of requester nodes.

Prophet Allocation [7] does not implement DAD testing during the allocation procedure. Prophet Allocation also operates with requester and initiator nodes. However, an initiator node obtains a valid address through a formula, and not random selection. This formula takes advantage of the uniqueness properties of prime numbers. Everynode is configured with a 2-tuple identifier, composed by their address and the formula's state. Everytime an address is calculated through the formula, the state is incremented by one. However, as stated by the authors, this formula might generate duplicated addresses within a network. Therefore, when a node starts it receives an address from an initiator in a message that also contains an identifier. This identifier is used to detect merging of networks. Unfortunately, no additional information on how the pro-

tocol handles merging is provided by the authors in [7]. Consequenly, due to the missing information, we did not considered partition and merging of networks in Prophet Allocation.

Finally, MANETconf [9] is a hybrid self-addressing protocol, which implements allocation tables and DAD procedure. MANETconf operation is simple but mainly broadcast-based for DAD and tables synchronization. This protocol also works requester and initiator nodes. The main difference between this and the previous protocols is that, after negotiating an address on behalf of a requester node, the initiator floods the network with the allocated address so that other nodes can update their tables. Allocation tables are used to increase the reliability of DAD procedures. Although MANETconf constantly floods the network with broadcast messages, it does not depend in any other technology.

B. Scenarios and Evaluation Metrics

The experiments were planned to evaluate the implemented protocols under critical situations of ad-hoc networks. The protocols were set to operate with very limited addressing resources (256 available addresses) and submitted to conditions of random deployment and mobility of nodes, leading to situations of networks isolation, partition and merging. Two different scenarios were simulated, as described below.

In *Scenario A*, 100 static nodes were randomly positioned in a predefined area, forcing situations of nodes isolation in the beginning of the simulation (i.e., due to geographical distance between nodes). In this scenario, merging was resulted from the deployment of intermediate nodes between the isolated ones. In *Scenario B*, two initially isolated networks, with 10 nodes each, were merged resulting in a network with 20 nodes. The configuration of the initial 10-node networks did not suffer from nodes isolation and, therefore, merging was resulted only from moving the networks toward each other. Simulation time and nodes arrival rate did not play an important role in our scenarios. In addition, from previous works, like [6] and [12], we learned that variations on addressing space do not significantly impact on the protocol's operations.

To determine the necessary number of simulation runs, we used a methodology described in [13]. This ways our results would fall sufficiently close to the mean, in a confidence interval of 95%. This guarantees that we perform a fair comparison between two or more protocols due to random variables and parameters (e.g., nodes positioning and mobility).

To evaluate the protocols, we considered three basic metrics: (a) control overhead, which quantifies the data generated and transmitted during the protocols operation; (b) configuration delay, which represents the time between the node deployment and its final configuration; and (c) address uniqueness, which measures the efficiency of protocols on avoiding problems with conflicting configurations.

IV. Results Analysis

A. Scenario A: Highly Populated Network

The graphics in Fig. 1 present the traffic generated and transmitted by the protocols while configuring all 100 nodes. However, the lines do not represent all the traffic generated during the entire simulation time, which was 2000 seconds, but the traffic from the beginning of the simulation to the configuration of the last node. Regarding the traffic in number of packets, as presented in Fig. 1(a), and the traffic in number of bytes, as presented in Fig. 1(b), one can conclude in a first moment that the best performances were achieved by the protocols StrongDAD and AROD. However, we must consider that the former implements only allocation of addresses and no further maintenance operations.

StrongDAD stops its operation just after configuring the last node. The traffic generated by this protocol is related to allocation procedure and DAD only. Therefore, if n nodes are configured with duplicated addresses, such problem will reamin unsolved until the end of the simulation. Although it is a broadcast-based approach, StrongDAD generated significantly less traffic than the other protocols. It happened because of the fragmentation of the network in the early moments of simulation, where the broadcast was not propagated throughout the entire network. In addition, the numbers of bytes and packets transmitted by StrongDAD grow in different rates. With DAD requests, the more hops a message is forwarded through, the bigger is the message's header, because the protocol keeps a list of hops so that a reply can be sent to the message's originator in case of a conflict is detected. It is also the case to the other protocols that implement DAD procedures.

As one can observe in Fig. 1, regarding traffic overhead, AROD performed better that StrongDAD. The main reason for this is the address reservation strategy, where AROD tested more than a single address at the same cost than testing only one in StrongDAD. AROD used the reserved addresses to configure new nodes without executing DAD. It is important to inform that in our simulations, due to the limited amount of available addresses, we configured AROD to allow nodes to reserve only one address. Obviously, the protocols performance can be improved if the addressing space is larger and nodes are allowed to reserve more addresses.

Prophet Allocation, which does not implement DAD procedure on address allocation, did not generated high overhead to configure nodes. The simple allocation procedure allows the protocol to quickly configure new nodes with two messages exchanged between requester and initiator. Given that in this experiments we did not considered cooperation between addressing and routing protocols, the strategy for address maintenance implemented by Prophet Allocation was the responsible for the most overhead. The periodical ack, broadcasted by all nodes, was the reason for the exceeding traffic of Prophet Alloca-

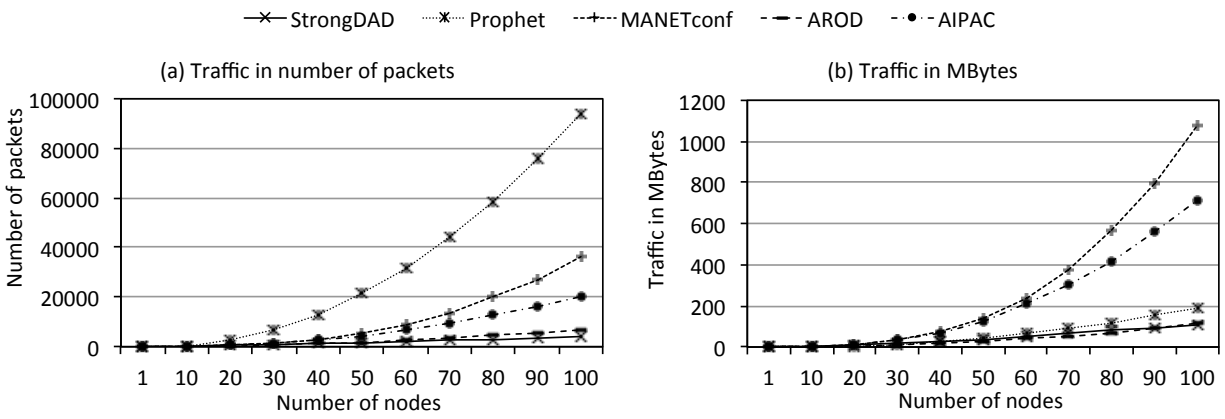

Fig. 1. Protocols performance in *Scenario A*

tion. In addition, due to the small payload of ack messages, the numbers of packets is much higher than the number of bytes, as observed in Fig. 1.

The amount of traffic generated by AIPAC when configuring nodes is similar to StrongDAD. However, the exceeding traffic generated by AIPAC, as observed in Fig. 1, is due to the procedure called "gradual merging". MANETconf generated most traffic when configuring nodes due to the excessive flooding during DAD and tables synchronization. MANETconf defines that nodes must reply positively or negatively to every received request. It results in a very high control overhead due to the increasing amount of replies transmitted as the network grows.

Due to address management, Prophet Allocation, AROD and AIPAC continued to generate traffic even after the configuration of the last node. However, simulation parameters were set so that the configuration of the last node would happen close to the end of the simulation. This way, additional traffic for these protocols did not impact in the final results and, therefore, were not accounted.

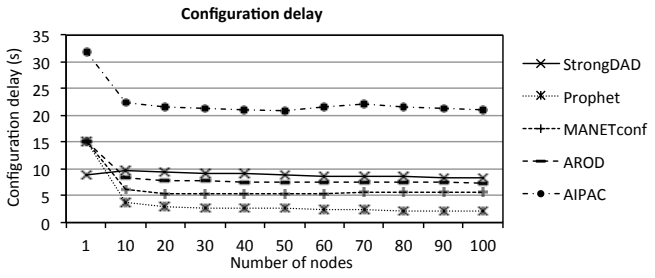

Fig. 2. Configuration delay

Fig. 2 illustrates the average delay achieved by the protocols for configuring the nodes in *Scenario A*. StrongDAD configured nodes with an average of ≈15 seconds. However, SooA configured isolated nodes, in the beginning of the simulation, without executing DAD and, consequently, the average delay at the end of the simulation dropped to ≈9 seconds.

MANETconf's configuration delay strongly depends on the duration of the DAD. Consequently, the average delay achieved by MANETconf was ≈5 seconds. In this case, the time for updating the allocation tables was not considered as part of the configuration delay because such procedure is executed after the node is configured. In merging situations, MANETconf solves the conflicts locally and, consequently, it did not significantly impacted on the protocol's performance.

Unlike MANETconf, the highly fragmented network had a strong impact on AROD operation. The average configuration delay achieved by AROD was ≈8 seconds, even with reserved addresses. The control overhead for address and merging management prevented the protocol to achieve lower configuration delay. Although being reliable, the merging procedure required time and generated excessive traffic.

Prophet Allocation achieved the best results reagarding configuration delay due to its simple handshake for address allocation. However, we must consider that the protocol's authors assume that the implemented mathematical equation may, at a certain moment, generate duplicate addresses. One can infer that by executing procedures for conflicts correction, which would be flooding-based DAD, the protocol's performance would suffer from higher control overhead and configuration delay. Considering this, we believe that Prophet Allocation's averages as presented in Fig. 2 would be increased in ≈5 seconds.

AIPAC had the highest configuration delay. Its DAD procedure lasted ≈15 seconds in average. Mainly, two factors contributed for such results. First, although merging problems were corrected later by the protocol, the "gradual merging" was unnecessarily executed several times during nodes configuration. The second reason is that AIPAC, unlike the other protocols, do not consider networks of one node. It means that isolated nodes do not configure themselves if they do not have at least one neighbor. Therefore,

in the beginning of the simulation, isolated nodes waited for neighbors to be deployed next to them. This waiting time increased the average configuration delay for the first nodes, as shown in Fig. 2.

A serious problem identified in this experiment was the number of configured with duplicated addresses. The network fragmentation in the early moments of simulation was the main reason for such conflicts. All evaluated protocols concluded their operations with duplicated addresses. The weak performance of AIPAC, as afore presented, was rewarded by the success achieved on merging a highly fragmented network and finishing the simulation with the lowest number of conflicts: 5 conflicts in average.

When compared to the other protocols, MANETconf also achieved good results on the number of conflicts: an average of 8 conflicts. It is due to the allocation tables implemented in all nodes. As well as AIPAC, the costly performance of MANETconf was justified by the reliability of its configurations at the end of the simulations. StrongDAD configured isolated nodes and did not handle merging situations to correct possible conflicts, which resulted in a very high number of conflicts: 29 conflicts in average.

Prophet Allocation achieved the highest number of conflicts at the end of the simulation: 45 conflicts in average. Given that it does not implement a procedure for handling merging, the fragmented network was also one of the reasons for its bad performance. Also, the protocol started several isolated nodes with different sequences of numbers for its formula, which resulted in many unsolved conflicts. Prophet Allocation can be implemented with more states in its formula, which would generate more distinct sequences of addresses and, according to its authors, less conflicts. However, we decided for implementing its equation with only one state due to our goal of simulating protocols in critical situations.

Although AROD ended the simulation with high number of conflicts (i.e., 30 conflicts in average), it is important to mention that the protocol's procedure for conflict resolution was not implemented in this experiment. However, AROD successfully identified all the conflicts in the network. Such conflicts were resulted from the merging of all isolated nodes and small networks. Considering that AROD implements DAD procedure for resolving address conflicts, triggering such operation would degrade the protocol's performance by increasing control overhead and configuration delay. This leads us to conclude that it would bring AROD's results close to AIPAC's.

B. Scenario B: Handling Networks Merging

Past works have evaluated self-addressing protocols under more controlled networking scenarios, considering mainly the allocation procedure and not the resources management after allocation. However, real ad-hoc network scenarios may be quite different. In these, nodes arbitrarily come and go. In some situations not only the nodes move, but entire networks may move from one place to another, temporarily or permanently merging with each other. In *Scenario B*, we tested the merging procedures of AROD and AIPAC in a scenario of merging between two networks. Each network was composed by 10 nodes. In a predefined moment, after the initial configuration of all nodes, network *N1* moved towards network *N2*, merging and creating a single network *N3* with 20 nodes.

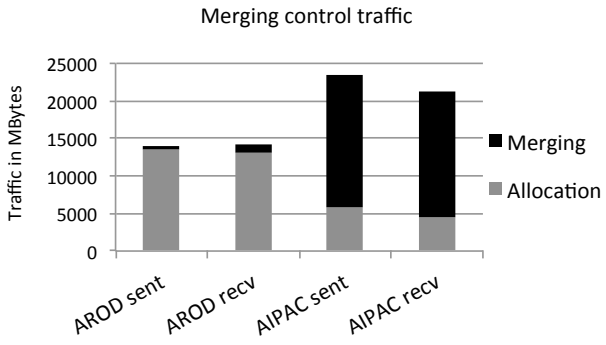

Fig. 3. Merging traffic Vs. Allocation traffic

AROD and AIPAC successfully performed network merging in *Scenario B*. The difference between the strategies implemented by the protocols for handling merging is clearly observed on the results presented in Fig. 3. The conflict resolution procedures were not considered in this experiment, but it is planned as future work.

As soon as AROD identifies that two networks start to overlap, the mechanism for merging is triggered and a single network is quickly formed. The merging procedure in AROD affects all the nodes in one of the overlapping networks. On identifying that it is necessary to join the other network, a node announce the merging to its network's leader and the latter floods the network with a reconfiguration message. It is a quick process where the entire network is reconfigured at once, and does not generate high control overhead. As one can observe in Fig. 3, in average only ≈5% of the total traffic generated by AROD was resulted from the merging procedure.

On the other hand, AIPAC needs more time for executing the merging of networks. In addition, as illustrated in Fig. 3, if compared to AROD, a much higher percentage of its traffic is related to the merging procedure. After successfully merging the two networks, the traffic related to merging accounted for ≈80% of the total traffic generated by the protocol. It was due to the "gradual merging" strategy implemented in AIPAC, where some nodes migrated to the other network and later migrated back to their original network. Therefore, such situations doubled the traffic generated by these nodes during the merging procedure.

Regarding the procedure delay, AROD detected and performed merging faster than AIPAC. AROD started the

merging procedure as soon as the first nodes identified the networks overlap. AROD needed in average ≈40 seconds for completing the merging process. With the "gradual merging" strategy, AIPAC took longer for detecting the merging and concluded the procedure with and average delay of ≈94 seconds.

V. Conclusion

Several self-addressing solutions have been proposed for ad-hoc networks. However, from experiments like the one presented in this paper, we can observe that such solutions lack on providing alternatives for handling complex networking situations. Addresses consistency is also very important due to its influence on the correct operation of, for example, routing. However, proposed strategies handle such situations with excessive control overhead and degradation on performance of basic addressing operations (i.e., allocation). In addition, dependency on other technologies is not a good approach for technologies that are designed to operate in autonomous networks. For instance, addressing solutions like AIPAC and Prophet Allocation, which depend on routing protocols, and even require modifications on routing operations, have their applicability drastically limited to scenarios with the specific routing protocol. It becomes more problematic if we consider scenarios where routing is also a dynamic operation and two or more routing protocols may coexist and cooperate. In the specific experiments of this work, the protocol AROD achieved the best overall performance by allocation and managing addresses in all situations, without suffering from excessive overhead or configuration delay. Moreover, AROD operates independently, without requiring supporting technologies.

This paper presented the first steps on evaluation of self-addressing approaches. For future work, the authors plan more extensive and complete simulations with self-addressing solutions, also covering other network scenarios and situations. Our main goals are: to contribute with the decision on the best general self-addressing protocol (or the combination of two or more solutions) for different ad-hoc networks; and to propose an alternative self-addressing solution for SooA [6], that is going to be used on the absence of addressing servers in temporary and ad-hoc networks.

References

[1] E. Baccelli, *Address Autoconfiguration for MANET: Terminology and Problem Statement*. IETF Internet Draft, 2008.

[2] AUTOCONF, *Ad-Hoc Network Autoconfiguration*. IETF Working Group. Available in: http://www.ietf.org. Accessed in: Sep. 2011.

[3] C. Bernardos, M. Calderon and H. Moustafa, *Ad-Hoc IP Autoconfiguration Solution Space Analysis*. IETF Internet Draft, 2008.

[4] K. Weniger and M. Zitterbart, *Address autoconfiguration in mobile ad hoc networks: current approaches and future directions*. IEEE Network Magazine, Special Issue on Ad Hoc Networking: Data Communications & Topology Control, volume 18, issue 4, pages 6-11, 2004.

[5] C. E. Perkins, J. T. Malinen, R. Wakikawa, E. M. Belding-Royer and Y. Sun, *IP Address Autoconfiguration for Ad Hoc Networks*. IETF Internet Draft, 2001.

[6] R. de O. Schmidt, R. Gomes, D. Sadok, J. Kelner and M. Johnsson, *An Autonomous Addressing Mechanism as Support for Autoconfiguration in Dynamic Networks*. In proceedings of the Latin American Network Operations and Management Symposium (LANOMS), 2009.

[7] H. Zhou, L. M. Ni and M. W. Mutka, *Prophet Address Allocation for Large Scale MANETs*. In proceedings of the 22nd Annual Joint Conference of IEEE Computer and Communication Societies (INFOCOM), volume 2, pages 1304-1311, 2003.

[8] Y. Sun and M. E. Belding-Royer, *Dynamic Address Configuration in Mobile Ad Hoc Networks*. Technical Report 2003-11, University of California at Santa Barbara, 2003.

[9] S. Nesargi and R. Prakash, *MANETconf: Configuration of Hosts in a Mobile Ad Hoc Network*. In proceedings of the 21st Annual Joint Conference of IEEE Computer and Communication Societies (INFOCOM), volume 2, pages 1059-1068, 2002.

[10] K. Fall and K. Varadhan, *The ns Manual (formerly ns Notes and Documentation)*. The VINT Project. Available at: <http://www.isi.edu>. May, 2010.

[11] M. Fazio, M. Villari and A. Puliafito, *AIPAC: Automatic IP Address Configuration in Mobile Ad Hoc Networks*. Elsevier Computer Communications (COMCOM), volume 29, issue 8, pages 1189-1200, 2006.

[12] N. Kim, S. Ahn and Y. Lee, *AROD: An address autoconfiguration with address reservation and optimistic duplicated address detection for mobile ad hoc networks*. Elsevier Computer Communications (COMCOM), number 30, pages 1913-1925, 2007.

[13] R. Jain, *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation and Modeling*. Wiley-Interscience, New York, NY, ISBN 0471503361, 1991.